上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

计算机网络课程项目

项目名称： 基于 B/S 架构实现的多人实时共享画板

小组成员： 苏勇文、穆凡、杨涵章

目录

一、 项目概况

B/S 架构即浏览器和服务器架构模式，是随着 Internet 技术的兴起，对 C/S 架构的一种变化或者改进的架构。在这种架构下，用户工作界面是通过 WWW 浏览器来实现，极少部分事务逻辑在前端（Browser）实现，但是主要事务逻辑在服务器端(Server)实现。由于多人共享画板对便携性和跨平台性有着较高的要求，如果使用传统的 C/S 架构进行开发，便会影响用户的使用体验。因此本项目使用 B/S 架构进行开发，使得用户能够通过简单的网络登入便能使用高效便捷的共享画板功能。

本项目基于 html5, JavaScript, Css 实现网页画板，基于 php, WebSocket 协议, Workerman 框架实现前后端数据通信和权限控制，基于 http 协议, java 实现画板网页的传输，最终实现了共享画板。最终画板由权限画板、普通画板、后端服务器组成。

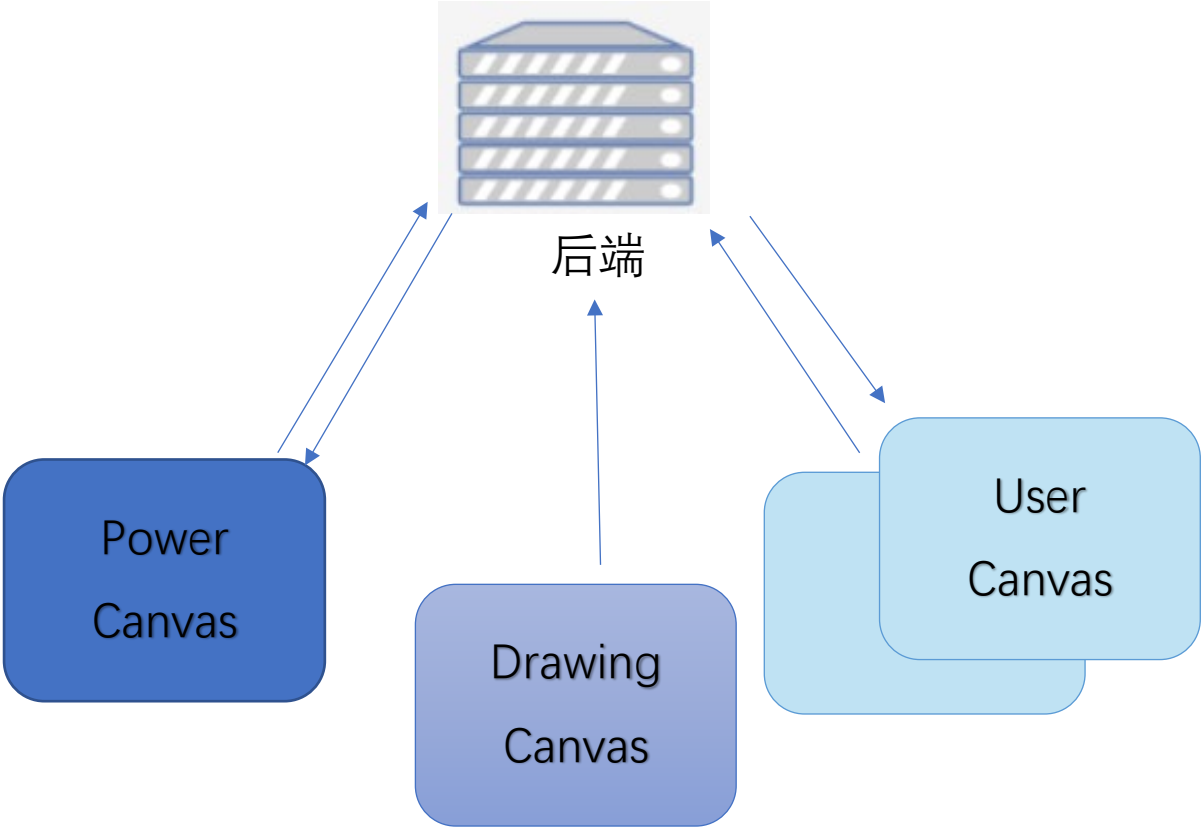

图 1：画板模型

权限画板：始终具有绘画权限，能够处理普通画板的绘画权限请求；
普通画板：需要自行请求画板权限，拥有画板权限之后即可进行绘画；
后端服务器：负责控制画板之间的数据同步和权限控制。

