说明：github中的链接 <http://github.com/yangyanzhe/Homework2_web>

Javascript练习题

软件22 杨妍喆 2012013323

联系方式: [yangyanzhe@126.com](mailto:yangyanzhe@126.com) 18010045724

基础练习一

var func = {

getNum: function() {return this.num;},

num:1

};

(function(){

return typeof arguments[0]();

})(func.getNum);

考察内容：关于arguments与作用域

问：如上的自执行函数的返回值是什么？请解释。

答：返回值是undefined

在func.getNum作为参数传入function, 作为arguments[0]。 arguments[0]后的括号表明执行该函数，也就是返回this.num。

由于this指的是调用处的域，该域中没有num，因此返回“undefined”。

基础练习二

考察内容：动态属性、构造函数返回值

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？

答：console.log出的是undefined

原因：

由于foo返回它本身，因此bar在构造出来是一个函数，而不是一个对象。因此bar不具有动态属性，访问bar.x时会显示undefined。

基础练习三

function bar(){

return foo;

foo=10;

function foo(){}

var foo='11';

}

alert(typeof bar());

考察内容：typeof、预编译

问：alert的结果是什么？为什么会这样？

答： alert的结果是 function。

原因：

alert()方法用于显示一条指定消息和一个OK按钮的警示框，而typeof的返回值只能是undefined、boolean、string、number、object、function六种。

为什么弹出的框内显示的是“function”，而不是number、string呢? 按照语句，alert(typeof bar())弹出的应该是bar()执行后返回的内容的类型，也就是返回的foo的类型。而返回foo值时，bar()还没有运行到赋值语句，因此反映的是预编译的内容。JS解析器在执行语句前会将函数声明和变量声明进行预编译，所以弹出的是function。

基础练习四

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());

考察内容：this指针

问：分别alert出的是什么？为什么会这样？

答：

alert(go())弹出的是3, alert(foo.baz.bar()) 弹出的是1.

原因：

this指针指向被调用对象的作用域。

go()的作用域是全局，因此this.x是3。

alert(foo.baz.bar())的作用域是foo.baz，此时this.x是1。

基础练习五

function aaa(){return {test: 1};}

alert(typeof aaa());

考察内容：语句

问：alert出的结果是什么？为什么会这样？

答：alert出的结果是object.

原因：

function内的语句相当于function aaa(){var a = {test: 1}; return a;}，因此返回的是一个对象。

进阶练习一

要求：

实现一个方法forecast，用来预测某支球队获胜的概率。参数要求为：第一个参数为一

个对象，包含16支球队的能力值；第二个参数为某个国家的代码。

返回值：第二参数所对应的获胜概率

思路：

1. 将对象转换为数组方便计算，分别计算16→8→4→2→1

2. 利用条件概率及全概率公式，分别计算每个队第一轮至第四轮获胜的概率，计算规则如下(以8进4的A1为例)：

在八进四的时候，A1可能会与C1、D2比赛，所以A1进入四强的概率为

p = p(A1进入八强) \* {p(A1胜C1)\*p(C1进入八强) + p(A1胜D2)\*p(D2进入八强)}

2. 将函数封装为方法，使之在调用时使用特定的this引用的简单方法。

源代码： 见 <src/forecast.js> 或

<https://github.com/yangyanzhe/Homework2_web/blob/master/src/forecast.js>

测试说明：

测试如下图示，测试数据输入格式附在forecast.js末端。

var value = {

A1: 0, A2: 20, B1: 35, B2: 46,

C1: 55, C2: 63, D1: 0, D2: 38,

E1: 49, E2: 0, F1: 91, F2: 83,

G1: 13, G2: 77, H1: 63, H2: 22

};

var order = 'H1';

var a = new forecast(value, order);

a.run();

测试结果与分析：

简单的样例通过分析判断结果的正确性，复杂的样例通过与不同同学比对，进行检验。下面为节选的一部分样例。通过测试样例，且将函数封装为方法，满足题目要求。

|  |  |  |
| --- | --- | --- |
| 参数一(A1, A2, B1, B2, C1, C2, D1 …) | 参数二 | 结果 |
| 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 | A1 | 0.0625 |
| 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 | A1 | 0.25 |
|  | F1 | 0 |
| 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 | A1 | 0.000189712282430006 |
|  | A2 | 0.00221112767596038215 |
|  | G1 | 0.09595312023303208 |
| 100, 20, 35, 46, 55, 63, 75, 38,  49, 16, 91, 83, 13, 77, 63, 22 | A1 | 0.1744601124234268 |
|  | A2 | 0.0040889381006781666 |
|  | D1 | 0.10154811745714781 |
|  | F2 | 0.15288152967390137 |
| 0, 20, 35, 46, 55, 63, 0, 38,  49, 0, 91, 83, 13, 77, 63, 22 | A2 | 0.004776119105676425 |
|  | F1 | 0.2248142683545148 |

