软件学院 赵馨逸 2012013277

web前端技术实训

2014年7月3日

JavaScript Quiz
WEB前端第二次作业

Section1：基础练习题

1、Argument 作用域

【题目】

```
var func = {
    getNum: function() { return this.num; },
    num: 1
};

(function(){
    return typeof arguments[0]();
})(func.getNum);
```

问：如上的自执行函数的返回值是什么？
请解释为什么会是这样的返回值？

【chrome控制台运行结果】

```
> var func = {
      getNum:function(){return this.num;},
      num:1
  };
  (function(){
      return typeof arguments[0]();
  })(func.getNum);
  "undefined"
>
```

【结果分析】

这个例子建立了一个名为func的对象，设置其属性num为1，并创建了一个名为getNum的方法。

在 JavaScript 中,上下文对象就是 this 指针,即被调用函数所处的环境。

执行时， JavaScript为函数的呼叫建立了一个运行上下文，通过”．“运算符把this指向被引用的对象，在此是getNum这个对象。之后这 个方法就可以通过this在window中找到它自身的属性，由于num不是function getNum的变量，找不到，返回undefined。

2、动态属性 构造函数返回值

【题目】

```
var x = 0;
function foo() {
    x++;
    this.x = x;
    return foo;
}
var bar = new new foo;
console.log(bar.x);
```

问：如上console.log出的是什么？
为什么会这样？

【chrome控制台运行结果】

```
> var x = 0;
  function foo() {
      x++;
      this.x = x;
      return foo;
  }
  var bar = new new foo;
  console.log(bar.x);
  undefined
< undefined
```

【结果分析】foo()函数含有return语句，所以new的运算返回结果为foo(),所以bar始终为foo(),为一个函数，bar.x是未定义，因此为undefined。

3、typeof 预编译

【题目】

```
function bar() {
    return foo;
    foo = 10;
    function foo() {}
    var foo = '11';
}
alert(typeof bar());
```

问：alert的结果是什么？
为什么会这样？

【chrome控制台运行结果】

【结果分析】

function bar（）是函数声明式定义，预编译的时候会将其视为声明，bar为function类的一个实例对象，而函数名bar仅仅是该实例的一个引用地址，typeof会返回该实例的类型，即为function。

4、this指针

【题目】

```
var x = 3;
var foo = {
    x: 2,
    baz: {
        x: 1,
        bar: function() {
            return this.x;
        }
    }
}

var go = foo.baz.bar;

alert(go());
alert(foo.baz.bar());
```

问：左边代码分别alert出的结果是什么?
为什么会这样?

【chrome控制台运行结果】

【结果分析】

本题中定义了多个x变量，第一个x=3被定义在global中，第二个x=2定义在foo中，第三个x=1被定义在baz中。

在 JavaScript 中,上下文对象就是 this 指针,即被调用函数所处的环境。上下文对象的作用是在一个函数内部引用调用它的对象本身。

在 JavaScript 中,本质上,函数类型的变量是指向这个函数实体的一个引用,在引用之间赋值不会对对象产生复制行为。我们可以通过函数的任何一个引用调用这个函数,不同之处仅仅在于上下文。

使用不同的引用来调用同一个函数时,this 指针永远是这个引用所属的对象。

第一个alert(go());中函数go()被调用的环境为global环境，所以返回3。

第二个alert(foo.baz.bar());中函数bar()被调用的环境为baz环境，所以返回1。

参考：http://www.cnblogs.com/bennman/archive/2013/09/08/3309024.html

参考练习：

```
var someuser = {
    name: 'byvoid',
    func: function() {
        console.log(this.name);
    }
};
var foo = {
    name: 'foobar'
};

someuser.func(); // 输出 byvoid

foo.func = someuser.func;
foo.func(); // 输出 foobar

name = 'global';
func = someuser.func;
func(); // 输出 global
```

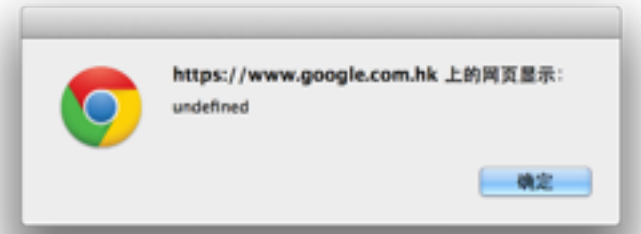
5、语句

【题目】

```
function aaa() {
    return
    {
        test: 1
    };
}
alert(typeof aaa());
```

问：alert出的结果是什么？
为什么会这样？

【chrome控制台运行结果】



【结果分析】

在javascript中一个文本行默认是一条语句，无论行末是否带有分号。所以aaa()执行时返回的并不是{test: 1};，因此typeof的结果是undefined。

