## 贪吃蛇需求分析

**要求：**

1.地图：蛇的活动区间

2.蛇：蛇可以在地图中按各个方向走，吃掉食物，蛇的长度可以增加

3.食物：食物随机出现在地图中的各个位置

**总结：**

地图：利用DOM创建div，通过给元素的属性赋值设置地图div的样式

蛇：用一个数组来存储初始蛇身的基本信息

蛇具有的属性：横坐标，纵坐标，方向，创建蛇身的div元素

var snakeBody=[

[3,5,"right",null],

[2,5,"right",null],

[1,5,"right",null]

];

做一个循环遍历数组，当数组中的第二层的第4个元素（snakeBody[i][3]）为空时，

利用DOM创建元素div，给div设计样式作为蛇身。

对于蛇移动：

数组中的后一个元素移动到前一个元素的位置，即：

snakeBody[2]=snakeBody[1];

snakeBody[1]=snakeBody[0];

通过键盘事件控制蛇的走向。通过函数中的event参数,由event.keyCode获取所按键盘的ASC2码的值，选中某四个键当作方向键。在该键盘事件调用的函数（蛇移动）中书写判断所按键盘的值，控制蛇的走向及坐标的移动情况。

**创建食物：**

利用DOM创建div盒子，给盒子设置样式。通过随机数函数产生盒子横纵坐标的随机数（

地图的宽/食物的尺寸利用Math的属性产生随机数获得横坐标的格子数，地图的高/食物的尺寸利用Math的属性产生随机数获得纵坐标的格子数，然后再与尺寸相同）

另起一个函数放入食物的位置，之后要用到

蛇吃到食物：蛇头的横纵坐标与食物的横纵坐标相等（用if判断，满足条件，调用食物位置的函数，实行食物被吃到后的移动）

蛇吃到食物后，蛇身要长长一节，

这时需要传递蛇身每一节的方向

for(var i=snakeBody.length-1;i>0;i--){

snakeBody[i][2]=snakeBody[i-1][2];

}

新的一节蛇身的方向与旧蛇蛇尾的方向相同

根据蛇尾的方向，设置新加的蛇尾的横纵坐标

//创建新蛇

var x;

var y;

var newDirection=snakeBody[snakeBody.length-1][2];

//要先获取旧蛇的方向，然后根据方向判断新蛇的位置

//传递旧蛇的方向

//创建新蛇

var x;

var y;

var newDirection=snakeBody[snakeBody.length-1][2];

//要先获取旧蛇的方向，然后根据方向判断新蛇的位置

//传递旧蛇的方向

for(var i=snakeBody.length-1;i>0;i--){

snakeBody[i][2]=snakeBody[i-1][2];

}

switch(snakeBody[snakeBody.length-1][2]){

case "right":

x=snakeBody[snakeBody.length-1][0]-1;

y=snakeBody[snakeBody.length-1][1];

break;

case "left":

x=snakeBody[snakeBody.length-1][0]+1;

y=snakeBody[snakeBody.length-1][1];

break;

case "up":

x=snakeBody[snakeBody.length-1][0];

y=snakeBody[snakeBody.length-1][1]+1;

break;

case "down":

x=snakeBody[snakeBody.length-1][0];

y=snakeBody[snakeBody.length-1][1]-1;

break;

default:

return;

}

var newBody=[x,y,newDirection,null];

snakeBody.push(newBody);

createSnake();

switch(snakeBody[snakeBody.length-1][2]){

case "right":

x=snakeBody[snakeBody.length-1][0]-1;

y=snakeBody[snakeBody.length-1][1];

break;

case "left":

x=snakeBody[snakeBody.length-1][0]+1;

y=snakeBody[snakeBody.length-1][1];

break;

case "up":

x=snakeBody[snakeBody.length-1][0];

y=snakeBody[snakeBody.length-1][1]+1;

break;

case "down":

x=snakeBody[snakeBody.length-1][0];

y=snakeBody[snakeBody.length-1][1]-1;

break;

default:

return;

}

利用push()将新的元素放入蛇身的数组中

var newBody=[x,y,newDirection,null];

snakeBody.push(newBody);

createSnake();

要使蛇不停的走下去，用一个计时器来实现setTimeout/setInterval(蛇移动的函数,执行一次的时间)

判断什么时候游戏结束

当蛇头与四周墙壁碰撞及蛇头碰到蛇尾时游戏结束：

蛇头与墙壁碰撞：蛇头的横纵坐标等于0或地图的宽度或高度（宽对应宽，高对应高），清除计时器，使程序不再被加载。

后续要实现的功能：

**实现选关**：

思路：利用DOM创造下拉框，获取选中的选项的值，将每个值，作为sw ith的条件，满足，某一个条件时，执行某个语句。