**1. 试写出程序运行结果（2）**

var a = 1;

function fn (){

this.a++;

a += 5;

var a = 8;

}

fn();

alert(a);

**2. 试写出程序运行结果（2）**

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);

**3. 试写出程序运行结果（4）**

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);

console.log(b);

**4. 试写出程序运行结果（8）**

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

arguments[0](1,2);

}

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

alert(this.length);

alert(this.callee.length);

alert(arguments.length);

alert(arguments.callee.length);

}

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

**5. 试写出程序运行结果（2）**

var a = true;

function fn(){

if(!a){

var a = 10;

}

console.log(a);

}

fn();

**6. 试写出程序运行结果(12)**

var number = 2;

var obj = {

number : 3,

fn1 : (function(){

this.number \*= 2;

number = number \* 3;

var number = 2;

return function(){

this.number \*= 4;

number \*= 5;

alert(number);

}

})(),

fn2 : function(){

this.number \*= 2;

}

};

var fn1 = obj.fn1;

alert(number);

fn1();

obj.fn1();

obj.fn2();

alert(window.number);

alert(obj.number);

**7. 试写出程序运行结果(10)**

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\_\_);

console.log(a instanceof A);

console.log(a instanceof B);

console.log(b instanceof A);

console.log(b instanceof B);

**8. 试写出程序运行结果(10)**

console.log(Object instanceof Object);

console.log(Function instanceof Function);

console.log(Number instanceof Number);

console.log(String instanceof String);

console.log(Function instanceof Object);

**9. 试写出程序运行结果(2)**

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);

**10.** 什么是事件委托？我们常用事件委托解决什么问题？(10)

**11.** 简述JSONP跨域的原理。(9)

**12.** 试定义两个类：老师类（Teacher）、学生类（Student)，让学生类继承老师类。(9)

**13.** 定义两个类，飞机场类和飞机类。飞机在new的时候要传递所在机场的实例。飞机的实例有flyto()出发去某地的方法，接受一个机场的实例，此时弹出\*\*\*飞机从\*\*\*机场飞往\*\*\*机场。飞机场类有info()方法，能够显示目前停靠在本机场的飞机。(20)