爱创课堂前端培训

javascript进阶贪吃蛇项目

班级：爱创课堂十七期

讲师：彭帅伟

日期：2018年10月10日

# 复习：

构造函数：

与普通函数的区别：

定义方式: 没有区别 （构造函数的首字母要大写 非强制要求）

目的：

普通函数： 实现某一功能

构造函数： 创建对象

使用方式：

普通函数： 直接调用

构造函数: 使用new 关键字 来调用

构造函数的四步：

隐秘的开辟一个新的内存地址

与this绑定

执行函数中代码

返回this

原型：每一个函数都有一个属性prototype，它的值是一个对象

特点：原型上的所有内容，都可以被实例化对象所访问

作用： 实例共享方法

原型查找：

当一个对象在调用一个方法的时候，先找自身，自身如果没有就找自身的原型，原型中还没有此方法，就会继续找原型的原型，直到终点为止。

hasOwnProperty: 该方法检测某一方法是在构造函数上还是在原型上。

继承：

分为3种：

1 类式继承

将子类的原型指向父类的实例

这种继承方式会丢失原来子类的constructor属性，所以要手工补回

2 构造函数式继承

在子类的构造函数中，执行父类的构造函数并使用apply改变this指向，然后将arguments传递，子类特有的属性要写在构造函数式的下面

3 组合式继承

类式继承 + 构造函数的继承

安全类： 无论外部如何调用，都会返回类的实例。

instanceof: 该关键字用于检测 某个对象是否是某个构造函数的实例

内置构造函数：

Object Array Function Number String Bloon RegExp Deta Error

Function:

该内置构造函数用于定义函数

使用方式：

接受任意个字符串参数，除了最后一个都是形参

RegExp:

该内置构造函数用于定于正则表达式

使用方式：

接受两个参数

第一个参数是： 定义正则表达式的表达体

第二个参数是： 正则表达式的修饰符 i, g, m

# 内置构造函数

## String

这是一个内置构造函数， 它是string值类型的对应包装类型

这也就是为什么字符串可以调用方法的原因

举例：

|  |
| --- |
| 1. // String 是 string值类型的对应包装类型 2. var str = "abcdefg"; 3. console.log(str[1]); 4. // 转为对应的包装类型 5. var str\_obj = new String(str); |

输出：

|  |
| --- |
|  |

## Boolean

这是一个内置构造函数， 它是bool值类型的对应包装类型

举例：

|  |
| --- |
| 1. // Boolean 2. // 它是bool值类型的对应包装类型 3. var f = false; 4. // 转为对应的包装类型 5. var f\_obj = new Boolean(f); 6. // f是new Boolean对象的原始值， 而new Boolean是f的对应包装类型 7. // !f true !f\_obj false 因为f\_obj是一个对象 8. // console.log(f\_obj.valueOf()) |

## Number

这是一个内置构造函数， 它是number值类型的对应包装类型

举例：

|  |
| --- |
| 1. // Number 2. var num = 3; 3. // 转为对应的包装类型 4. var num\_obj = new Number(num); 5. // Number本身不是一个安全类， 当Number自执行的时候， 会作为类型转为函数使用 6. var a = "3a"; 7. var aa = Number(a); |

# 内置构造函数之间的关系

所有的函数都是Function的实例

所有的对象都是Object的实例

函数也是对象

# 贪吃蛇

我们决定使用面向对象的方式书写贪吃蛇游戏

游戏是最合适使用面向对象的方式书写  
 我们把整个游戏看成是一个“游戏”类

游戏类可以有各种各样的属性

我们可以把贪吃蛇中的蛇作为一个属性

还有地图属性

食物属性

障碍物属性

我们可以把蛇当做一个“蛇”类

有数组属性

有方向属性

有增长方法

有移动方法

可以把地图看成一个地图类

有行属性

有列属性

有宽度属性 （总宽）

有高度属性 （总高）

有一个数组属性，数组中存放的是每一个小方格

可以把食物看成一个食物类

有一个x属性

有一个y属性

img属性

可以把障碍物看成是一个类

有一个数组属性

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |