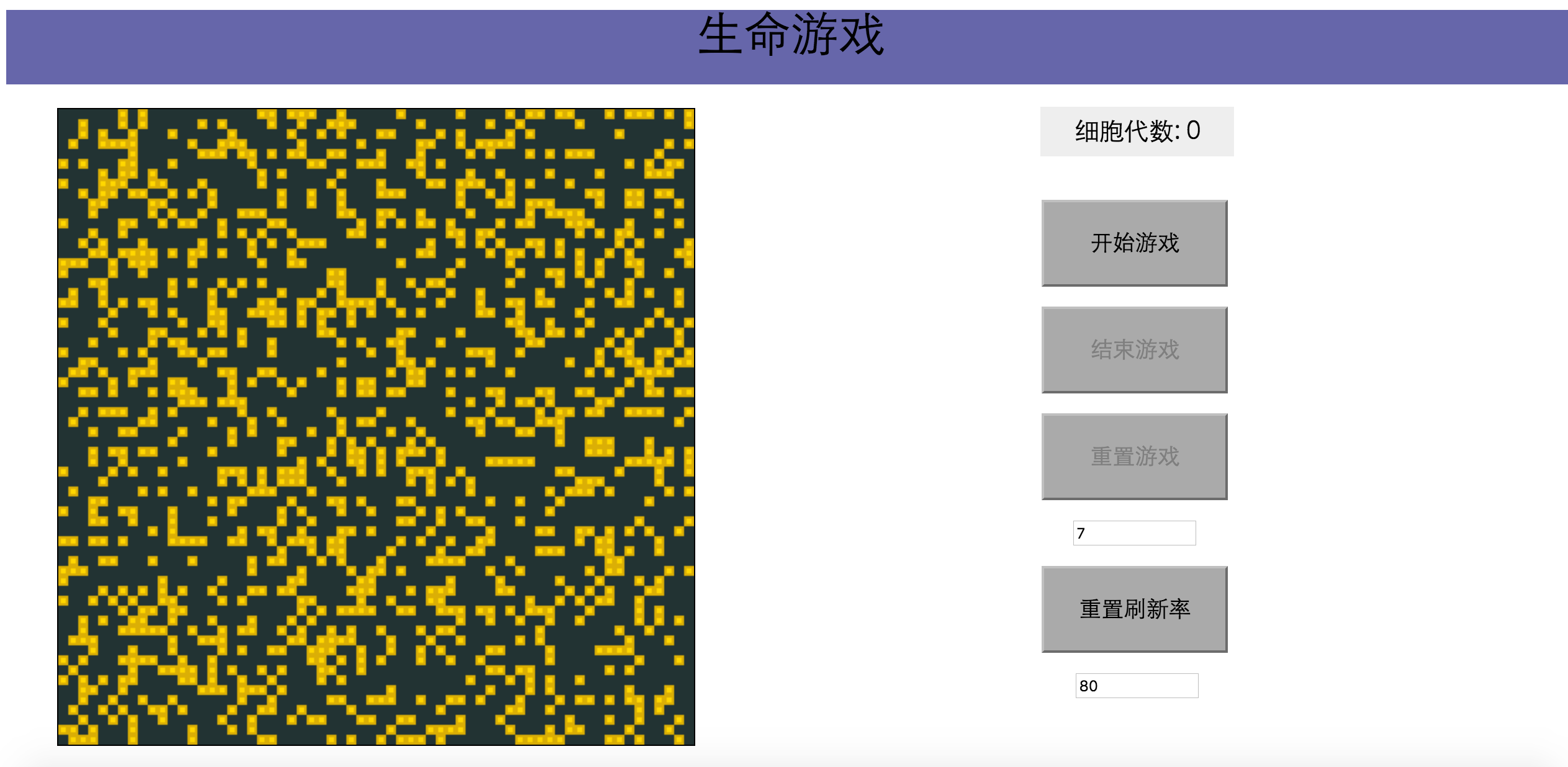
# 生命游戏程序说明文档

1. **作业目的**
2. 提高代码的设计质量，增强代码的可读性、可修改性和可扩展性；
3. 掌握单元测试技术，学会运行测试驱动的开发方法。
4. **生命游戏实现内容及方法**

