IT Workshop II

Rekha Singhal

[2018-02-06 Tuesday]

Total Time :: 1.5 hrs **Total Points** :: 90

Identification :: Write your roll number on the question paper.

Reasonable assumptions :: No questions will be entertained during the exam. If you have a doubt, proceed after making reasonable assumptions.

1 General [10 pts]

1. Who is known as "Father of Computer"? [**2 pts**]
    - A. Charles Babbage
    - B. Albert Einstein
    - C. Bill Gates
    - D. Thomas Edison
2. Key features of Unix operating system are [**2 pts**]
    - A. Multitasking and multiuser
    - B. Very efficient virual memory and lean kernel
    - C. High portability and machine independence
    - D. All of the above
3. Bash is a scripting language for Unix shell. It stands for [**2 pts**]
    - A. Basic shell
    - B. Bourne Again shell
    - C. Big App shell
    - D. None of the above

4. Communication between client and server has following steps. Place them in right order. [**2 pts**]

   - x. Client sends request for data to server.
   - y. Client establishes the connection with server using HTTP protocol.
   - z. Client gets back a response from server

5. Process creation in Unix is unique because [**2 pts**]

   - A. It has single step of spawning the process.
   - B. It has 2 steps: Exec clones the parent process while fork overlays the new executable in newly created process.
   - C. It has 2 steps: Fork clones the parent process while exec overlays the new executable in forked process.
   - D. None of the above

2 Javascript [10 points]

1. When Javascript was intiallaly developed, it was called [**2 pts**]

   - A. Java
   - B. Java Beans
   - C.LiveScipt
   - D. None of the Above

2. In Javascript functions are [**2 pts**]

   - A. First class objects.
   - B. Second class objects
   - C. not objects
   - D.None of the Above

3. In Javascript, what kind of typing system is used? [**2 pts**]

   - A. Static typing
   - B. Dynamic typing
   - C. Basic typing

- D. None of the above

4. Inheritance model used in Javascript is [**2 pts**]
   - A. Class Based
   - B. Prototypal
   - C. Classical
   - D. None of the above
5. Promise is used to address following problem. [**2 pts**]
   - A. Efficiency Issues
   - B. Closure Bug
   - C. Callback hell
   - D. None of the above

3 Scope [8 points]

- A. Anonymous functions do not have any name and can be invoked right after they are defined. What will be the outcome for following anonymous function invocations?[**4 pts**]

I. [**2 pts**]

```
(function (){
     var x  = 4;
     var f = function(x){
                 return x + 5;
             };
     console.log(x);
     console.log(f(x));})();
```

II. [**2 pts**]

```
(function (){
     var x  = 4;
     var f = function(x){
                 var x = -3;
                 return x + 5;
             };
     console.log(x);
     console.log(f(x));})();
```

- B. `var` construct has function scope while `let` construct has block scope. What is the console output for the following code? [**4 pts**]

I. [**2 pts**]

```
function foo(){
   var x = 5;
   {
      var x = 6;
      console.log (x);
   }
   console.log(x);
};
foo();
```

II. [**2 pts**]

```
function foo(){
    var x = 5;
    {
       let x = 6;
       console.log (x);
    }
    console.log(x);
};
foo();
```

4 Closure [14 points]

- A. In a function, free variables are bound to lexical environment creating closures. [**2 pts**]
  - I. What is the console output for following code?
  - II. What would be the output if runtime dynamic environment is used to bind the free variables?

```
function foo(){
    var x = 20;
    return function(){
              console.log(x);
           }
};
var x = -20;
var printx = foo();
printx();
```

- B. In following sample code, what will be the output on console for numbered lines? [**5 pts**]

```
function getAdder(initialVal){
            var counter = initialVal;
            return function(){
                      counter +=1;
                      console.log(counter);
                   };
};
   var add = getAdder(3);
1. add();
2. add();
3. add();
   var add2 = getAdder(-3);
4. add2();
5. add();
```

- C. When closure captures a loop variable, it might give you unexpected outcome. What is the console output for code below?[**2 pts**]

```
(function foo(){
    var arr = [];
    for (var i=0; i<2; i++){
       arr[i]= function(){return i;};
    }
    console.log(arr[0]());
    console.log(arr[1]());
})();
```

- D. How would you change the above code to create array of functions which when invoked returns varying value of i? [**3 pts**]