进阶练习二

考察内容：函数重载

要求：构建search方法方便进行查找。

信息元用对象来表示{name: ”Amy”, Age: 21, Hometown: ‘Beijing’}。参数要求为：一为由上述元素构成的数组；二为一个可变参数，其值可以是数字、字符串或对象。

返回值要求：

1. 若参数二是一个数字，则返回所有年龄与该数字相同的数组，找不到返回false;

2. 若参数二是一个字符串，则返回和该字符串相同的第一个信息元，找不到返回false;

3. 若参数二是一个对象，对象中包含一个或多个信息，要求找到和这个对象匹配的同学信息组成的数组，如果找不到，返回false

思路：

1. 判断参数二的类型分别处理；

2. 如果是一个对象，则

2.1 按name、age、hometown的顺序依次筛选

2.2 如果参数二没有age、name或hometown类型，则跳过；

2.3 如果参数二有该类型，但不匹配，直接返回false

3. 将函数封装为方法

源代码：见 <src/search.js>

或<https://github.com/yangyanzhe/Homework2_web/blob/master/src/search.js>

测试说明：

测试格式如下图示，测试数据输入格式附在search.js末端。

//创建实例对象

var stu1 = new person("Amy", 21, 'Beijing');

var stu2 = new person("Bob", 22, 'Beijing');

var stu3 = new person("Amy", 23, 'Shanghai');

var myArray = new Array(stu1, stu2, stu3)

var test = {name: "Amy", hometown: 'Shanghai'};

var a = new search(myArray, test);

a.run();

测试结果与分析：

在测试样例下，结果正确。

学生信息（简化）：

“Amy”, 21, ‘Beijing’

“Bob”, 22, ‘Beijing’

“Amy”, 23, ‘Shanghai’

“Sam”, 22, ‘Shanghai’

输入输出:

|  |  |
| --- | --- |
| 输入 | 输出 |
| “Amy” | person{name: “Amy”, age: 21, hometown: “Beijing”} |
| “Annie” | false |
| 22 | [person{age: 22, hometown: “Beijing”, name: “Bob”}, person{age: 22, hometown: ”Shanghai”, name: “Sam”}] |
| 23 | [person{age:23; hometown: “Shanghai”, name: “Amy”}] |
| 3 | false; |
| {name: “Amy”, hometown: “Shanghai”} | [person{age: 23, hometown: ”Shanghai”, name: “Amy”}] |
| {name: “Amy”} | [person{age: 21, hometown: “Beijing”, name: “Amy”}, person{age: 23, hometown: “Shanghai”, name: “Amy”}] |
| {name: ”Annie”, hometown: ”Beijing”} | false |
| {name: “Daming”,  age: 22, hometown:”Beijing”} | false |

BONUS 1: 查找不同

要求：

针对进阶练习2，写一个diff函数。

参数为两个同学信息数组，返回一个数组——包含那些存在于第二个数组但是却不存在于第一个数组中的同学信息（比较时只看name）组成的数组。

思路：

1. 将两个数组排序；

2. 遍历数组二，如果元素不在数组一中，即search返回false, 则存入结果数组中。

源代码： 见 <src/diff.js>

或 <https://github.com/yangyanzhe/Homework2_web/blob/master/src/diff.js>

测试结果与分析：

1. 学生信息:

var stu1 = new person("Amy", 21, 'Beijing');

var stu2 = new person("Bob", 22, 'Beijing');

var stu3 = new person("Amy", 23, 'Shanghai');

var stu4 = new person("Sam", 22, 'Shanghai');

var stu5 = new person("Jane", 10, 'Anhui');

var stu6 = new person("Annie", 20, 'Henan');

var stu7 = new person("Daming",30,'Beijing');

var stu8 = new person("Jane", 21, "Beijing");

2. 表中测试样例为简化信息，具体格式为：

var first = new Array(stu1, stu2, stu3, stu4)

var second = new Array(stu2, stu5, stu6, stu3)

diff(first, second);

一个测试样例放在diff.js末尾。

|  |
| --- |
| 测试样例  数组1： [stu1, stu2, stu3, stu4]  数组2： [stu5, stu3, stu7] |
| 输出(简化)： [stu7, stu5] |
| 测试样例  数组1： [stu1, stu2, stu7]  数组2： [stu3, stu2] |
| 输出： false(设定如果找不到不同的元素，返回false) |
| 测试样例  数组1： [stu1, stu8, stu7]  数组2： [stu6, stu5, stu2] |
| 输出(简化) [stu6, stu2] |