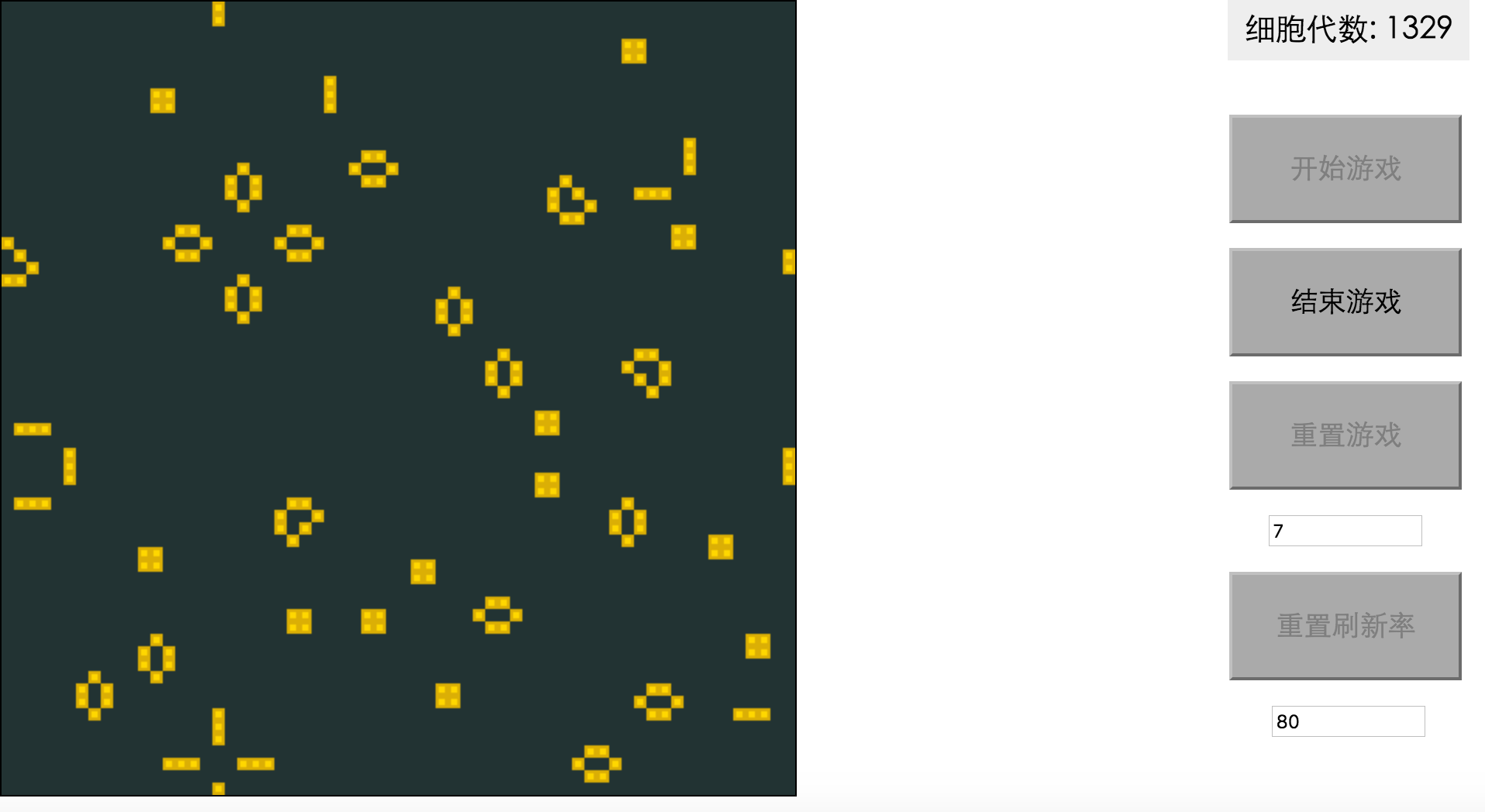
**内容**：

主要运用JavaScript实现生命游戏，网页效果图如下：

左侧为细胞效果图，右侧为控制按钮；

点击‘开始游戏’，则细胞开始进行生命的更新变化，在“细胞代数”出会显示细胞更新的次 数。点击‘结束游戏’，细胞更新会停止，在重置游戏下面的输入框中输入一个数n(5 <= n <= 10),再点击‘重置游戏’按钮，则图中左侧的细胞的大小会发生改变；更改‘重置刷新率’按钮下的输入框中的数值，再点击‘重置刷新率按钮’，游戏的刷新速度会有所改变（数字越大，刷新速度越慢，单位：ms）。

