1、（网站）function Super(name,type){

this.name = name;

this.type = type;

}

Super.prototype = {

typeId:1,

say:function(){

alert(this.name);

}

}

var cat = new Super();

function Bird(n,t){

Super.apply(this,arguments);

}

var dj = new Bird("杜鹃","飞鸟")

console.log(dj.typeId)打印结果是？

A: 1

B: undefined

C: 程序报错

D: 以上都不对

**2.**[单选题](5分)

（网站）以下哪种方法属于通过字面量方式创建对象？

A: var obj = [];

B: var obj = {}

C: var obj = ()

D: var obj = //;

3、function Person () {}

Person.prototype = {

say: function() {

console.log('Hello,' + this.name);

}

};

var person = new Person();针对以上代码，打印结果为false的有（ ）？

A: person.constructor==Person;

B: person.constructor==Object;

C: person instanceof Person==true

D: person instanceof Object==true

4、function Car (Name) {

this.name = Name;

}

Car.prototype = {};

var car = new Car();针对以上代码，打印结果为false的有（ ）？

A: car.\_\_proto\_\_==Car.prototype

B: car.constructor==Object

C: Object.prototype.\_\_proto\_\_==null;

D: Object.\_\_proto\_\_==null;

5、function Foo(y) {

this.y = y;

}

var b = new Foo(20);以下结果为true的有？

A: b.construcotr==Function

B: b.\_\_proto\_\_ === Foo.prototype

C: b.constructor === Foo

D: b instanceof Object

6、（网站）data只有两种可能，要么数组，要么对象字面量，以下检测data到底是不是数组，方法正确的有（ ）？

A: data instanceof Object == true;

B: typeof(data) == Object;

C: data instanceof Array == true;

D: Object.prototype.toString.call(data) == "[object Array]";

7、在查找一个对象的属性时，倘若在当前对象找不到该属性，则会沿着原型链一直往上查找，直到找到为止，如果到了原型链顶端，还没找到，则返回undefined

8、var Classify = function(name){

this.name = name;

}

Classify.prototype.getName = function(){

var name = "lili";

return this.name;

}

var son = new Classify("huahua");

console.log(son.constructor);打印结果是（  Classify 构造函数 ）

9、所有的JavaScript对象，都存在一个( )属性,该属性指向它的构造函数的原型?

* A: \_\_proto\_\_
* B: prototype
* C: constructor
* D: instanceof

10、JS中所有的东西都是对象,所有的东西都由Object衍生而来,所有对象 instanceof Object返回的结果都是？

* A: undefined
* B: false
* C: true
* D: null

11、一列有关面向对象中的原型介绍无误的有（ ）？

* A: 是为每个实例对象存储共享的方法和属性
* B: 所有的实例是共享同一个原型对象，因此有别于实例方法或属性，原型对象仅有一份。
* C: 在js的所有函数，都存在一个属性prototype，该属性对应实例对象的原型。
* D: 可以借助原型实现对象的继承。

12、prototype原型链的存在主要目的是为javaScript实现继承的。对

13、我们创建每一个函数都有一个prototype（原型）属性，这个属性是一个指针，指向一个对象，而这个对象的包含所有实例共享的属性和方法。对

14、javascript中，每个对象有一个proto 属性，当我们访问一个对象属性时，该属性在当前对象中找不到，则去proto 指向的对象里面找，一层一层找下去，这就是javascript原型链的概念。对

15、在查找一个对象的属性时，倘若在当前对象找不到该属性，则会沿着原型链一直往上查找，直到找到为止，如果到了原型链顶端，还没找到，则返回（ undefined ）

16、( )是function对象的一个属性，它定义了构造函数制造出的对象的公共祖先。

* A: Function.prototype
* B: Function.\_\_proto\_\_
* C: objecg.prototype
* D: 以上都不对

17、判断一个对象是否包含了某个属性，应该使用的方法是( )?

* A: typeof
* B: instaceof
* C: hasOwnProperty
* D: 以上都不对

18、var obj = [];检测obj是数组还是对象字面量，可以使用Object.prototype.( )?

* A: typeof obj
* B: toString.call(obj)
* C: obj instanceof Object
* D: 以上都不对

19、关于构造函数与原型，以下说法错误的是？

* A: 每个构造函数都有prototype属性，用于指向它的原型对象
* B: 原型对象的construct指向它的构造函数
* C: 实例对象的\_\_proto\_\_属性用于指向构造函数的原型对象
* D: 不是所有的对象都有\_\_proto\_\_属性，只有实例化构造函数创建的对象才有

20、（ ）可以用于检测属性是否属于实例自身,返回的结果是布尔类型的数据，true表示是？

* A: hasOwnProperty
* B: instanceof
* C: prototype
* D: \_\_proto\_\_

21、借用构造函数是为了解决原型中包含引用类型值所带来的问题，在子类构造函数的内部调用父类构造函数，可以借助( )来改变对象的执行上下文。

A: \_\_proto\_\_

B: apply

C: instanceof

D: constructor

22、获取对象属性的方法可以是?

* A: 对象.属性
* B: 对象(属性)
* C: 对象["属性"]
* D: 对象{"属性"}

23、function F() {

this.faName = 'father';

}

F.prototype.getfaName = function() {

console.log(this.faName);

};

function C() {

this.chName = 'child';

}

C.prototype = new F();

C.prototype.constructor = child;

C.prototype.getchName = function() {

console.log(this.chName);

}; 这种只使用原型链实现的继承，它的缺点有哪些（ ）？

* A: 无法实现函数复用，每个子类实例都持有一个新的fun函数，太多了就会影响性能
* B: 如果父类包含引用类型的属性，那么子类所有实例都会共享该属性
* C: 在创建父类实例时，不能向子类的构造函数传递参数
* D: 在创建子类实例时，不能向父类的构造函数传递参数

24、以下哪些选项是用于确定原型和实例之间的关系？

* A: constructor
* B: typeof
* C: instanceof
* D: isPrototypeOf()

25、使用组合继承方式，可以把方法定义在原型上，属性定义在实例上。 对

26、构造函数中的（ ）对象指向它的实例？

* A: object
* B: this
* C: string
* D: prototype

26、function Foo(){

this.name = "marry";

}

var foo = new Foo();以上代码中this是对( )的引用。

* A: window
* B: Foo
* C: foo
* D: function

27、console.log(typeof([]))打印结果是（ ）？

* A: array
* B: Array
* C: object
* D: Object

28、var name = 'john';

function showName() {

console.log(this.name);

}

var obj = {

name: "jacky",

showName: showName

};

obj.showName();打印结果是john。 错

29、函数的（ prototype  ）属性指向它的原型对象？

30、function C1(name){

if(name) this.name = name;

}

function C2(name){

this.name = name;

}

function C3(name){

this.name = name || 'John';

}

C1.prototype.name = 'Tom';

C2.prototype.name = 'Tom';

C3.prototype.name = 'Tom';

alert((new C1().name)+(new C2().name)+new C3().name)弹出结果是（ Tom,undefined,John  ）

31、function Person(name,age){

this.age = age;

}

Person.prototype = {

getAge:function(){

alert(this.age);

}

};

var p = new Person('bob','21');

console.log(p.constructor==\_\_\_\_\_);

横线处为（ Object ）时打印true？

32、通过（）语句可以给日期对象的原型扩展一个format方法?

* A: Date.prototype.format = function () {}
* B: date.prototype.format = function () {}
* C: Date.constructor.format = function () {}
* D: 以上都不对

33、var temp = {

'x1': 1,

'x2': {'x3': 3}

};

var copyTemp = temp.x2;

temp.x2 = {'x3': 4};

console.log(temp.x2, copyTemp);以上脚本的运行结果是（ ）？

* A: {'x3': 4},{'x3': 4}
* B: {'x3': 4},{'x3': 3}
* C: {'x3': 3},{'x3': 3}
* D: {'x3': 3},{'x3': 4}

34、以下哪些方法可以用于检测一个对象是数组还是对象？

* A: instanceof
* B: Object.prototype.toString.call()
* C: hasOwnProperty
* D: defineProperty

35、var Name = "花花";

var dog = {

Name:"旺财",

prop:{

Name:"小卡",

show:function(){

return this.Name;