Section2：进阶练习题

1、中国队是冠军

（1）代码：详见forecast.js文件

（2）代码说明：forecast(na,re)//na为每一支国家队的实力，re为用户需要获取的队伍的胜率。

（3）算法说明：

本题主要运用概率论-条件概率知识。

根据全概率公式，每一支队伍进入下一轮的概率 = 自身进入本轮的概率 * (Σ(每一支本轮可能遇到的队伍进入本轮的概率*两支队伍的胜率))。

每两支队伍相遇的胜率很容易计算，在代码中为calculate_win()函数。

难点在于如何寻找可能遇到的队伍，并且计算它们进入本轮的概率。

每一支队伍可能遇到的队伍为与其在其他区域的队伍，例如，1/4决赛时，可以认为16支队伍被分为4个大区，第1大区将遇到第2大区的四支队伍。所以可以根据每一轮的区域用取余数方法来计算每一支队伍可能遇到的队伍（在计算前，已经按照A1,B2,C1,D2...顺序重新排序，简化取余计算的判断）。

同时由于每一轮计算需要其他队伍进入本轮的胜率，我采用了计算每一轮所有队伍胜率的方式，并将它们储存在round1,round2,round3,round4中。写为四个函数，分别计算。（代码有注释）。

最后，根据用户输入的result，获取round4中相应的队伍的胜率。

（4）其他说明：

测试方法：

方法1：令16支队伍实力相当，输出round4中每一支队伍胜率，均为0.0625.正确。

方法2：计算round4中所有队伍胜率和，为0.9999999998.接近1，正确。

2、找呀找呀找同学

（1）代码：详见search.js文件

（2）代码说明：forecast(information,num)//information为同学们的基本信息，num为用户需要查询匹配的信息。

（3）算法说明：

本题考虑JS中没有函数的重载，需要通过argument来判断传入参数的类型。通过typeof判断，如果第二个参数为数字，查找所有年龄相同的同学。如果第二个参数为字符串，查找第一个同名同学。如果为object，则对object中每一项进行匹配。

（4）其他说明：

测试方法：

每一个if语句均有找得到和找不到的情况进行测试，对于需要反馈所有结果的，均会测试有2个以上匹配结果的测试。由于test为false时consile.info输出有误，所以测试函数需要判断test是否为真。

Section3：untrusted游戏

1\https://gist.github.com/anonymous/db3ecee2239d047057bc扩大边缘范围
2\https://gist.github.com/anonymous/ca138c0c5fa5f3577597更改出口位置
3\https://gist.github.com/anonymous/247dff40bbfe522e597c扩大边缘，好吧，才知道第一个直接删代码就行，根本不用扩大边缘，我傻啦！！
4\https://gist.github.com/anonymous/064c450ab2c0ded9cc99加了一个出口，那。。。第二关我是不是又想多了
5\https://gist.github.com/anonymous/f3e6454983bbece035af把mine的地方变成黑色！眼睛都被晃瞎了尼玛
6\https://gist.github.com/anonymous/ac62a6f7f1b163678f98只能说这个me太不机智了，用block把它憋住!
7\https://gist.github.com/anonymous/e7f7b8386346f8f9a11aatlocation这个函数谁知道它能用!
8\https://gist.github.com/anonymous/345392f41a082969ad2e没啥说的。。。能改的只有那个地方><
9\https://gist.github.com/anonymous/7bb07f0f26f04dca6243

10\https://gist.github.com/anonymous/bf711db0ab127b5a0274每一种动一个就行了，本来还以为他们可以相互攻击呢><不好玩
11\https://gist.github.com/anonymous/88be4637bf526a2b2e74总有种推箱子的赶脚
12\https://gist.github.com/anonymous/34c2ad636628cdd0dc51
13\https://gist.github.com/anonymous/b418af0ee61bb8caab6e我是一个不智能的人><
14\https://gist.github.com/anonymous/e1959d723a270c073ac7忘记路线了><
15\https://gist.github.com/anonymous/9c8a8a13b7609ee3c675呵呵呵
16\https://gist.github.com/anonymous/0533d3fa0c155c67d674把getrandom的那个函数全返回零，然后线就画不出来啦~
17\https://gist.github.com/anonymous/4b17304e9d12d187026b我不会，我以为可以让它标记传送门，但是出来以后发现不是所有的房间都能标记出来的，然后在没有标记的房间我就得乱试，然后就过了><
18\https://gist.github.com/anonymous/2a3f4bd45ae0e72c318a仿照他的样子写一个在左半部分向上跳的就好了
19\https://gist.github.com/anonymous/378cbb567a51dfc99d73他们说随便乱按就好了。。。。。
20\https://gist.github.com/anonymous/716d8d6f3ebdb8308ea3要拿到phone，先想的是能挡子弹，但又不能用#，然后就随便找了个树，放了一排。复制了一下bullet的代码，写了一个向上发的子弹。最开始就是直接放了一排向上的子弹，结果没干掉全部boss就没了，所以就把他们放在一个函数里面，执行很多次，触发用的是callback。结果发现boss吐出来的A在上面，被tree挡住了，所以就在左侧留了一个小口，方便进出。
21\https://gist.github.com/anonymous/af832e9359c238efa8e3menu里面object的exit改掉就好了。（自己没想到）
22\balabala