为了进行高效的开发，本组将开发任务划分为画板实现、数据通信实现、组件连接三部分。

1. 画板

随着浏览器版本的更新换代，功能丰富，应用广泛的 Html5 技术已经被大多数主流浏览器所支持。Html5 技术是 html, css, javasript 多项技术的组合，这些技术使得其拥有快速处理不同多媒体信息的能力。由于其功能强大，设计简便，特别是其支持多设备，跨平台的性能，本小组选用 html5 技术来设计网页画板，以满足不同型号设备的需要。

canvas 是 html5 中新增的一个用于定义图形的标签，作为 Html5 中已经定义好的图形容器，它本身没有行为，但提供了绘图 API 给予客户端 JavaScript，只需对其增加相关脚本即可满足网页画板的对应功能。通过 canvas 提供的 API 编写脚本，可以实现在网页实时生成图像。

2. 数据通信

为了实现各个画板之间的数据共享，本组成员决定使用 WebSocket 作为数据传输的协议。相较于传统的 Ajax 轮询，WebSocket 协议使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在 WebSocket API 中，浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。这样的特性使得画板之间的数据共享更加便捷。

使用 WebSocket 进行数据传输同样存在一定的问题，即在建立连接之后，客户端和服务端之间的数据传输需要自行约定数据协议，再由客户端和服务端进行数据的实别和处理。因此基于 WebSocket 协议，本组选用 Workerman 作为框架，实现共享画板。

3. http 服务器

本项目是依赖 html 文件接入后端而实现的跨平台画板，为了实现客户端 html 的快速分发，决定搭建一个 http 服务器负责传输用户界面的 html 文件，http 服务器主要与客户端使用 socket 通信，解析请求头然后发送答复报文。该 http 服务器启动后开始监听 8080 端口，当同一个局域网的浏览器访问本主机的 8080 端口时将会启动连接，然后返回所需要的文件。

启动程序后将会在 UI 界面显示当前 http 服务器的 IP 地址和监听端口号，只要客户端开启浏览器输入网址即可接入画板。接入后客户端将于服务端在局域网内进行通信。

二、 项目流程

1. 数据通信的设计与实现

1.1 通信流程介绍

本项目需要在多个客户端之间实现画板数据的传输与共享，并进行绘画权限的控制。但由于 WebSocket 协议无法进行数据类别的区分，因此需要自行设计数据报文的识别方式。以下将通信流程拆分为连接登录、权限申请、绘画数据同步等步骤分别进行阐述。