}

}

}

console.log(dog.prop.show.call(this));打印结果是 花花

36、Object.\_\_\_\_\_\_\_\_\_\_\_\_.call(obj)横线处的脚本是（ ）时可以检测出obj的类型？

* A: prototype
* B: \_\_proto\_\_
* C: prototype.toString()
* D: prototype.toString

37、{a:1,b:2}=={a:1,b:2}的返回结果是true 错

38、var Abc = function(){

this.num = 1

}

Abc.prototype['exam'] = function(){

this.num++;

}

var abc = new Abc();

console.log(abc.exam())结果是？undefined

39、function foo(){

this.add = function(x,y){

return x+y;

}

}

foo.prototype.add = function(x,y){

return x+y+10;

}

Object.prototype.substract = function(x,y){

return x-y;

}

var f = new foo();

alert(f.add(1,2));

alert(f.substract(1,2));弹出结果是（ 3，-1 ）？

40、构造函数中的this指向（ ）？

* A: 原型对象
* B: 构造函数
* C: window
* D: 实例对象

41、var arr = [1,2];

通过instanceof运算符如何判断arr是否为一个数组，它的写法是( )?

* A: arr instanceof Array
* B: Array instanceof arr
* C: arr.instanceof(Array)
* D: arr.instanceof = Array

42、var obj={a:1,b:function(){}};Object.prototype.toString.call(obj)的返回结果是（ ）？

* A: [object Object]
* B: [object Array]
* C: [object Number]
* D: [object String]

43、（）可以用于确定原型和实例的关系？

* A: indexOf
* B: instanceof
* C: constructor
* D: 以上都不对

44、下列程序的输出是：

var name = 'a';

var person = {

name: 'b',

sayName: function () {

console.log(this.name);

}

};

person.sayName();

person.sayName.call(null);

* A: a b
* B: a a
* C: b a
* D: b b

45、function Animate(){}

var cat = new Animate();请选择以下结果为true的表达式?

* A: cat instanceof Animate
* B: cat instanceof Object
* C: cat.constructor==Animate
* D: cat.constructor==Object

46、var menus = document.getElementsByTagName('span');以下哪些选项可以将menus转成真数组？

* A: [].slice.call(menus)
* B: Array.prototype.slice.call(menus)
* C: Array.from(menus)
* D: Array.isArray(menus)

47、类数组转数组的方法有？

* A: Array.from(类数组）
* B: [...类数组]
* C: [].slice.call(类数组)
* D: Array.prototype.slice.call(类数组)

48、以下关于原型继承原理的说法，正确的有？

* A: 每个函数都有一个原型对象的属性（F.prototype）,每个函数都可以是一个构造函数
* B: 通过new关键字可以创建任意函数的一个实例。
* C: JS中是没有类的，所有继承属于基于原型的继承。
* D: 在调用某个对象的属性或方法时，js引擎会优先在该对象自身上查找该属性或方法，如果没有便去该对象的构造函数中去查找，如此逐级递归查找，直到原型链的最顶层。

49、用组合模式创建面向对象，构造函数上一般存放非公用的属性和方法 对

50、如下程序的输出是\_\_\_\_another\_\_\_\_\_。

var name = 'global'

var obj = {

name: 'obj',

sayName: function () {

console.log(this.name);

}

};

var func = obj.sayName;

func.call({name: 'another'});

51、function SuperType(){}

var ins=new SuperType();

console.log(ins instanceof SuperType)的打印结果是（ ）

A: TRUE

B: FALSE

C: Animate

D: Object

52、function CreateObject(){}

var obj = new CreateObject();

console.log(obj.\_\_proto\_\_ === CreateObject.prototype)关于以上代码的描错误的有（ ）？

* A: obj是实例对象
* B: CreateObject是obj的构造函数
* C: obj的\_\_proto\_\_属性指向构造函数CreateObject。
* D: obj的\_\_proto\_\_属性指向构造函数CreateObject的原型，所以打印结果为true。

53、( )是function对象的一个属性，它定义了构造函数制造出的对象的公共祖先。

* A: Function.prototype
* B: Function.\_\_proto\_\_
* C: objecg.prototype
* D: 以上都不对

54、function S() {

this.property = true;

}

S.prototype.get = function () {

return this.property;

};

function U() {

this.p = false;

}

U.prototype = new SuperType();