3. 可以看出在测试样例下，结果正确。

BONUS 2：排序

要求： 将《算法导论》中的排序算法用JS实现

源代码： 见 src/sort.js

或 <https://github.com/yangyanzhe/Homework2_web/blob/master/src/diff.js>

测试示例：

var a = new Array(1, 2, 10, 7, 6);

test(a);

测试结果与分析：

|  |  |  |  |
| --- | --- | --- | --- |
| 输入 | 冒泡排序 | 插入排序 | 选择排序 |
| [1, 2, 10, 7, 6] | [1, 2, 6, 7, 10] | [1, 2, 6, 7, 10] | [1, 2, 6, 7, 10] |
| [9, 0, 5, 0, 8] | [0, 0, 5, 8, 9] | [0, 0, 5, 8, 9] | [0, 0, 5, 8, 9] |
| [3, 2, 9, 8, 3] | [2, 3, 3, 8, 9] | [2, 3, 3, 8, 9] | [2, 3, 3, 8, 9] |
| [-1, -3, 0, 9, 8] | [-3, -1, 0, 8, 9] | [-3, -1, 0, 8, 9] | [-3, -1, 0, 8, 9] |
| [3000000000000, 9, 8, 7, 10, 11] | [7, 8, 9, 10, 11, 3000000000000] | [7, 8, 9, 10, 11, 3000000000000] | [7, 8, 9, 10, 11, 3000000000000] |

BONUS 3: Untrusted游戏

|  |  |  |
| --- | --- | --- |
| 关卡名 | 通关方法 | 链接 |
| 1 | 把exit口挪到框里面才可以移动到exit处 | <https://gist.github.com/anonymous/8f2fbeac82ef83682486> |
| 2 | 把#墙注释掉 | <https://gist.github.com/anonymous/80bffd3c971f2a419193> |
| 3 | 将上面的线向上移动两个单位，就有了一个出口 | <https://gist.github.com/anonymous/5d6406505a52c5e36863> |
| 4 | 放一个exit出口在@旁边 | <https://gist.github.com/anonymous/2c611a3972d599615b53> |
| 5 | 使用API的函数，setSquareColor，从而看到雷在哪儿 | <https://gist.github.com/anonymous/6d51e9e38523f247d6bc> |
| 6 | 建立一面墙，可以挡住d使其只能水平移动 | <https://gist.github.com/anonymous/c4de3bef03ddc570f877> |
| 7 | 使用atLocation的API函数，到达该位置时，按下Q，调用phone的函数，使@和色块颜色一样，就可以过去了 | <https://gist.github.com/anonymous/9baa944f2030bdfeb81b> |
| 8 | 每次走一点，按下Q重新随机化森林，总可以再近一点，再近一点 | <https://gist.github.com/anonymous/60cec12579c4fcb12d4a> |
| 9 | 类似raft构造一个object，将type设为dynamic,就可以搭在已经有物体的位置，transport改为true，就可以不怕水 | <https://gist.github.com/anonymous/3c7bb5bf7508fe83c18d> |
| 10 | 主要问题是红色和绿色的中间的两个，添加判断语句.getY()来得到两个元素，接着判断，如果可以上移就上移，不能上移就左移，从而使障碍物把中间让出来 | <https://gist.github.com/anonymous/e18b9f1f42806940cc1a> |
| 11 | 设置机器人，能右移就右移，如果不能右移就下移。这样随便移动@，会使机器人先拿到Key,然后会卡死在右下角，找机器人拿走Key，就可以去exit处过关了 | <https://gist.github.com/anonymous/d656ee2659270f85a387> |
| 12 | 利用API中的函数，object.getX()来判断位置，设置Robot行走的路径，顺利拿到钥匙，同时，@在最右侧等着R，接到钥匙就可以到exit处了 | <https://gist.github.com/anonymous/61cc8cd8de19efbe59ae> |
| 13 | 设置了如果@在某一个位置，控制R的左右移动，像遥控器一样，然后就可以人工走迷宫了 | <https://gist.github.com/anonymous/76a7e3eb9e86336c0e2a> |
| 14 | 试了一下还是要改代码，改成了blue，就绕着一圈走了一下，拿了A，就走了 | <https://gist.github.com/anonymous/ad55357864ff28df8964> |
| 15 | 这样是不是很邪恶……如果被kill了，就再新place一个放在下面 | <https://gist.github.com/anonymous/a04a7a0589b15bf06436> |
| 16 | 首先要显示出来线的颜色，把white改成color；然后借用之前的phone的回调函数，按一下Q改变一个颜色，找到与线匹配的颜色即可通过 | <https://gist.github.com/anonymous/3a548b89a10201cfe485> |
| 17 | 利用16中画线的代码，如果两遍都是teleporter，就说明是通路，画一条线连接两个teleporter。使用canvas的API，这样就很容易沿线走找到exit。 | <https://gist.github.com/anonymous/0f07e1f44b318faad876> |