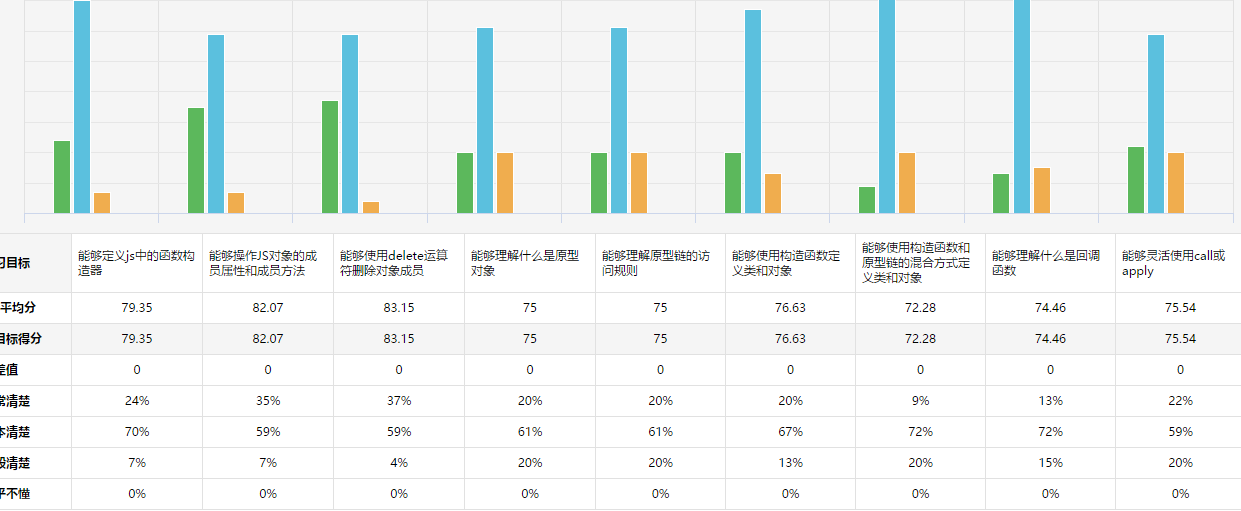
# 一、回顾



1、什么样的函数是构造函数？

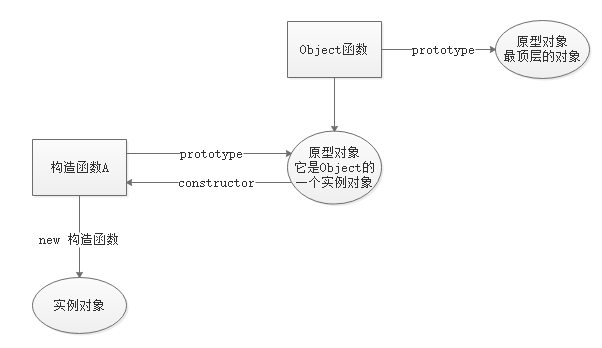
一个函数被new了，那么这个函数就是构造函数，或者叫做构造器，或者叫做“类”。

2、能够对对象的成员进行增删改查操作

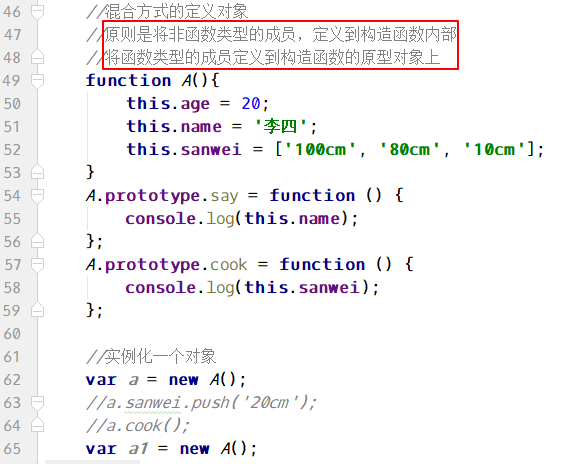
3、什么是原型对象

原型对象，它是在一个对象被创建的时候，由系统自动分配的一个对象。所有自定义的对象的默认的原型对象是Object的一个实例。

原型对象和实例对象和构造函数的关系：



4、混合方式定义对象



另外的指定原型对象的写法：



5、回调函数

就是一个函数的参数是另外一个函数，那么这个参数可以叫做回调函数。

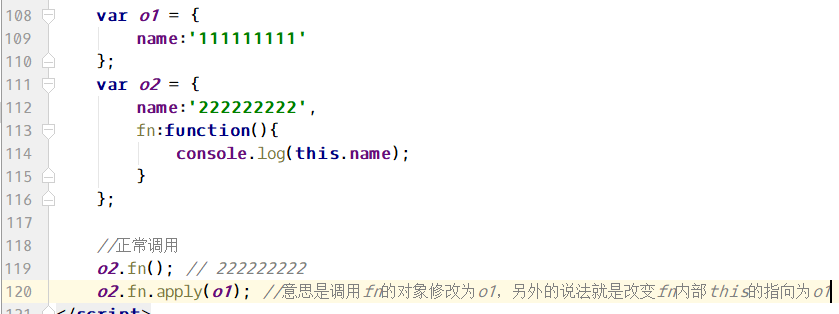
6、apply、call、bind

他们三个都可以改变一个函数内部this的指向。改变的是哪个函数呢？哪个函数来调用apply、call、bind，改变的就是这个函数内部的this。