1.1.1 连接登录

由于本项目使用 WebSocket 协议进行数据的传输，因此首先需要启动后端，在网页登录之后需要与后端进行连接，之后才能进行各个画板之间的数据同步

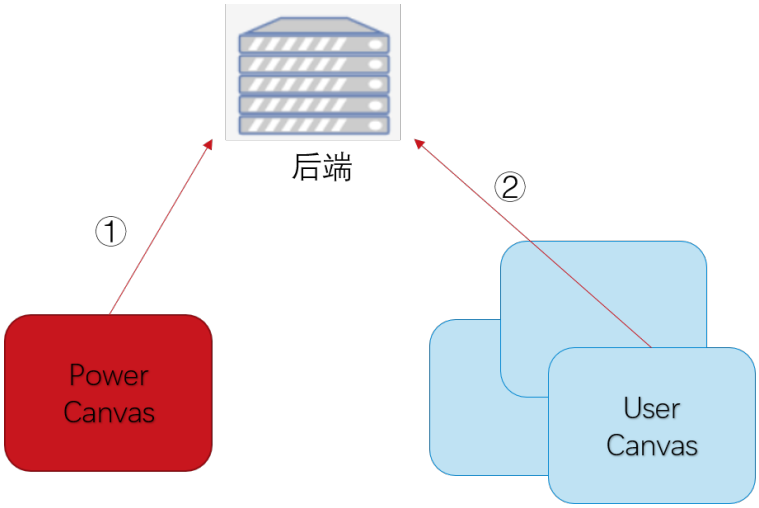

图 2：连接登录示意图

由于权限画板需要对绘画权限进行选择，因此需要首先进行权限画板的连接保证权限控制流程的正常。图中步骤①为权限画板进行 WebSocket 连接。权限画板连接之后普通画板便可以进行步骤②，打开网页登陆连接，每当一个普通画板连接，后端便会发送当前已有数据，将数据实时同步到每一个已经连接的画板。

1.1.2 权限申请

为了保证绘画过程中不存在多人同时作画造成作画混乱或者出现其余意料之外的错误，因此将该共享画板设计为同一时间仅能由一人作画，作画权限的控制由权限用户进行，这样既保证了绘画流程的正确性，也使得开发流程简单了许多。

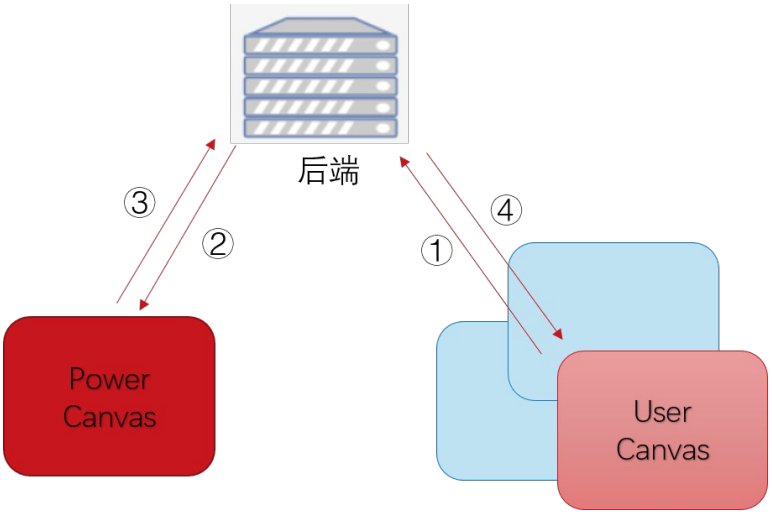

图 3：权限申请示意图

图中步骤①为普通用户当前触发请求绘画事件，向后端发送绘画权限申请，步骤②为后端将普通用户的申请转发给管理用户，并将当前所有用户的权限申请情况展示给管理者。这时需要管理者进行权限的分配选择，之后通过步骤③将权限获得信息反馈给后端。最后经过步骤④后端将权限获取情况反馈回普通用户，再由普通用户端根据权限分配情况决定当前是否能够进行绘画。

