第二次作业问题汇总

1 基础练习 1

推荐解答：undefined。Javascript 中，函数作用域中的 this 指针指向调用该函数对象的作用域。在本题中，函数调用为 arguments[0]()，故其 this 指针指向 arguments（arguments 是一个特殊的对象）。又由于 arguments 对象无 num 成员，故输出为 undefined。

解题误区：

- this 指向 window。通过添加 window.num = 1 测试。
- this 指向 arguments[0]。arguments[0]是函数本身，调用函数的是[function]。

2 基础练习 2

推荐解答：undefined。foo()函数的返回值是函数这一个引用类型的数据，因此 foo 作为构造函数用 new 运算符执行构造时，运行结果将被它的返回值，也就是 foo 函数取代。由于构造的结果 bar 不是一个实例，因此没有 x 属性。

概念误区：

- 将 x 属性叙述为 x 变量。类成员是属性。
- 将实例与类混淆。new 出来的是实例，函数本身可以看做一个类。但是当函数返回一个函数时，构造出来的不是一个实例。
- 误认为去掉 return foo 后，就能返回 2。事实上，此时返回的是一个 object，不能被 new 第二次。

3 基础练习 3

推荐解答：function。在预编译时，将所有 var 变量创建为 undefined（不执行相应的赋值语句）。然后预编译定义式的函数。因此正确的顺序是 var foo; function foo(){}; return; foo=10; foo = ‘11’。

解题误区：

- 认为 var foo 不声明。

4 基础练习 4

推荐解答：3，1。this.x 的值取决于调用该函数的作用域。go 即 foo.baz.bar 这个函数。go 在全局作用域调用，此时 this.x = window.x = 3。foo.baz.bar()这一调用，是通过对象进行的调用。因此查看 foo.baz 的作用域，发现 x 属性为 1。

解题误区：

- 这题大家做得比较好，普遍没有什么错。

5 基础练习 5

推荐解答：undefined 对于一个语句，如果在行尾没有分号，JavaScript 会为该处添加分号。因此此处 return 空。注：该情况对于一个语句或一个表达式成立。如字符串不支持跨行。但是 object 或数学表达式支持跨行。

解题误区

- 忽略了 JS 编译特性。
- 直接使用 chrome 控制台尝试。默认了代码等效。

6 进阶练习 1

推荐解题思路：

思路 1

- 求解一张 Hash Map，记录每个球队在 16 强-决赛可能遇到的各个对手。
- 定义函数：计算队伍 t 进入轮次 r 的概率。利用全概率公式（递归式，递推式）。其中递归可读性更强（推荐）。

思路 2

- 不定义 Hash Map，而是定义一个 check 函数，检查队伍 x 在轮次 r 是否会遇到对手 y。这样复杂度会提高，但是更为直观。
- 同样利用递归或递推的方式求解。

存在的一些问题：

- 暴力求解所有的概率：绝对不推荐！！！对于暴力求解的同学，我只想说，我要令轮次 n>20。
- 枚举每一轮：虽然轮次内不是暴力求解。但是依然是暴力求解，轮次依然有必要被设置为 n>20。

- 循环过于抽象：过多的循环嵌套，以及循环变量含义的不明。完全无法突出核心的递推或者递归式。
- 数组和对象使用不当。这样的现象在大家的代码中太普遍了：需要使用映射关系的情况下，定义了 switch(name){case: 'A1' return 0}这样的映射函数的现象。代码冗长而不美观。
- 一个函数完成所有的事。如果逻辑清晰尚可。但是如套了 4 个并行的循环（对应一个轮次），每个轮次又套着 n 层类似的代码的写法就不太合适了。
- 代码风格亟需改善：
  - new Array()和 new Object()可使用[]和{}代替。尽量使用后两种方式进行初始化。切忌出现 new Array()之后跟着一排 arr[i] = xx 或 new Object()后跟着一排 obj.x = xx 的语句。还有 Array，Object 混合暴力初始化的，我除了跪下别无办法。
  - {}对齐问题等。Try http://jsbeautifier.org/
  - 可以合并的两个 if 语句，写了两层嵌套！
  - C 风格太严重。
  - ……
- 测试过于不友好
  - 测试数据写死。
  - 乱用自执行函数，影响测试。
  - 一堆 debugger 在代码中报复社会。
  - 控制台编译不通过，需要助教帮忙检查修改语法错误的（这种情况下回出现一律不给分）。

7 进阶练习 2

推荐思路：

- 函数重载：使用 switch 分支语句，对输入查找项的类型进行判定
- 对于 undefined 属性的判定：对于一个对象，遍历其属性名。在这样的情况下 obj[a]肯定是存在的，但是查找的对象 s 中 a 属性可能不存在，为 undefined。此时依然可以判定二者是否相等。for var a in obj 可以避免暴力枚举。

```
for(var a in obj)
    if(obj[a] != s[a] )
```

- 对象查找到要求所有信息匹配。采用不等除去所有判定分支。

存在的一些问题：

- 匹配 n 个属性，就有 2^n 个分支。if 分支多了会很恐怖。
- 条件语句过于不统一。互斥条件，一会 if、else if，一会又冒出另一个 if，造成逻辑混乱。代码可读性很差。
- 进阶练习 1 中存在的一些共性问题。如代码风格等。

8 BONUS 1

- 可以利用一些小 trick 来让代码更优雅简单，比如 a=new Array()直接用 a=[]就够了。
- 这题其实可以利用上一题的结果。
- 或者可以直接利用 JS 的对象：扫描一遍数组 1，把每个数组元素的 name 作为 JS 对象 obj 的一个域，设置 obj[arr1[i].name]为 1；扫描数组 2，对于每个数组元素，直接判断 obj[arr2[i].name]，若不存在就将该元素加入结果数组即可。

9 BONUS 2

- 显然和数组有很大关系。那么，JS 数组有什么特别的呢？那就是一整套原生方法。
- http://www.w3school.com.cn/jsref/jsref_obj_array.asp
- 举个例子，比如插入排序，是不是可以借助 slice 来进行数据的整片移动呢？那样我们的代码中就能少一重循环了。

10 附录

- JS 代码风格指南：http://chajn.org/jsguide/javascriptguide.html