- E. In closure, free variables are lexically bound to the references

not to the values. What is the console output for code below?[**2 pts**]

```
function foo() {
      var y = 2;
      var ret = function() {return y;}
      y = 10;
      return ret;
};
var f = foo();
console.log (f());
console.log((foo())());
```

5 This [10 points]

- A. What will be the outcome at each numbered line when you pass the following statements through javascript interpreter?[**6 pts**]

```
(assume global scope here.)
 1. var x = 5; console.log(x);
 2. console.log(this.x);
 3. console.log(x == this.x);
    function foo (){ var x = -1;
 4.                  console.log(this.x);
                     this.x = -2;
 5.                  console.log(x);};
    foo();
 6. console.log(this.x);
```

- B. A function is called a constructor when its called from `new` operator and it then populates the fields of newly created object which is finally returned by new operator. What will be the outcome in code below when a function is called in 2 different ways - as a constructor and as a simple function?[**4 pts**]

```
    var Point = function (x, y) {this.x = x; this.y = y;};
    var myObj1 = new Point(2, 3);
1.  console.log(myObj1.x + myObj1.y);
2.  console.log(this.x + this.y);
    var myObj2 = Point(-2, -3);
3.  console.log(myObj2.x);
4.  console.log(this.x + this.y);
```

6 Object inheritance [10 points]

An object's prototype is represented by its internal property`__proto__` which points to prototype object saved in its constructor's `prototype` property. When a property is searched on an object, first object is searched, if not found then its prototype (pointed by its `__proto__` or its constructor's prototype property.) is searched. What will be the console output for following numbered lines?

```
    var Point = function(x, y){
                    this.x = x;
                    this.y =y;
                 }
    var a = new Point(2, 3);
1.  console.log(a.constructor);
    Point.prototype.color = "red";
2.  console.log(a.color);
3.  console.log(Point.color);
4.  console.log(Point.prototype.color);
5.  console.log(a.__proto__ == Point.prototype);
6.  console.log(a.hasOwnProperty('x'));
7.  console.log(a.hasOwnProperty('color'));
8.  console.log(Point.prototype.hasOwnProperty('x'));
9.  console.log(Point.prototype.hasOwnProperty('color'));
10. console.log(Point.__proto__ == Function.prototype);
```

7 Arrow [8 points]

Arrow functions provide shorter syntax for functions and ensure lexical "this". What will be the console output for these 2 sample code?

I. [**4 points**]

```
this.x = "Hello world!"
var myobj = {};
var foo = function(){ console.log(this.x); };
myobj.f = foo;
//function foo is called as method
myobj.f();
//function foo is called as non-method
foo();
```

II. [**4 points**]

```
this.x = "Hello world!"
var myobj = {};
var foo = ()=> console.log(this.x);
myobj.f = foo;
//function foo is called as method
myobj.f();
//function foo is called as non-method
foo();
```

8 Asynchronous functions [20 points]

In Synchronous mode, things always happen in sequential order. In asynchronous mode, things do not necessarily happen in sequential order, specially when execution of things take time and can be done in background making way to complete other tasks.

- A. Standing in queue to buy train ticket is synchronous or asynchronous? [**2 points**]
- B. Standing in queue to purchase your food in a restaurant is synchronous or asynchronous? [**2 points**]
- C. What is the console output for following code, given that setTimeout is an asynchronous function. [**2 points**]

```
function foo(){
   var str = "Hello World";
   var async = function(){setTimeout(function(){ console.log(str); }, 6000)}
   async();
   str = "Life is Great!"
   console.log(str);
}
foo();
```

- D. What will be the console output for above code if setTimeout is made synchronous. [**2 points**]
- E. What is the callback function in above code? [**2 points**]
- F. It becomes difficult to modify a deeply nested callback code. What is this problem called? [**2 points**]

- G. What is the proposed solution to make such code look flat and more manageable? [**2 points**]
- H. What is the console output for following code, given that setTimeout is an asynchronous function? [**6 points**]

```
function timeouts() {
   setTimeout(function() {
                  console.log('1. First thing setting up second thing');
                  setTimeout(function() {
                                 console.log('2. Second thing setting up third thing');
                                 setTimeout(function() {
                                                console.log('3. Third thing completed');
                                 }, 2000);
                                 console.log('Second thing completed');
                  }, 2000);
                  console.log('first thing completed');
   }, 2000);
   console.log("All over???");
}
timeouts();
```