apply(this指向的新对象, [数组形式的其他参数，传递给调用apply的那个函数]); //会自动执行调用apply的函数

call(this指向的新对象, 其他参数，传递给调用call的那个函数); //会自动执行调用call的函数

bind(this指向的新对象, 其他参数，传递给调用bind的那个函数); //不会自动执行调用bind的函数



# 二、今日目标

1、理解原型链的访问规则

2、了解Object函数及Object函数的原型对象上的方法

3、能够实现JS对象的继承

4、能够明白事件处理函数中的this指向

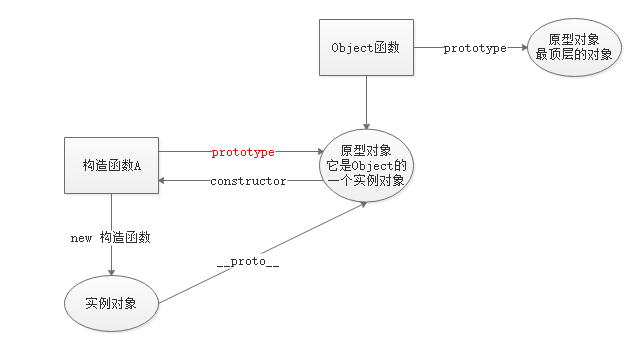
5、能够明白构造函数中的this指向

6、能够明白定时器中的this指向

# 三、Object

1、介绍

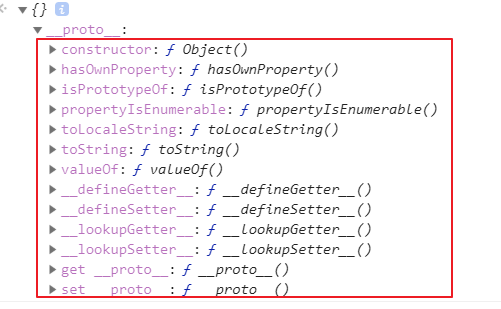
Object也是JS内置的一个构造函数，它的实例通常被认为是所有其他对象的原型对象。实际上Object函数也有原型对象，这个Object的原型对象才是最顶层的一个对象。总之，根据原型链的关系，Object构造函数内提供的一些方法，以及Object函数的原型对象中的方法，都会为每个实例对象所拥有。



Obejct函数中自带的一些成员：



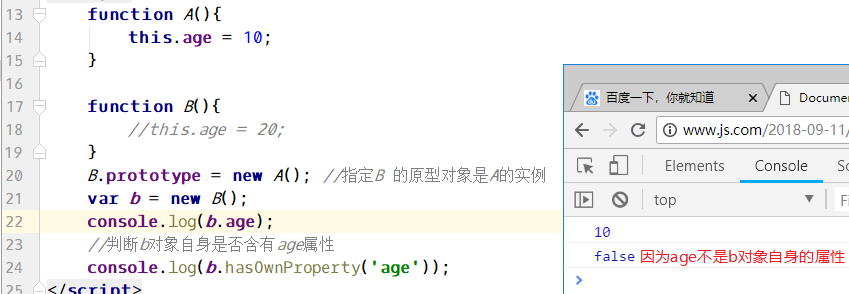
Object的原型对象的成员：



2、hasOwnProperty(property)

hasOwnProperty方法是Object函数的原型对象上的方法。

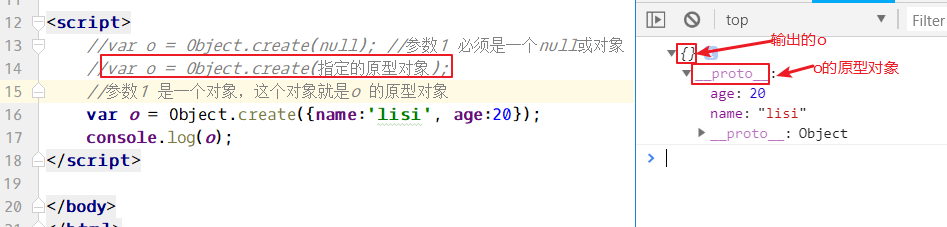
hasOwnProperty方法用于检测实例对象是否有非继承的属性，也就是自身有该属性。



3、create()

create()方法是Obejct构造函数内的成员方法，而且是一个静态方法（调用方式是直接使用Object来调用）。

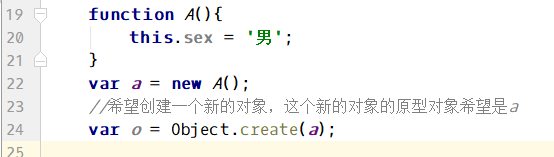
create()方法用于创建对象，创建对象的时候，可以指定对象的原型对象。



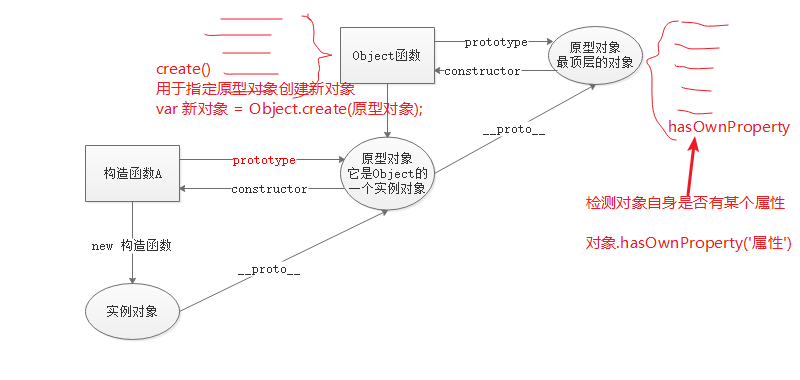
var o = Object.create({原型对象}, {o对象的内容});

var o = Object.create({父对象}, {子对象o本身});

或者另外的写法：



4、小结



# 四、继承

JS要实现继承，有非常多的方式，除了下面列举的三种以外，还有for…in可以实现继承。ES6中有extends关键字，可以实现继承。

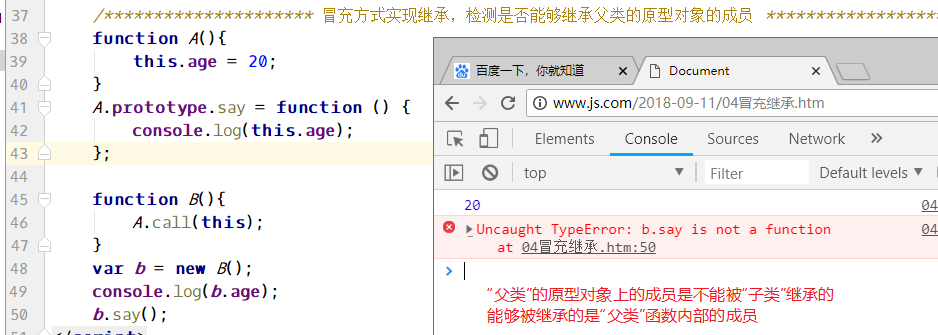
1、冒充继承

案例中定义两个构造函数（类），父类中定义几个成员，然后让子类的实例去调用。如果调用成功，表示实现了继承。

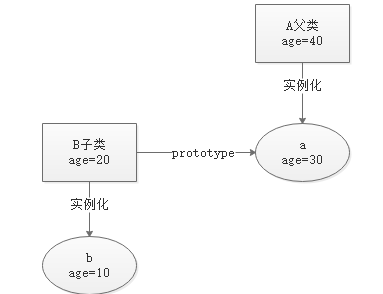
**父类.call(子类的对象, 其他参数); //意思是让子类的对象，去代替父类中的this**。



这种实现继承的方式，是不能继承原型对象上的成员的：



2、原型链继承

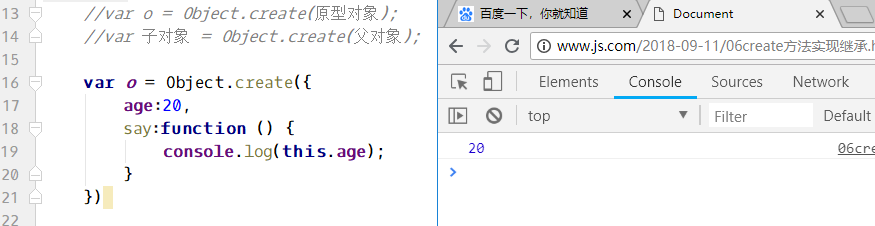




3、create()继承

create()是IE9才开始支持的。

var 新对象 = Object.create(原型对象); 该方法就是用于指定原型对象的。所以就可以直接使用create方法实现继承。

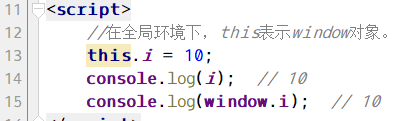


# 五、this指向总结

this永远指向一个对象。

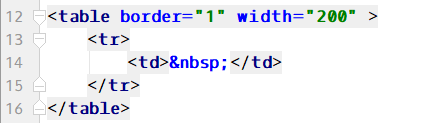
**this指向的对象是谁，关键要看this运行的环境。也就是this所在的函数被赋值给哪个对象了，那么this就表示这个对象**。

全局环境下，this表示window对象。

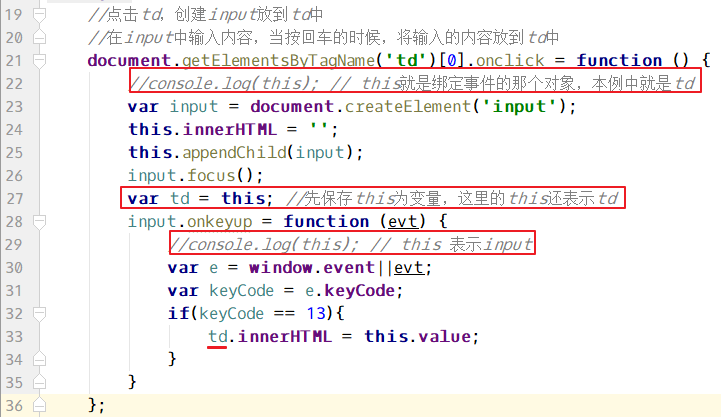


1、事件处理函数中的this

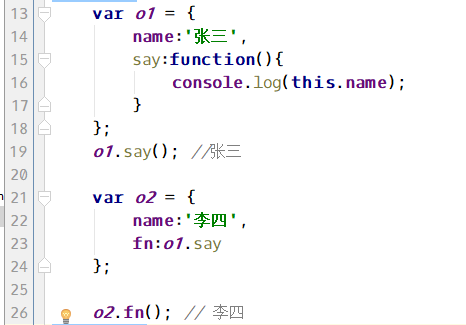
HTML：



JS：

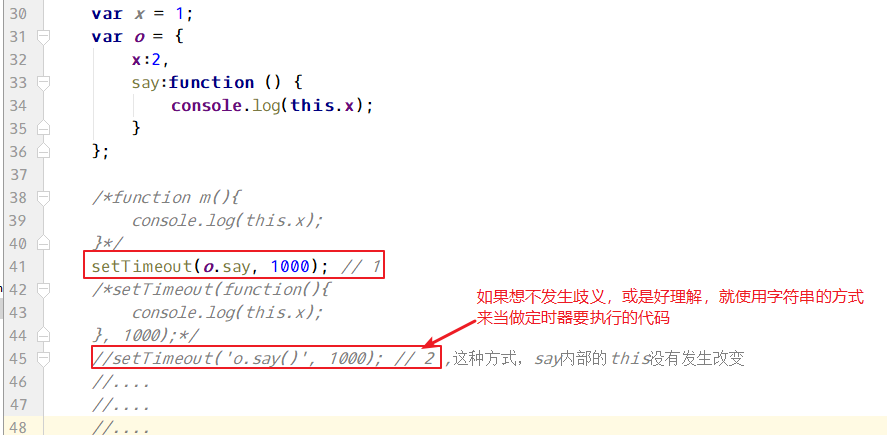


2、面向对象中的this



区分this表示哪个对象，关键看调用函数的时候，函数是由谁来调用的。由o1调用，this就表示o1，由o2调用，this表示o2.

3、定时器中的this



# 六、案例

1、分析

小游戏中共有三个对象。一个是地图（Map）、一个是食物（Food）、一个是蛇（Snake）。

以面向对象的方式来开发，应该创建三个对象。

每个对象的特点：

地图Map：

宽width

高height

背景颜色backgroundColor

显示方法：show

食物Food：

宽width

高height

背景颜色backgroundColor

x轴坐标：0~29(每个坐标实际像素是20px)

y轴坐标：0~19（每个坐标实际像素是20px）

蛇Snake：

身体body（默认有三个div组成，其中一个蛇头，两段蛇身，都有颜色）

能够移动move

能够拐弯，改变移动方向

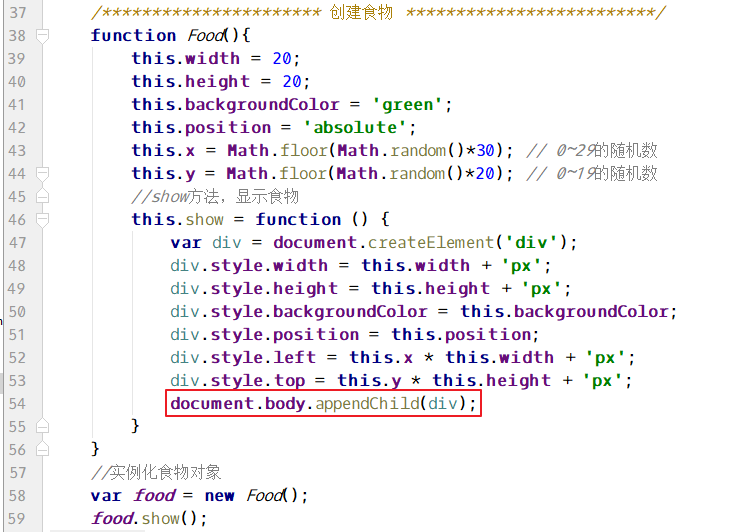
能够吃食物，变长

2、创建地图



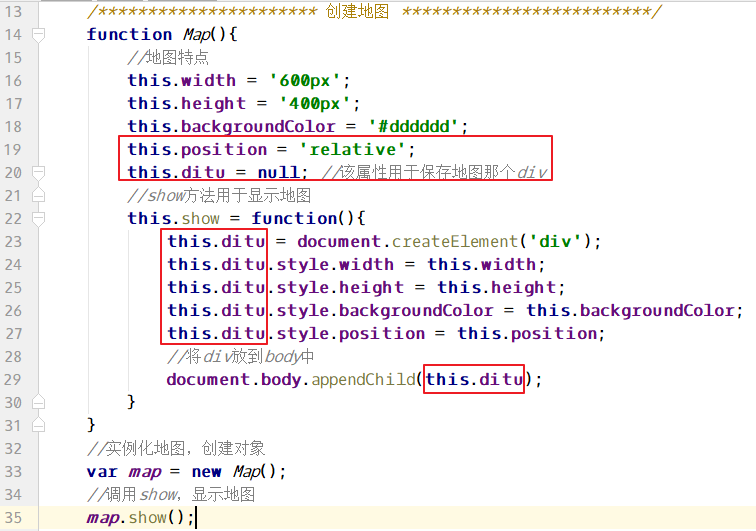
3、创建食物

创建一食物，让它显示在body中。



食物没有和地图对齐，解决办法是让父级元素地图position是relative，让子元素食物position是absolute，并且让食物显示在地图中，而不是显示在body中。

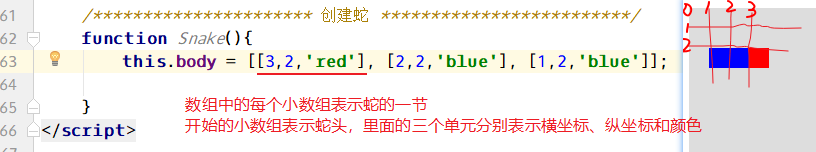
所以修改Map，添加position属性，并且用一个属性\_map来保存地图：



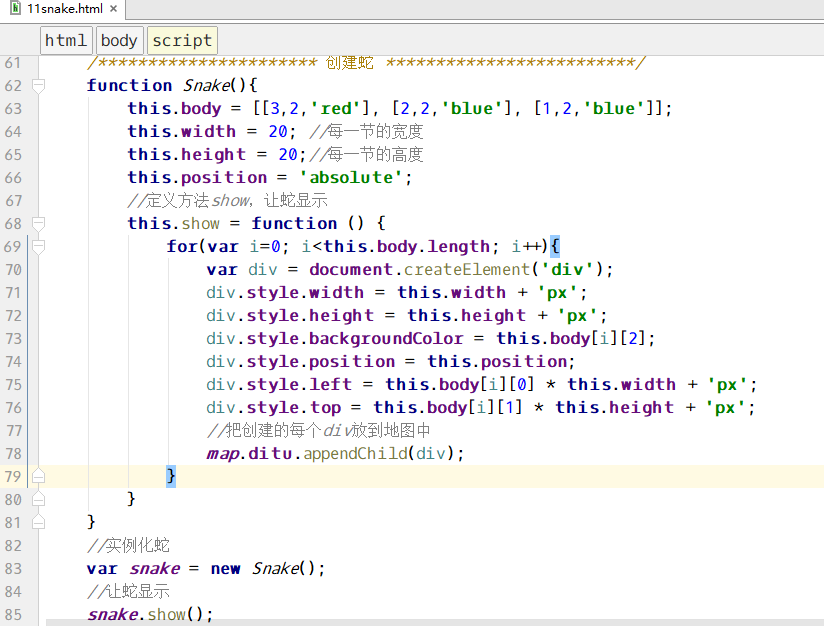
然后修改食物，让其显示在地图中即可：



4、创建蛇

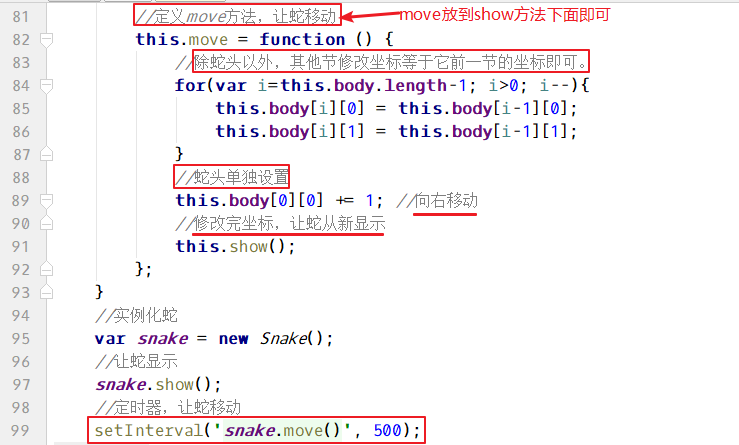


蛇的每一节用一个div来表示，每一节都有宽度、高度、背景颜色、坐标。创建蛇，并让它显示到地图上。



5、让蛇移动

移动方向：默认先向右移动。



这样做有个问题，当蛇移动的时候，会调用show方法，让蛇从新显示。调用show方法，会一直循环创建div。所以看到的效果是蛇移动了，但是越走越长。

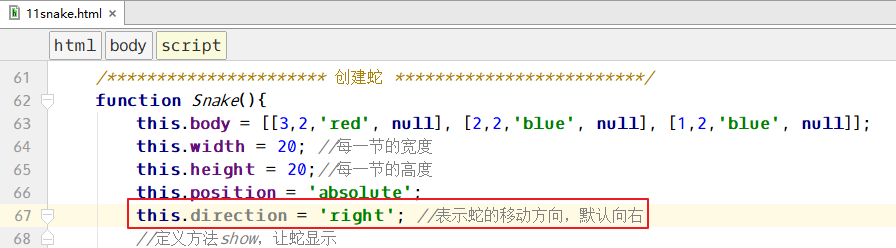
解决办法：



6、让蛇拐弯

控制蛇头即可，因为身体一直跟着蛇头走。

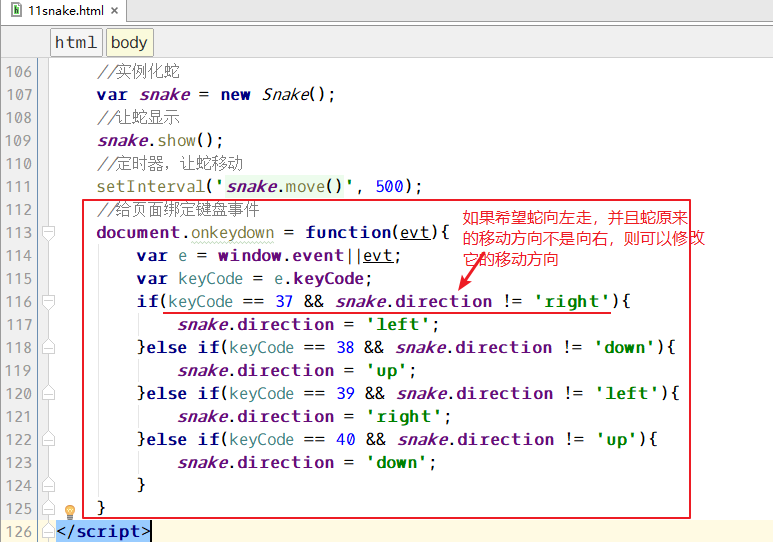
蛇头的方向肯定是上下左右四个方向，可以用一个属性来表示蛇的移动方向。



当蛇移动的时候，根据方向修改蛇头的坐标：

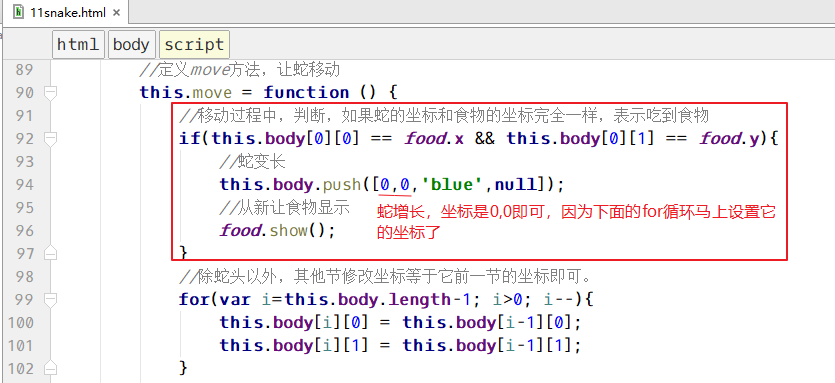


下面添加键盘事件，修改蛇的运行方向：

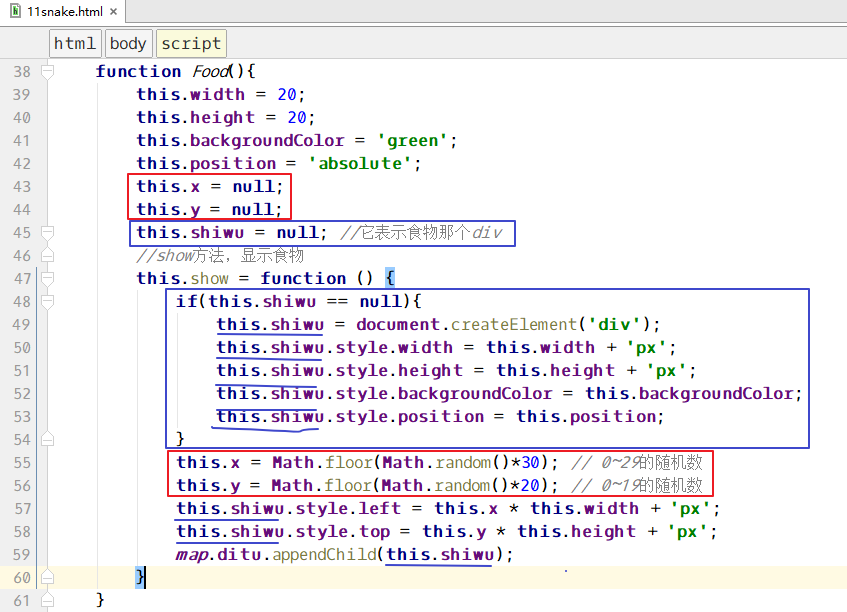


7、让蛇吃食物

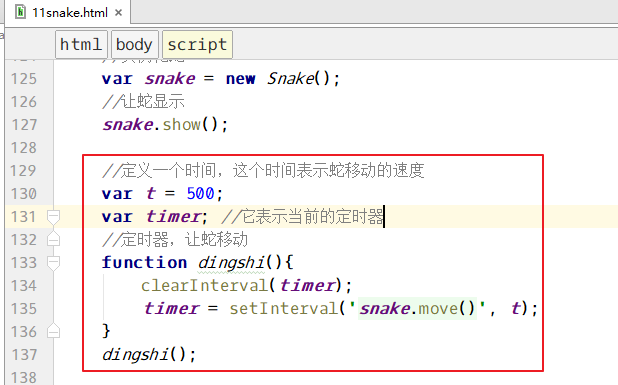
蛇头和食物重叠，才算吃到食物。即在移动过程中，如果发现蛇头的坐标和食物的坐标一样，才表示吃到食物。



这样做，吃完食物，发现食物没有从新显示。修改Food函数：



8、让蛇加速



当蛇吃到食物的时候，让时间变小，速度加快：

