Advanced exam

Total points 24/52

time : 1hr allowed

0 of 2 points

Name *

···/2

Mostafa Ahmed Mohammed Negm

Choose the correct answer

24 of 38 points

what is the output of the below code : * 2/2

```
const person = { name: 'Lydia' };
Object.defineProperty(person, 'age', { value: 21 });
console.log(person);
console.log(Object.keys(person));
```

- a. { name: "Lydia", age: 21 }, ["name", "age"]
- b. { name: "Lydia", age: 21 }, ["name"]
- c. { name: "Lydia"}, ["name", "age"]
- d. { name: "Lydia"}, ["age"]

What is the prototype chain in JavaScript? * 2/2

- a. A chain of objects that are linked through their prototype properties.
- b. A chain of events that occur during the creation of objects
- c. A chain of inheritance where objects inherit properties and methods from their prototypes.
- d. A chain of conditional statements used to determine object behavior.

*

0/2

```
function Counter() {
  this.count = 0;
}

Counter.prototype.increment = function () {
  this.count++;
};
var counterOne = new Counter();
counterOne.increment();
counterOne.increment();

var counterTwo = counterOne;
counterTwo.increment();

console.log(counterOne.count);
```

- (●) 1
- ( ) 2
- ( ) 3
- ( ) 4

the output of L1 will be * 0/2

```
var box={outerW:50};

function bWidth(){
        alert(this.outerW)
} ;

bWidth.apply(box); //L1
bWidth(); //L2
```

- (●) a. undefined
- ( ) b. This will give an error
- ( ) c. outerW=50
- ( ) d. {outerW:50}
- ( ) e. 50
- ( ) f. None of the above

AJAX Stands for: * 2/2

- (●) a. Asynchronous JavaScript and XML
- ( ) b. Abstract JSON and XML
- ( ) c. Another Java Abstraction for X-Windows
- ( ) d. Another Java and XML Library

*

0/2

```
var x = 10;
var y = {
 x: 20,
 getX: function () {
 return this.x;
 }
};
var getX = y.getX;
console.log(getX());
```

- a. 10
- b. 20
- c. undefined
- d. error: getX is not a function

Which of the following is the most common way to achieve inheritance in JavaScript? *2/2

- a. Using the extends keyword (like in Java)
- b. Prototypal inheritance
- c. Class inheritance (using the class keyword)
- d. Interface implementation

*

2/2

```
function Person(firstName, lastName) {
  this.firstName = firstName;
  this.lastName = lastName;
}

var member = new Person('hamada', 'hamada');
Person.getFullName = function() {
  return this.firstName + " " + this.lastName;
};

console.log(member.getFullName());
```

- (●) a. TypeError
- ( ) b. undefined undefined
- ( ) c. "hamada hamada"
- ( ) d. none of the above

what will be the output of the below script in L1 : * 0/2

```
var a={},
b={key:'b'},
c={key:'c'};
a[b]=123;
a[c]=456;

console.log(a[b]); //L1
console.log(a.b); //L2
```

- a. "b"
- b. key:"b"
- c. 123
- d. 456
- e. none of the above

what will be the output of the below script in L2 : * 2/2

```
var a={},
b={key:'b'},
c={key:'c'};
a[b]=123;
a[c]=456;

console.log(a[b]); //L1
console.log(a.b); //L2
```

- a. "b"
- b. key:"b"
- c. 123
- d. 456
- e. none of the above

What does the this keyword refer to within an object method? * 2/2

- a. The object itself
- b. The global window object
- c. The method that is currently being executed
- d. The value of the previous this in the calling scope

*

2/2

```
var person = { name: 'hamada' };

function sayHi(age) {
  return this.name + " is " + age;
}

console.log(sayHi.call(person, 21));
console.log(sayHi.bind(person, 21));
```

- ○ undefined is 21 hamada is 21
- ○ function function
- ○ hamada is 21 hamada is 21
- ◉ hamada is 21 function

what is the output of the below code : * 2/2

```
var person = { name: 'hamada' };
var members = [person];
person = null;
console.log(members);
```

- a. null
- b. [null]
- c. []
- d. none of the above

What is the result of alert(a.fun(2))? *

0/2

```
var a = (function (y,x) {
var x = null;
return {
                fun : function (x) { return this.fun2(x*y);},
                fun2 : function(x) {return x + y;}
        }
} )(3,4)
```

- a. 7
- b. 9
- c. null
- d. 12
- e. NaN
- f. fun1 is not accessed, it is an inner function
- g. None of the above

the output of L2 will be * 2/2

```
var box={outerW:50};

function bWidth(){
        alert(this.outerW)
} ;

bWidth.apply(box); //L1
bWidth(); //L2
```

- (●) a. undefined
- ( ) b. This will give an error
- ( ) c. outerW=50
- ( ) d. {outerW:50}
- ( ) e. 50
- ( ) f. None of the above

what is the output of the below code : * 0/2

```
var xhr = new XMLHttpRequest();
xhr.open("get", "text.txt");
xhr.send();

if(xhr.readyState == 4){
    console.log(xhr.response)
}
```

- a. log text from the file
- b. undefined
- c. Error
- d. None of the above

*

0/2

```
function Person(name) {
  this.name = name;
}

Person.prototype.greet = function () {
  console.log("Hello, my name is " + this.name);
};

var john = new Person("John Doe");
john.greet();

var jane = Object.create(Person.prototype);
jane.name = "Jane Doe";
jane.greet();
```

- a. `john` inherits from `Person.prototype` directly.
- b. `jane` inherits from `Person.prototype` directly.
- (●) c. Both `john` and `jane` inherit from `Person.prototype` directly.
- d. Neither `john` nor `jane` inherit from `Person.prototype` directly.

In the context of XMLHttpRequest, what does the readyState property represent? *2/2

- a. The status code of the HTTP response (e.g., 200, 404)
- (●) b. The state of the request (e.g., uninitialized, loading, complete)
- c. The type of response (e.g., text, JSON)
- d. The number of bytes received so far

*

2/2

```
var a = 3;
var b = new Number(3);
var c = 3;

console.log(a == b);
console.log(a === b);
console.log(b === c);
```

- a. true false true
- b. false false true
- c. true false false
- d. false true true

Answer the following

0 of 12 points

Revise the provided code snippet to **address the error** encountered during execution. *···/6

Refactor the code to prevent errors and ensure that the Cat object can successfully make its sound and knock things without any issues.

Explain the necessary adjustments made to resolve the errors and achieve the intended functionality

```
function Animal(s) {
  this.sound = s;
  this.makeSound = function () {
    return "my sound is " + this.sound;
  };
}

function Cat() {
  Animal.call(this ,s);
}

Cat.prototype.knockThings = function () {
  console.log("cup knocked");
};

Cat.prototype = Object.create(Animal.prototype);

var c = new Cat("meow");
c.knockThings();

var sound = c.makeSound();
console.log(sound);
```

first error is not passing s in cat function and second error is not putting the function in prototype to non creation in each instance and third error is add KnockThing before inherit methods from Animal and missed constructor c.KnockThing display is not a function

solution is

```
Animal.prototype.makeSound = function () {
return "my sound is" + this.sound;
}
function Cat (s) {
Animal.call(this,s)
}
```

```
Cat.prototype = Object.create(Animal.prototype);
Cat.prototype.constructor = Cat;

Cat.prototype.knockThing = function () {
console.log("cup knocked")
}
```

Develop an object featuring a **name** property defined with an data descriptor and an **age** property defined with a accessor descriptor. Enforce validation rules to ensure the name property can only be accessed and not directly modified, while the age property accepts only non-negative numerical values.
Discuss the distinct roles of accessor and data descriptors in maintaining data and controlling access to object properties."

*···/6

```
var obj = {}
Object.defineProperties(obj,{

name : {
writable : false , // default
enumerable : true,
configurable : false,  // default
},
age : {
get : ( function () {
var initial = 1;
return initial
}) () ,

set : function (val) {

if (typeof val != "number") {
    throw `it must be number` ;
}
if (val < 0 ) {
   throw `it must be non-negative value`
}
 initial = val ;
}
}
})
```

data descriptors is for putting constrains that can't edit or delete or looping on property in object
but accessor descriptor is for putting constrains that control get and set properties and also looping and deleting