

**BlackJack游戏说明文档**

罗徐玲



2016-4-14

目录

[一、 实现功能 2](#_Toc448409162)

[二、 单机版概要设计 2](#_Toc448409163)

[三、 联机版概要设计 4](#_Toc448409164)

[联机实现——socket.io 4](#_Toc448409165)

[逻辑实现——Client&Server 5](#_Toc448409166)

[类设计 5](#_Toc448409167)

[交互设计 6](#_Toc448409168)

[界面实现 8](#_Toc448409169)

[主页面框架设计 8](#_Toc448409170)

[其他设计 8](#_Toc448409171)

[四、 使用说明 8](#_Toc448409172)

[1. 游戏开始 9](#_Toc448409173)

[正常开始游戏 9](#_Toc448409174)

[多个玩家同时创建房间 10](#_Toc448409175)

[无法进入房间情况 11](#_Toc448409176)

[2. 玩游戏 12](#_Toc448409177)

[3. 离开游戏 15](#_Toc448409178)

[普通玩家离开游戏 15](#_Toc448409179)

[庄家离开游戏 16](#_Toc448409180)

[五、 可改进点 16](#_Toc448409181)

# 实现功能

* 1至6人同时游戏；
* 可使用1至5副扑克；
* 玩家功能有：下注，开始游戏，买保险，加牌，停牌；
* 庄家功能有：开始游戏，加牌，停牌；
* 多人联机游戏；
* flex布局。

# 单机版概要设计

使用JavaScript，html和css，没有使用其它框架。

采用面向对象的思想进行开发。将整个游戏分为四类：扑克（poker），玩家（wj），庄家（zj），人（person）。

扑克类负责分发扑克。人负责该玩家所在块的界面初始化，按钮事件初始化，扑克牌分数计算等。

* 扑克类图如下：



* 人类图如下：



* 庄家类图如下：



* 玩家类图如下：



其中，玩家和专家都继承自人类，关系图如下：



单机版逻辑较为简单。不详细描述。逻辑和界面设计在联机版进行介绍。

# 联机版概要设计

## 联机实现——socket.io

本项目使用soket.io来实现联机作战。为什么选择socket.io呢？理由如下：

1. socket.io消息的传递是基于传输的，而非全部依赖webSocket，因此可以在绝大部分浏览器上运行，从IE6到IOS都支持。
2. socket.io自动使用像long polling一样复杂的技术，但其API保持了与websocket一样的简洁。
3. socket.io监听的是connect事件而不是open事件，disconnect事件而不是close事件，因此提供了可靠的事件机制。在实际情况下，应用不依赖TCP连接一定能够正常关闭。因此，可以专注应用逻辑本身，无需过多担心网络不确定的情况。

使用socketio，io.on('connection', function (socket) {})监听所有连接到io端口的socket，利用此socket在服务器和客户端之间交互信息，触发各种事件。

## 逻辑实现——Client&Server

### 类设计

要将单机版游戏改为联机版，主要是分析两个问题：哪些类在客户端哪些在服务器端，这些类之间如何进行交互。

对于第一个问题，如果每个客户端都生成一个扑克牌实例会导致牌的分发无法统一，因此，扑克牌肯定要放在服务器端；玩家类和庄家类放在客户端，在每个客户端生成person实例，负责该端的页面修改和按钮事件等。