1.1.3 绘画数据同步

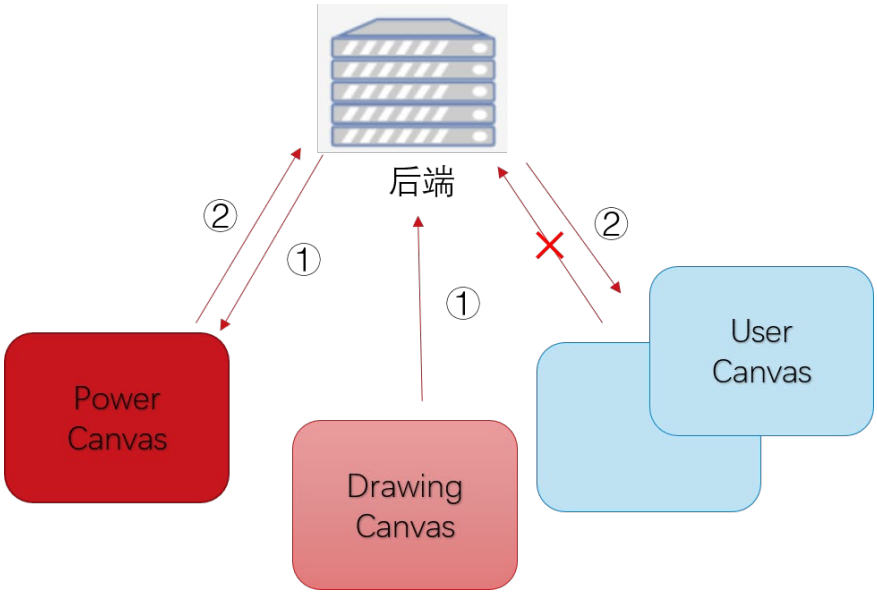

图 4：绘画数据同步示意图

绘画数据的同步主要在居有绘画权限的用户和普通用户之间的同步，步骤①为拥有绘画权限的用户/管理用户每进行一次单位绘画行为便会向后端传输绘画数据（可以看到此时不具有绘画权限的用户无法绘画并传输数据）。之后在后端进行一定的数据处理，将数据经由步骤②对数据进行储存，并转发到所有用户。

1.2 数据协议设计

由于 WebSocket 无法进行多种类型数据的实别，因此本组成员自行设计数据协议，主要包括普通用户和后端，管理用户和后端之间的数据协议设计。

1.2.1 普通用户发往后端

	data head	data	when to send
普通用户	"account:"	用户账号	登录时
	"request:"	可为空	点击请求
	"data:"	画板数据	画板更新
	"close!"	空	该用户关闭网页、断开连接

表 1：普通用户发送数据设计

1.2.2 管理用户发往后端

	data head	data	when to send
管理用户	"account:"	超级用户账号	登录
	"data:"	画板数据	画板更新
	“response_user:”	获得权限的用户账号	点击决定权限用户
	"close!"	空	超级用户关闭

表 2：管理用户发送数据设计

1.2.3 后端发往普通用户

	data head	data	when to send
后端	"data:"	画板数据	
	"response:"	1 为获得权限；0 为未获得权限	后端对请求权限用户的回应

表 3：后端发往普通用户数据设计

1.2.4 后端发往管理用户

	data head	data	when to send
管理用户控制	"ask_user:"	当前请求权限的用户账号	后端向权限用户发送请求权限的账号
	"data:"	画板数据	
	"connect_user:"	当前接入的用户账号	后端向权限用户发送接入账号
	“disconnect_user:”	当前断开的用户账号	后端向权限用户发送断开账号

表 4：后端发往管理用户数据设计

2. 画板的设计与实现

2.1 画板功能介绍

本组利用 html5 编写网页画板，除了满足基本的绘制涂鸦的功能外，还增加了橡皮擦，画笔颜色调节和清除功能，同时其还能在移动端和 PC 端均可使用，这些功能都能通过在 canvas 提供的 API 中编写来实现。

涂鸦功能使得用户在画布上按下鼠标后绘制一条跟随鼠标箭头的线段并且在放开鼠标后停止绘画；橡皮擦功能则可以使用户点击橡皮擦按钮后通过按下鼠标并移动其位置来清除之前画下的痕迹；颜色调节功能可以改变画笔的线条颜色；清除功能可以使用户在按下清除按钮后清空整个界面。

2.2 画板主体设计

canvas 标签提供了可供 js 编辑的 API，从而进行 2D 图像的绘制。在 JavaScript 中通过调用 canvas 提供的 getElementById 函数即可使用 canvas 功能，然后通过 getContent 函数获得 ctx 对象，用于绘制路径、矩形、圆形、字符以及添加图像等。

当用户使用鼠标点击或者触摸屏幕时将触发 Html5 的一个“事件”(event)，在 html5 中这些事件被分为了不同类型，以鼠标触碰事件为例，其有如下类型：

事件类型	说明
Onmousedown	与 ontouchstart 事件基本相同，表示开始点击/触碰
Onmousemove	与 ontouchmove 事件基本相同，表示移动
onmouseup	与 ontouchup 事件基本相同，表示点击/触碰结束