最终实现的效果图之一如下：



**方法**：

1. 用HTML和CSS写好基本的页面的格式，用JS来实现一些逻辑设置；
2. 利用canvas绘图，将一个table分成很多个小格，每个小格是否绘制“细胞”与它周围的八个小格是否有“细胞”有关系（及生命游戏的规则）。先随机的将每个小格设置为1（存活）或0（死亡），然后运用生命游戏的规则（计算下一次此细胞是否存活），不断地进行变换（用setTimeout()函数实现定时器的迭代）。
3. 游戏的重置、暂停等功能都是通过getElementById来获取id，然后更改变量实现的。
4. 呈现细胞变化的页面的长和宽都设置为512，所以（细胞的宽度\*细胞的行数）或者（细胞的宽度\*细胞的列数）都得等于512。
5. **测试实现**
6. 单元测试的环境是mocha
7. 测试用例的设计：
8. 用assert.isFunction()函数测试函数是否满足是函数的要求；

结果：所写的七个函数均满足。

1. 用assert.equal()来判断变量的个数是否正确；

结果：有变量的前两个函数均测试正确。

1. 用assert.throws()和expect(fun).to.throw(Error)均判断各个函数是否会抛出异常；

结果：七个函数均可抛出异常处理。

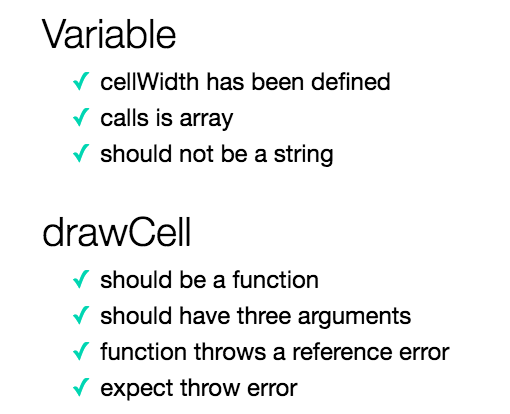
1. 用assert.strictEqual()函数检查判断返回值是否严格的相等；

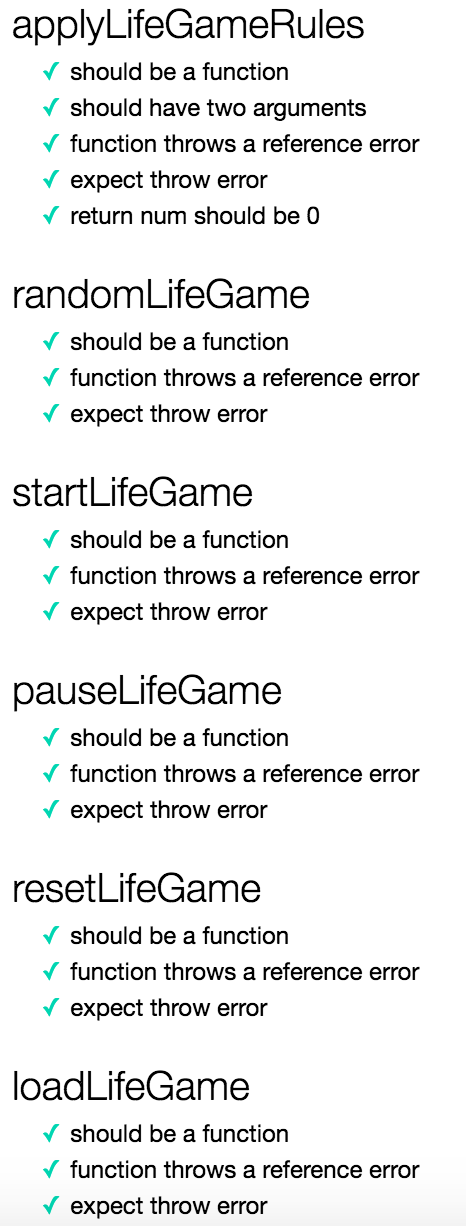
结果：抽取有返回值的applyLifeGameRules(1,1)来进行测试，算出其返回值结果为0，将测试结果设为0，测试函数的返回值正确。

1. 用assert.isDefined()判断变量是否定义，用assert.isArray()判断是否是数组，用 expect(val).to.not.be.a('string')判断变量的数据类型；

结果：所抽取变量来进行测试的结果均正确。

以下是测试输出的结果图：





1. **实验总结**

总的来说本次试验主要是js逻辑的实现比较重点，以及程序需要进行多方面的单元测试。总体感觉自己的代码风格居中水平，代码模块设计了多个函数分别拥有不同的功能，分成不同的模块。

Js水平着实有待提高~

软件42 刘想

2014013447

[LX1739871502@163.com](mailto:LX1739871502@163.com)