联机版较单机版多一个room（房间）类，顾名思义是这个房间的所有属性和操作的封装。其中包括玩家个数，扑克牌实例，等。下图是room类图：



因此，room和poker放在服务端，poker在room中创建。

wj和zj放在客户端。

### 交互设计

游戏大致可以分为四个步骤：

* 创建房间
* 选择位置加入游戏
* 开始游戏
* 停牌和记分
* 离开房间

下面是这些过程的交互图：



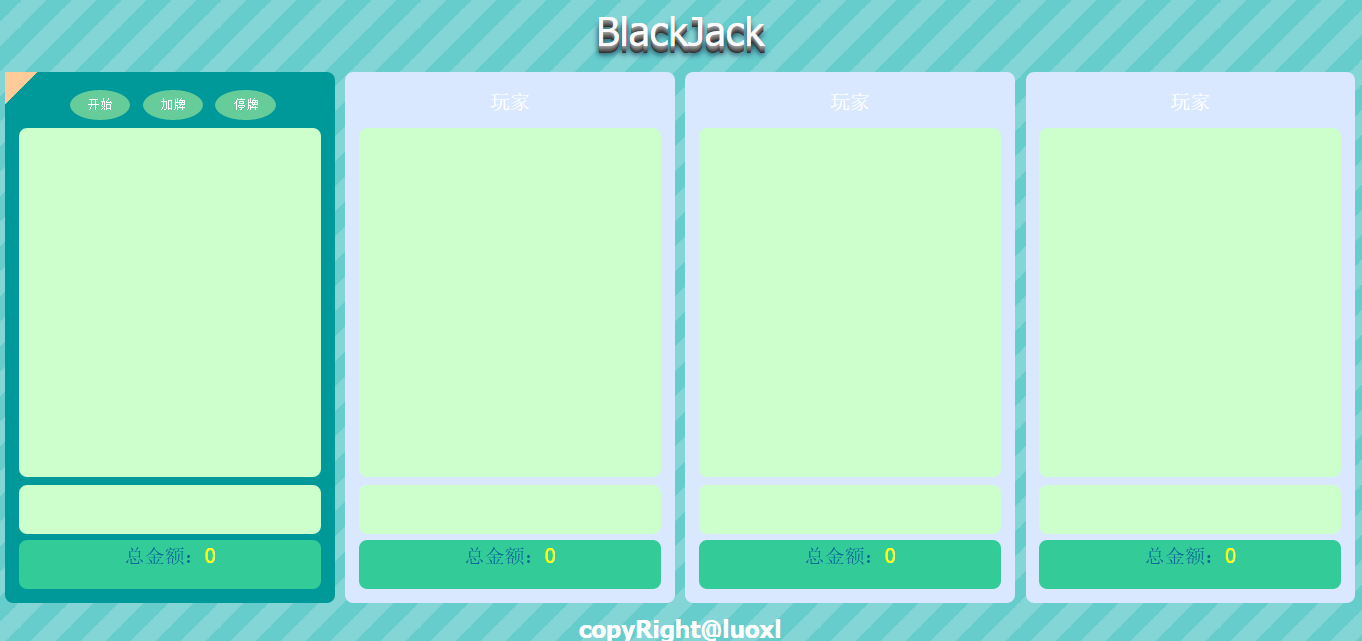
主要控制游戏的是person界面上的四个按钮：开始游戏，加牌，停牌，下注。这些按钮需要控制按键的情况。有些情况需要避免，比如点击停牌后未开始游戏就点击加牌，此加牌应该无效。按键状态图如下：



## 界面实现

### 主页面框架设计

使用flex布局能够使界面在收缩中显得较为自然。



上图界面分为三个部分包裹在flex元素（flex-direction: column;）中，分别是头部，中间和底部。中间部分的flex-grow设计的比头部和底部要大。

中间部分设置为一个flex元素（flex-direction: row;）。里面的cur元素标记当前玩家，这个玩家设计了min-width属性，避免窗口缩很小时无法正常显示按钮。

每个玩家设置为一个flex元素（flex-direction: column;）。分为四个区域：按钮/身份区，扑克牌显示区，当前状态区和总金额显示区。

### 其他设计

使用animation，transform，rotate，box-shadow，text-shadow，linear-gradient实现loading效果，发光按钮，凸出字体，条纹背景。

