Practice Problem Set 2
Prototype

Q1	Midterm 201806 P4	prototype	___ / 12 points
Q2	Midterm 201703 P3	prototype	___ / 12 points
Q3	Midterm 201706 P2	prototype	___ / 10 points
Q4	Midterm 201603 P6	Prototype	___ / 12 points
Q5	填空	Data Tampering	___ / 12 points
Q6	Final 201703 P16	Data Tampering	___ / 12 points
		Total	___ / 70 points

Problem #4 (12 points)

Given the following class and method definitions:

```
function Rectangle(length, width) {
      this.length = length;
      this.width = width;
}

Rectangle.prototype.hypotenuse = function () {
      var a_sq = Math.pow(this.length, 2);
      var b_sq = Math.pow(this.width, 2);
      var c_sq = a_sq + b_sq;
      return Math.sqrt(c_sq);
}

Rectangle.area = function() {
      return this.length * this.width;
}

var myVeryOwnRect = new Rectangle(5, 10);
```

Part (A): Please select which of the following will throw an error (there may be more than one). Briefly justify each selection.

```
A. myVeryOwnRect.area();
B. myVeryOwnRect.hypotenuse();
C. Rectangle.area();
D. Rectangle.hypotenuse(myVeryOwnRect);
```

… continued from previous page..

Part (B): It is not uncommon in JavaScript method functions to have a body whose first line is:

```
var self = this;
```

The remainder of the function body then uses `self` rather than `this` to access the object. Describe both the language feature of JavaScript and the usage that makes this a useful convention.

Problem #2 (12 points)

```
var object = {numProp: 1, stringProp: "foo", obj1Prop: {prop:
'foo'}};
object.__proto__ = { stringProp: "bar", obj1Prop: {prop: 'bar'},
   obj2Prop: {}};
print("1", object); print("1p", object.__proto__);

object.numProp = 2;
object.stringProp = "foo2";
object.obj1Prop.prop = "prop2";
object.obj2Prop.prop = "prop2"

print("2", object); print("2p", object.__proto__);

function print(tag, obj) {
  console.log(tag, "numProp", obj.numProp,
     "stringProp", obj.stringProp, "obj1Prop", obj.obj1Prop,
      "obj2Prop", obj.obj2Prop);
}
```

When the above code is executed it prints four lines to the console log. Each line contains four properties from either the object or its prototype. Fill in what this code would output in the form below. Assume console.log prints the entire object (e.g. `console.log({prop: 'foo'})` prints `'{prop: "foo"}'`.
Hint: The property __proto__ returns the prototype of an object.

1 numProp ________ stringProp ________ obj1Prop ________________ obj2Prop ________________

1p numProp _______ stringProp ________ obj1Prop ________________ obj2Prop ________________

2 numProp ________ stringProp ________ obj1Prop ________________ obj2Prop ________________

2p numProp _______ stringProp ________ obj1Prop ________________ obj2Prop ________________

Problem 2 continued…

```
var globalVar = 1;
function foo(argVar) {
  var localVar = 2;

  return function (arg) {
            argVar += arg;
            localVar += arg;
            globalVar += arg;
            console.log('F', argVar, localVar, globalVar);
        };
}

var func1 = foo(1);
var func10 = foo(10);
func1(1);
func10(2);
console.log('G', globalVar);
```

Show what the above code will output to the console log when executed.

Problem #6 (10 points)

The following JavaScript program prints some lines to the console log. In the places indicated below, describe what is printed to the log.

```
var gColor = 'blue';
function CrayonStore() {
  var color = 'red';
  this.getFunc = function() {
    this.color = 'purple';
    return function() {
      console.log(this && this.color);
      console.log(color);
      console.log(gColor);
    };
  };
  this.getFuncObject = function() {
    var colors = ['green', 'blue'];
    var retObj = {};
    for (var idx = 0; idx < colors.length; idx++) {
      retObj[colors[idx]] = function() { console.log(colors[idx]); };
    }
    return retObj;
  };
}
var crayonStore = new CrayonStore();
var func = crayonStore.getFunc();
func();  // (a) What is printed by this call?



gColor = 'green';
crayonStore.color = 'black';
func();  // (b) What is printed by this call?



var objectFunc = crayonStore.getFuncObject();
objectFunc.green();  // (c) What is printed by this call?


objectFunc.blue();  // (d) What is printed by this call?
```

Problem #7 (12 points)

(a) Using the tree structure of the DOM it is possible to navigate to an arbitrary DOM node by starting at the root of the tree and traversing the tree properties (e.g. `document.body.firstChild.nextSibling.firstChild`). Explain why this kind of navigation is not the preferred way of getting to DOM nodes.

(b) In the lecture notes we mentioned that for a DOM node the `offsetParent` is not necessarily the same as `parentNode`. Use a code fragment to describe something you could do in HTML and/or CSS that would cause these two properties to be different.

Problem 7 continued on next page ….

Problem #6 (8 points)

Consider the following JavaScript:

```
function Jeep() {
    this.organization = "DaimlerChrysler";
}
Jeep.prototype.color = "blue";

var blue = new Jeep();

console.log(blue.organization);      // Line A
console.log(blue.color);              // Line B

var red = new Jeep();
red.color = "red";

console.log(red.organization);      // Line C
console.log(red.color);              // Line D

Jeep.organization = "Chrysler";
Jeep.prototype.color = "green";

console.log(blue.organization);  // Line E
console.log(blue.color);         //  Line F
console.log(red.organization);   // Line G
console.log(red.color);          // Line  H
```

In the table below list what is printed at each console.log statement:

console.log statement	What is printed by the console.log
Line A	
Line B	
Line C	
Line D	
Line E	
Line F	
Line G	
Line H	

Problem 5: Data Tampering Detection

Sender-recipient communication
Step 1: sender computes ___ using ___ and ___.
Step 2: sender sent ___ and ___ to recipient.
Step 3: recipient computes ___ using ___ and ___.
Step 4: recipient compares when ___ from step ___ and ___ from step ___.

Server-browser communication
Step 1: server computes ___ using ___ and ___.
Step 2: server responds to HTTP requests. Server sends ___ and ___ to browsers.
Step 3: browsers return ___ and ___ to browsers in its subsequent requests.
Step 4: server validates computes ___ using ___ and ___.
Step 5: server validates requests when ___ from step ___ and ___ from step ___ matches.

Possible choices:

(A) MAC
(B) Data
(C) key (cipher)

Final 201703

Problem #16 (12 points)

In general any information coming in over the Internet to the backend of an web application must be treated as suspect by the backend. Even if the information is known to come from the web application's code running in a user's browser, an attacker might have hacked the browser to view or manipulate the application's data or communication to the backend.

Because of this lack of trust in the web frontend environment, having the web app frontend hold information for the backend can be tricky. For example, if the backend passes some information to the frontend the attacker in the browser could view, delete, or modify the information, or even create fake versions of the data.

1. Describe a mechanism covered in class that would allow the backend to send information to the frontend and when it came back the backend could tell if it had been modified or faked.

2. Describe a mechanism covered in class that would allow the backend to send information to the frontend and both detect if it was changed and be confident an attacker wasn't able to view it.