Draw&Guss

* 切换颜色
* 词库
* 创建房间
* 怎么传递？

1. 设置界面
   * 选择词库
   * 设置最大人数
   * 选择作画时间

## 接口：

由于此次课设使用的技术为websocket技术，因此有别于传统接口，不用考虑浏览器跨域等问题。

**游戏逻辑接口：**

* startGame： 双端收发，接收roomID与userID，返回房间成员信息、房间设置、行为栈
* submitDraw：前端发送，接收userID,roomID,base64（作画图片的base64格式）,gameStep（当前游戏步骤）,memberIndex（房间成员序号）,username, 返回前端当前已完成作画人数、总人数、actionStack（行为栈）
* submitWord：前端发送，接收userID,roomID,gussWord（玩家猜测的词汇）,gameStep（当前游戏步骤）,memberIndex（房间成员序号）,username, 返回前端当前已完成作画人数、总人数、actionStack（行为栈）
* turnToGuss：后端发送，用来告知前端房间成员准备完毕，转化为猜测模式，无返回。
* turnToDraw：后端发送，用来告知前端房间成员准备完毕，转化为作画模式，无返回。
* turnToReview：后端发送，当游戏轮数等于房间成员数则告知前端转化为游戏复盘模式。
* forbidStartGame：后端发送，当用户通过修改浏览器sessionStorage获得房主权限来开始游戏时，后端识别用户非法操作，并禁止用户行为。

**房间逻辑接口：**

* createRoom：前端发送，告知后端房主创建房间，接收userID，用户名，返回roomID ，roomSettings
* roomMemberHasChanged：后端发送，当有用户加入或退出房间，告知前端当前房间人数情况，返回memberNums(当前房间人数)，sumNums(房间总人数)。
* updateRoom：后端发送，接收roomID，返回房间人员信息，用来更新房间状态。
* closeRoom：双端收发，接收roomID，告知所有用户房间已关闭。
* joinRoom：前端发送，接收userID，roomID触发updateRoom，roomMemberHasChanged，无返回。
* exitRoom：前端发送，接收userID，roomID触发updateRoom，roomMemberHasChanged，无返回。
* prepare：前端发送，接收userID，roomID，告知后端成员已准备，返回房间成员状态，并触发updateRoom
* cancelPrepare：前端发送，接收userID，roomID，告知后端成员取消准备，返回房间成员状态，并触发updateRoom
* saveSettings：前端发送，保存设置，接收userID，roomID，roomSettings，告知后端设置已修改，用来触发settingsHasChanged，无返回。
* settingsHasChanged：后端发送，告知前端设置已修改成功，返回前端roomSettings。

**主逻辑接口：**

* connection：双端收发，有用户连接， 返回唯一userID。
* disconnect：双端收发，用户断开连接，返回断开连接用户userID。

## 存储设计：

由于本作品没有永久或长期记录数据信息的需求，因此用户数据短暂存放在前端浏览器sessionStorage中。

**创建房间前：**

* userID 用户唯一标识
* username 用户昵称

**创建房间（加入房间）后：**

* userID 用户唯一标识
* username 用户昵称
* roomSettings 房间设置
* userStatus 用户身份（0普通用户，1房主）
* roomID 房间唯一标识
* peopleCounting 房间人数信息
* roomInfo 房间成员信息

**进入游戏后：**

* gameStep 游戏轮数
* gameStatus 游戏状态（DRAW作画魔术、GUSS猜测模式、WAITING等待模式）
* memberIndex 玩家序号
* actionStack 游戏行为栈，用来记录每个用户作画过程，用来结尾复盘
* time 当前倒计时时间，防止用户刷新时间重置