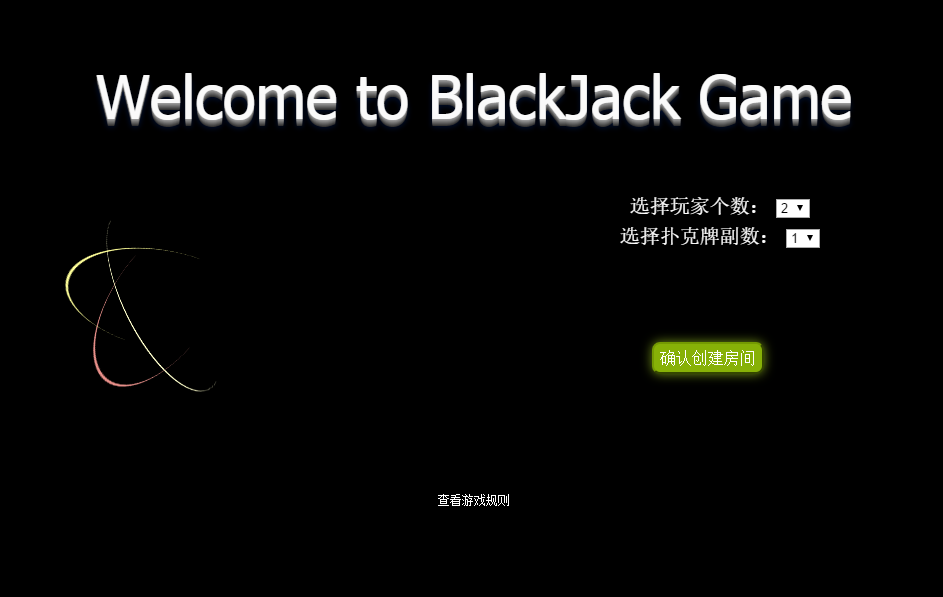
# 使用说明

在根目录下输入 node app.js，访问localhost:3000。

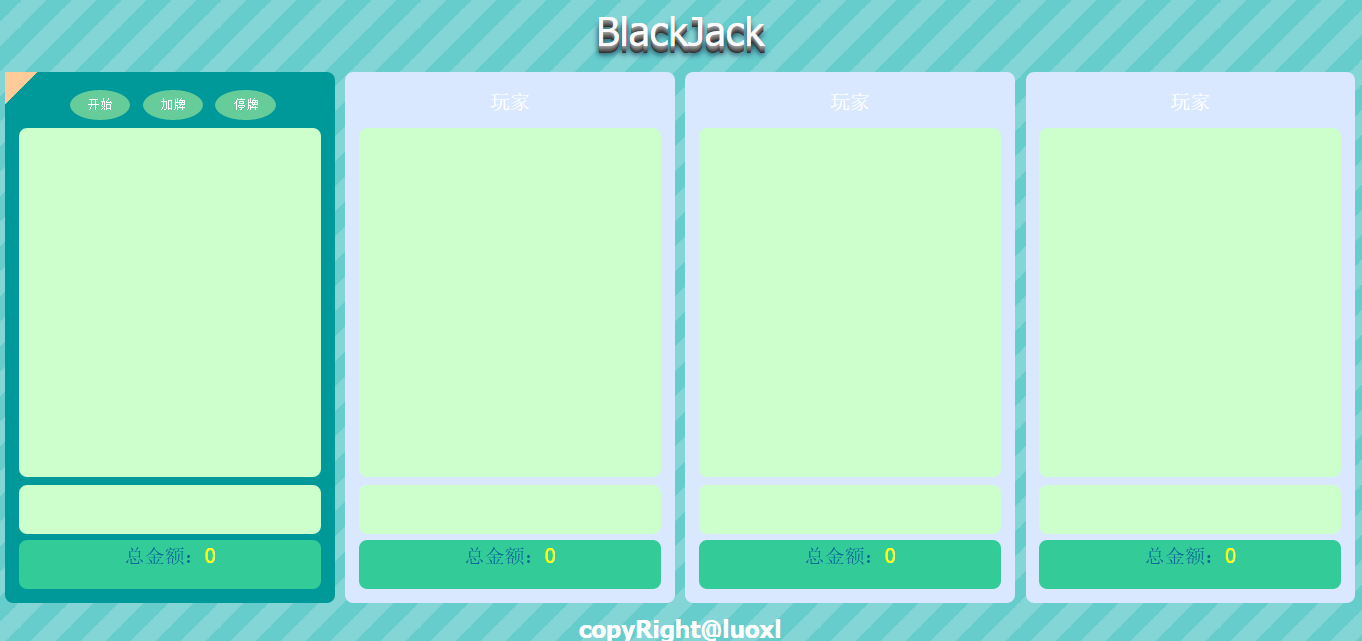
## 游戏开始

### 正常开始游戏

进入首页面：

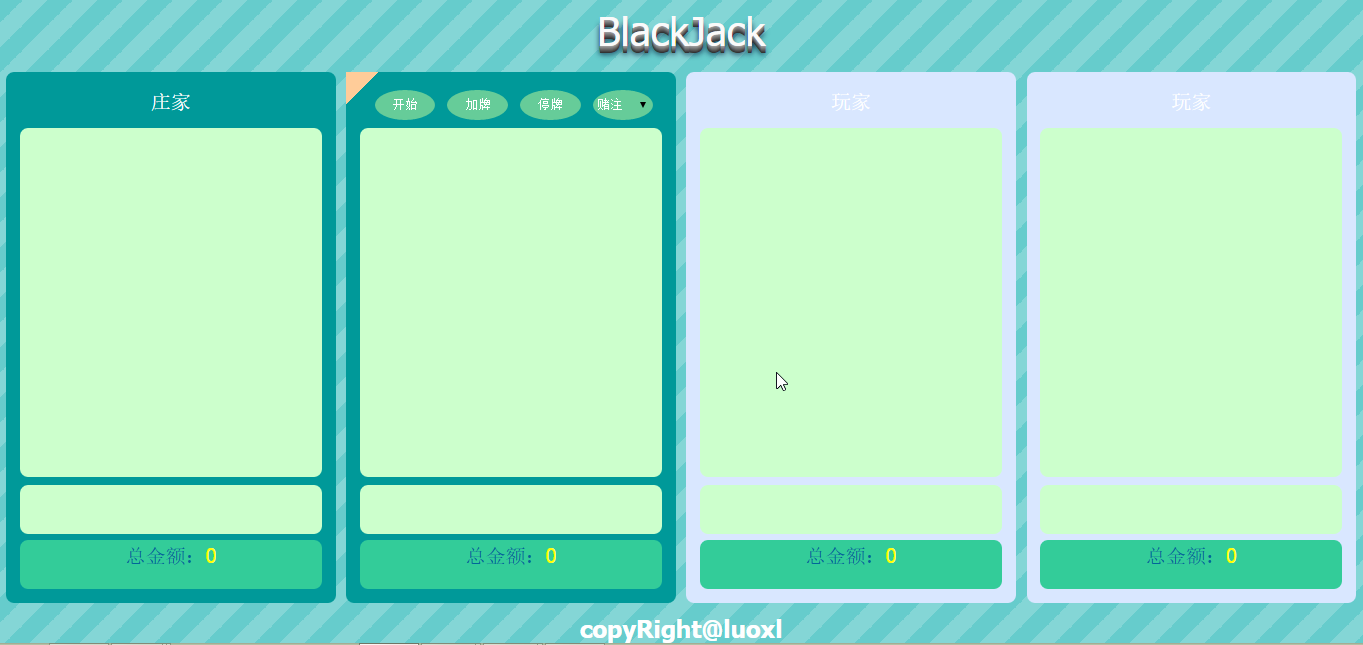


当选择4个玩家时页面如下：



背景颜色为浅色的表名当前模块没有被选择，左上角有个橘黄色小标签表示这个模块是本玩家模块。（默认第一个进入的是庄家，第一个模块为庄家模块）

再输入localhost:8000，点击第二块，则默认选择该模块。也可以选择3或4模块。



当用户点击已经有玩家占用的位置时，就会报如下提示：



位子选好以后就可以开始游戏了。此游戏并不是创建了4人游戏就必须四个人玩。而是最多四个人玩，只要选定了位置的人都已经点击开始游戏，游戏就能够开始。这样设定是为了防止有玩家中途掉线导致整个游戏无法开始。

