**JavaScript进阶和面向对象**

**验 收 考 试**

**1. 试写出程序运行结果**

var a = 1; //2

function fn (){

this.a++;

a += 5;

var a = 8;

}

fn();

alert(a); //2

**2. 试写出程序运行结果**

var m = 1;

var obj = {

fn1: function() {

return this.fn2();

},

fn2: fn2,

m: 2

};

function fn2() {

return this.m;

}

var result = obj.fn1();

console.log(result); // 2

**3. 试写出程序运行结果**

var length = 5;

var arr = [fn1,fn2];

function fn1(){

return this.length;

}

function fn2(){

return this[0];

}

var a = arr[0]();

var b = arr[1]()();

console.log(a); //2

console.log(b); //5

**4. 试写出程序运行结果**

function fun1(f,a,b,c){

arguments[0](1,2);

}

function fun2(p,q,r,s,t){

alert(this.length); // 6

alert(this.callee.length); //4

alert(arguments.length); //2

alert(arguments.callee.length); //5

}

fun1(fun2,1,2,3,4,5);

**5. 试写出程序运行结果**

var a = true;

function fn(){

if(!a){

var a = 10;

}

console.log(a);

} //u

**6. 试写出程序运行结果**

var number = 2; // 4 16

var obj = {

number : 3, //12 24

fn1 : (function(){

this.number \*= 2;

number = number \* 3;

var number = 2; // 2 10 50

return function(){

this.number \*= 4;

number \*= 5;

alert(number); //10 50

}

})(),

fn2 : function(){

this.number \*= 2;

}

};

var fn1 = obj.fn1;

alert(number); //4

fn1(); //10

obj.fn1(); //50

obj.fn2();

alert(window.number); //16

alert(obj.number); //24

**7. 试写出程序运行结果**

function A(){}

function B(){

return new A();

}

A.prototype = B();

B.prototype = new B();

var a = new A();

var b = new B();

console.log(a.\_\_proto\_\_ == b.\_\_proto\_\_); //t

console.log(a instanceof A); //t

console.log(a instanceof B); //f

console.log(b instanceof A); //t

console.log(b instanceof B); //f

**8. 试写出程序运行结果**

console.log(Object instanceof Object); //t

console.log(Function instanceof Function); //t

console.log(Number instanceof Number); //f

console.log(String instanceof String); //f

console.log(Function instanceof Object); //t

**9. 试写出程序运行结果**

var length = 5;

function getLength(){

return this.length;

}

function foo(){

this.length = 1;

return (function(){

var length = 2;

return {

length : function(a,b,c){

return this.arr.length

},

arr : [1,2,3,4],

info : function(){

return getLength.call(this.length);

}

}

})();

}

var result = foo().info();

console.log(result); //3

**10.** 什么是事件委托？我们常用事件委托解决什么问题？

**11.** 简述JSONP跨域的原理。

**12.** 请使用观察者模式完成小程序编写：有老师类（Teacher）、学生类（Student），老师有发布作业方法（sendhomework），学生有姓名属性（name）。调用老师的发布作业方法后，将会在控制台输出：“\*\*学生收到了老师布置的作业”，“\*\*学生收到了老师布置的作业”。（\*\*为学生姓名，有多少学生，就输出多少条）。

**13.** 试定义两个类：老师类（Teacher）、学生类（Student)，让学生类继承老师类。

**14.** 如何将类数组对象转换为数组？

**15.** 试写一个函数deepclone，接收一个任意类型的参数，实现深度克隆。

**16.** 试写一个函数currying，接收一个函数fn做参数，可以将fn科里化。