表 5：事件属性

这些事件被触发后可以将其与 JavaScript 函数相关联，通过对这些事件的定义，即可完成对画笔功能的实现。

如下图 Fig 1 所示，当收到用户触发的事件后，首先对事件类型进行判断，然后根据事件类型定义所触发的处理函数即可完成画板功能。

其伪代码如下：

Algorithm 1 Board Main Algorithm

Input: User interact event e
Output: draw line

```
judge the type of e;
if isTouch then
    do ontouchstart: move to start position; send data;
    do ontouchmove: draw line; send data;
    do ontouchup: end drawing;
else
    do onmousedown: move to start position; send data;
    do onmousemove: draw line; send data;
    do onmouseup: end drawing;
end if
```

算法 1:画板主要流程

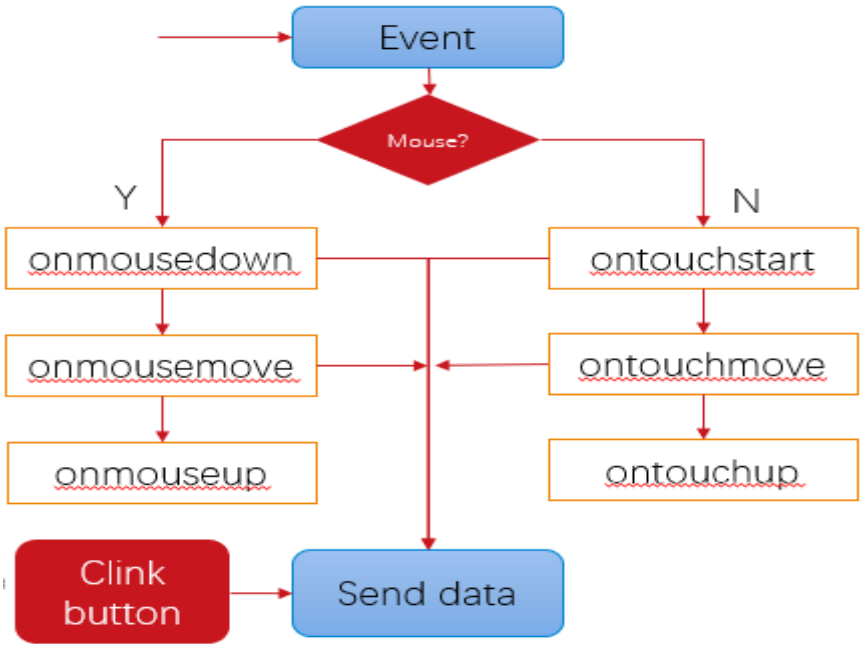

图 5：画板流程

2.3 后端交互功能

WebSocket 协议提供较为完善的 API，使得前端定义 websocket 之后可以使用其函数 websocket.onopen, websocket.onmessage 和 websocket.onclose 与后端进行交互，其中

websocket.onmessage 可以接收到服务器发来的数据，而 websocket.send 则可以将数据发送到后端。因此通过这两个函数即可实现对后端数据的交互。

当现在画板数据更新时将点位信息发向后端，并接受后端信息并将其绘制在画板上，即实现了在线画板的同步。

2.3.1 传输数据的设计

通过自定义的 sendPoint 函数向后端传输数据，当点位为线段起点时 type 设为 1，当点位为线段终点时 type=2。传输画板的长宽则是为了解决不同设备画板大小不一时传输图像的适配问题。其中“data”串是为了便于后端分辨前端法来的数据类型。

因此设计传输数据结构为:Points(x,y,type,height,width); 其中(x,y)为当前鼠标所在坐标，type 表示传输数据的种类，根据绘画需求，本组供设计了五种种类，将在后面几节详细介绍。数据中 height 以及 width 数据传输的是当前画板的长和宽，通过它们即可实现不同屏幕大小的画板信息的统一和缩放。

2.3.2 数据传输位置

根据上文所述，对画板进行操作主要在两个部分进行，一是在点击/触碰事件开始时，要采用 MoveTo 函数将画笔移动至当前位置；二是在点击/触碰移动时要实时绘出当前曲线，因此需要在画出曲线时，传输相应数据。另外为了增加画板的功能，通过设计按钮触发函数，此时也应该将画板数据传输以便其他在线画板同步数据。