U.prototype = {

set: function () {

return this.p;

}

};

var ins = new S();

alert(ins.get());以上代码的运行结果是（ ）？

* A: TRUE
* B: FALSE
* C: undefined
* D: 程序报错instance.getSuperValue is not a function

55、data只有两种可能，要么数组，要么对象字面量，以下检测data到底是不是数组，方法正确的有（ ）？

* A: data instanceof Object == true;
* B: typeof(data) == Object;
* C: data instanceof Array == true;
* D: Object.prototype.toString.call(data) == "[object Array]";

56、以下哪些选项是用于确定原型和实例之间的关系？

* A: constructor
* B: typeof
* C: instanceof
* D: isPrototypeOf()

57、创建自定义对象最简单的方式是通过new Object()或{}来实现 对

58、A intanceof B，它的工作原理是，检测对象B的prototype指向的对象是否出现在Ad对象的[[Prototype]]链上，换句话说，A对象的原型链继承线路上，有没有B的存在。 对

59、组合继承解决了原型链继承中引用类型属性被实例共享的问题 对

60、工厂模式解决了重复实例化的问题 ，但还有一个问题,那就是识别问题，因为根本无法 搞清楚他们到底是哪个对象的实例。 对

61、instanceof可以用于检测对象是否是某个构造函数的实例。对

62、以下哪条语句会产生运行错误？

* A: var obj = ();//语法错误
* B: var obj = [];//创建数组
* C: var obj = {};//创建对象
* D: var obj = //;

63、所有的JavaScript对象，都存在一个( )属性,该属性指向它的构造函数的原型?

* A: \_\_proto\_\_
* B: prototype
* C: constructor
* D: instanceof

64、用组合模式创建面向对象，构造函数上一般存放非公用的属性和方法 对

65、构造函数中的（ ）对象指向它的实例？

* A: object
* B: this
* C: string
* D: prototype

66、var Abc = function(){

this.num = 1

}

Abc.prototype.exam = function(){

return this.num++;

}

var abc = new Abc();

console.log(abc.exam())结果是？

* A: undefined
* B: 1
* C: 2
* D: null

67、创建一个对象字面量，以下哪些创建方法是正确的（ ）

* A: var obj = {};
* B: var obj = [];
* C: var obj = new Object()
* D: var obj = new Array()

68、下列代码中访问1属性的方式是错误的。

var person = {

1: 98066

'name' : "john green",

'first-name' : “john"

};

person.1 对

69、var timer = null;console.log(typeof(timer))的打印结果是（ ）？

* A: undefined
* B: null
* C: object
* D: ""

70、（网站）创建图片对象的方法是？

* A: new Img()
* B: new Image()
* C: create Img()
* D: create Image()

71、var fullname = ""张三"";

var obj = {

fullname:""李四"",

prop:{

fullname:""王五"",

getFullname:function(){

return this.fullname;

}

}

}

var test = obj.prop.getFullname;

console.log(test.call(obj.prop));打印结果是( )知识点：Math，来源：练习手册5单元"

A、张三;

B、李四

C、王五

D、undefined

72、以下哪个运算符用来检测引用型数据(如对象，数组等）的类型？（ ）知识点：数组，来源：自出

A、typeof

B、prototype

C、instanceof

D、 type

73、new运算符可以用来调用构造函数，最终返回一个对象字面量。错 返回的是实例对象

74、var d = new Date();

console.log(typeof d, d instanceof Date);程序输出结果为object,true 对

75、利用( ),( )方法可以改变this指向的方法也可以实现继承 call(),aplly()

76、通过（）语句来遍历一个对象？ for in

77、Function.prototype.delay = function(){}对于这样的写法解释不合理的是？

A、相当于在Function的原型链添加了一个方法

B、var say = function(){};say.delay()可以这样使用

C、定义过后，所有的函数都可以使用delay这个方法

D、任何一个对象上都可以直接调用delay()来执行此方法

78、var obj={a:1,b:function(){}};Object.prototype.toString.call(obj)的返回结果是（ ）？

A、[object Object] B、[object Array] C、[object Number] D、[object String]

79、（）可以用于确定原型和实例的关系？

A、indexOf B、instanceof C、constructor D、以上都不对