### 多个玩家同时创建房间

当游戏开始时，如果房间还未创立，同时有几个玩家都到了首页面：

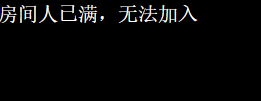


由于此游戏没有设计多个房间，因此默认第一个创建的房间有效，第二个点击确认创建后并不会创建，而是默认以玩家的身份加入第一个创建的房间。

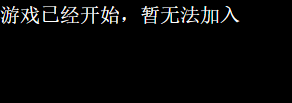


### 无法进入房间情况

当游戏没开始时，房间没满，还可以进入房间。如果房间满了提示：

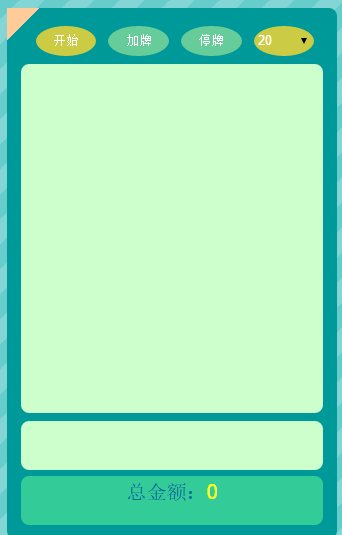


如果房间没满，但是加入游戏的人都已经点击开始游戏：

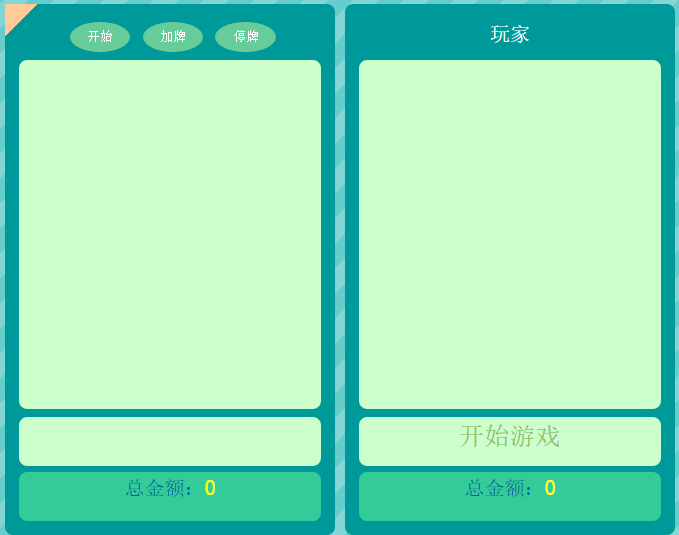


## 玩游戏

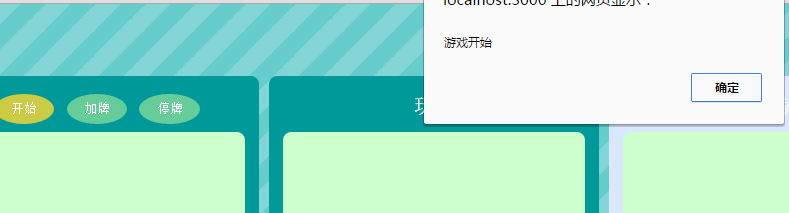
玩家下注后就可以等待游戏开始：



当一个玩家点击开始后，其他玩家会受到动画提示：



所有玩家都点击开始后，提示游戏开始，动画结束。



开始游戏后，随机生成两张牌，其他玩家无法看到该玩家的第一张牌，也叫作暗牌：



点击加牌后，本玩家和其他玩家都会收到该玩家加的牌：



点击停牌后，其他玩家动态提示该玩家已停牌。所有玩家都停牌后计算分数：





## 离开游戏

### 普通玩家离开游戏

其他玩家界面该玩家位置变灰，表示已离开。

### 庄家离开游戏

庄家离开游戏，表示此游戏倒桩了，没有庄家，玩家的牌无法对比，因此整个房间解散。玩家界面显示：



# 可改进点

**程序健壮性和易用性方面：**

* + 由于js没有使用框架，可能某些DOM操作不兼容所有浏览器；
  + 可以设计多个房间；
  + 可以设计交换玩家和庄家身份；

**界面显示方面：**

* 利用css3特性制作的动画不兼容所有浏览器；
* 没有实现手机上可以玩；
* 可以实现游戏时桌面和扑克的3D视觉效果，本来用perspective和transform实现了桌面效果，但排版看起来很丑。