原型链题目

1 、prototype和proto的概念（简答题）

prototype是函数的一个属性（每个函数都有一个prototype属性），这个属性是一个指针，指向一个对象。它是显示修改对象的原型的属性。

proto是一个对象拥有的内置属性，是JS内部使用寻找原型链的属性。

2、prototype和proto的区别？

prototype是函数的内置属性，**proto**是对象的内置属性

3、js中的原型链以及最顶端是什么？（填空题）

Object.prototype

4、以下代码会输出什么？（ ）（ ）（ ）（ ）（ ）（填空题）

```
function Foo() {
    getName = function() {
        alert(1)
    };
    return this;
}
Foo.getName = function() {
    alert(2)
};
Foo.prototype.getName = function() {
    alert(3)
};
var getName = function() {
    alert(4)
};

function getName() {
    alert(5)
};
Foo.getName(); //2
getName(); //4
Foo().getName() // 1
getName(); //1
new Foo.getName(); //2
```

答案：2　　4　　1　　1　　2

注释： getname() 在调用的时候 ，前面声明了两个getname函数 一个是function通过函数声明 另一个是通过 var 也就是函数表达式声明函数

根据预编译的过程 ，函数表达式和函数声明都会发生变量提升

var getname=function(){alert(4)}

function getname(){alert(5)}

过程：

var getname;

funciton getname(){alert(5)}

getname=function(){alert(4)}

5、以下代码会输出什么？（ ）（填空题）

```
function A() {
    this.do = function() {
        return 'foo'
    }
}
A.prototype = function() {
    this.do = function() {
        return 'bar'
    }
}
var x = new A().do()
console.log(x)
```

答案：foo

6、以下代码会输出什么？（ ）（填空题）

```
function C1(name) {
    if (name) {
        this.name = name;
    }
}

function C2(name) {
    this.name = name;
}

function C3(name) {
    this.name = name || 'join';
}
C1.prototype.name = 'Tom';
C2.prototype.name = 'Tom';
C3.prototype.name = 'Tom';
```

```
console.log((new C1().name)+'/' + (new C2().name)+'/' + (new C3().name));
```

答案：Tom/undefined/join

new c1().name 已知 c1中没有name属性（实例化的时候没传参数） 所以在访问name属性的时候 会访问c1原型中的name属性 即后面赋值原型的语句 C1.prototype.name = 'Tom'; c2在访问name属性的时候 会先从c2内部找 已知 c2内部已经有 this.name=name了 虽然在实例化的时候没有传入参数作为name的值 但是this.name赋值的步骤没有经过任何条件判断 所以会赋值为 undefined new c3().name c3.name会先从c3的实例对象找 已知 c3的实例对象中this.name也是没有任何条件判断必然会执行 他的值是如果name有值就是传入的name的值 如果没有传入参数name的值 就是join 实例化的时候没有传参数 所以是join

7、以下代码会输出什么？（ ）（ ）（ ）（填空题）

```
 function A() {};
 function B(a) {
   this.a = a;
 }
function C(a) {
   if (a) {
    this.a = a;
   }
 }
 A.prototype.a = 1;
 B.prototype.a = 1;
 C.prototype.a = 1;
 console.log(new A().a);
 console.log(new B().a);
 console.log(new C(2).a);
```

答案：1 undefined 2

分析：首先看当前构造函数是否定义这个属性，若有直接输出，否则在其原型链上依次查找。
A：原型链查找到 a=1；B：函数本身有 a属性；C：函数先判断参数；若参数的布尔值为真则输出 a的值,否则输出原型链上a的值

8、以下代码会输出什么？（ ）（ ）（ ）（ ）（填空题）

```
var F = function() {};
        Object.prototype.a = function() {
            console.log("a()");
        };
        Function.prototype.b = function() {
            console.log("b()");
        };
        var f = new F();
        F.a();
        F.b();
        f.a();
        f.b();
```

答案：a()　　b()　　a()　　报错

分析：
函数内部和函数原型上没有 a，沿着其原型链查找属性。
f.b()，f 的构造函数是 F 而非 Function ，从上述原型链图观察，f 的原型链查找不到 Function的原型。

F.a() 因为F是 函数 继承自 Function Function又继承自 Object 所以a属性不在F中时会按顺序查找到 object.prototype.a的结果为 a()

F.b同理

f.a() 因为f是通过F函数构造出的对象 所以 他属于object 能访问object.prototype 但是不属于 function 所以不能访问 function.prototype

所以 f.b()报错

9、以下代码会输出什么？（ ）（ ）（ ）（ ）（填空题）

```
var A=function(){}
A.prototype.n=1
var b=new A()
A.prototype={
n:2,
m:3
}
var c=new A()
console.log(b.n,b.m,c.n,c.m)
```

答案：1　　undefined　　2　　3

10、f有方法a吗？有方法b吗？（ ）（ ）（填空题）

```
var F = function () {}
Object.prototype.a = function () {}
Function.prototype.b = function () {}

var f = new F()
```

答案：有a　　没有b

11、以下代码会输出什么？（ ）（ ）（ ）（ ）（填空题）

```
function Person(){}
    Person.prototype.n=1
    var p1=new Person()
    Person.prototype={
      n:2,
      m:3
    }
    var p2=new Person()
    console.log(p1.n,p1.m,p2.n,p2.m)
```

答案：1　　undefined　　2　　3