2.3.3 数据传输类型

根据传输数据的 type 值，将其分为五种类型的画板数据，由下表所示，不同类型的画板数据记录不同的数据内容并且在接收这些数据时根据 type 类型判断处理。

Type	x	y	属性	
1	x	y	线条起点位置	
2	x	Y	线条终点位置	
3	x	Y	表示选择橡皮擦并传输橡皮位置	
4	0	0	表示选择清空	
5	\	\	Color=current color	表示调色，并传输颜色信息

表 6：传输数据类型

2.3.4 接收数据并同步方式

由上述对 websocket 的介绍可知，websocket.onmessage 可获得后端广播而来的数据，在判断数据类型后即可对其进行处理。如图所示，获得后端发送的数据后，首先对数据本身的类型进行判断，如果为画板数据则选择画板数据类再进行处理。

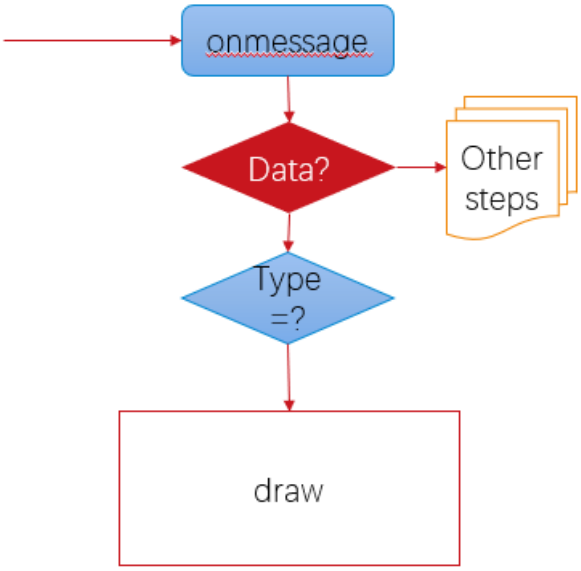

图 6：接收流程

获得画板数据的 type 类型后即可根据下表对其进行处理。对 type=1 的数据进行 ctx.moveto 处理，对 type=2 的数据进行 ctx.lineto 处理。可以增加 type 数量以增加画板的功能。通过下表对不同类型数据进行处理。

Type	处理方式
1	Ctx.moveTo 移动到画笔起点
2	Ctx.lineTo 完成划线
3	Ctx.clearRect 清除对应区域
4	清空整个画笔
5	设置画笔颜色

表 7：数据处理方式

2.4 多端处理

在 PC 端和移动端用户绘制画板使用的工具不同。在 PC 端，用户使用鼠标的按键，移动触发 onmouse 事件，从而完成对线段的绘制；在移动端，这些操作则被手指触摸点击造成的 ontouch 事件替代。这两种触摸方式决定了对于不同设备的事件应该单独设计其事件处理函数，并且画板需要能够判断当前设备类型以调用相应函数。因此采用如下方式进行判断：

Algorithm 2 Event type judge Algorithm

Input: Users' interacting evet e

Output: Event type *canvas*

```
if onTouchstart is undined then
    do mouse event;
    canvas=mouse;
else
    do touch event;
    canvas=touch;
end if
```

算法 2：:事件判断算法

该算法函数通过判断 ontouchstart 事件是否触发来选择是否使用鼠标触发函数，或者选择点击触发函数。

点击触发函数与鼠标触发函数在结构上完全相同，其中 ontouchstart 对应 Onmousedown 事件，表示手指开始点击屏幕；ontouchmove 对应 onmousemove;ontouchup 对应 onmouseup 表示此次手指点击事件结束，停止绘画。

2.5 其余功能实现

2.5.1 橡皮擦功能

canvas 画板提供了 ctx.clearRect(x,y,a,b)函数来完成对画板的擦除工作，其中 x,y 表示要擦除位置的中心点，a,b 分别对应了擦除区域的长宽大小。因此在事件中调用该函数以替代划线函数 ctx.lineTo 即可实现擦除功能。在该画板中，本组设计了两个按钮来供用户选择画笔和橡皮功能，按下按钮时将 isPencil 值进行改变以便在事件中使用条件语句进行控制。

其算法流程如下所示，即在上述每个划线的地方进行判断：若 isPencil 为真，代表采用划线函数，反之则采用清除函数。

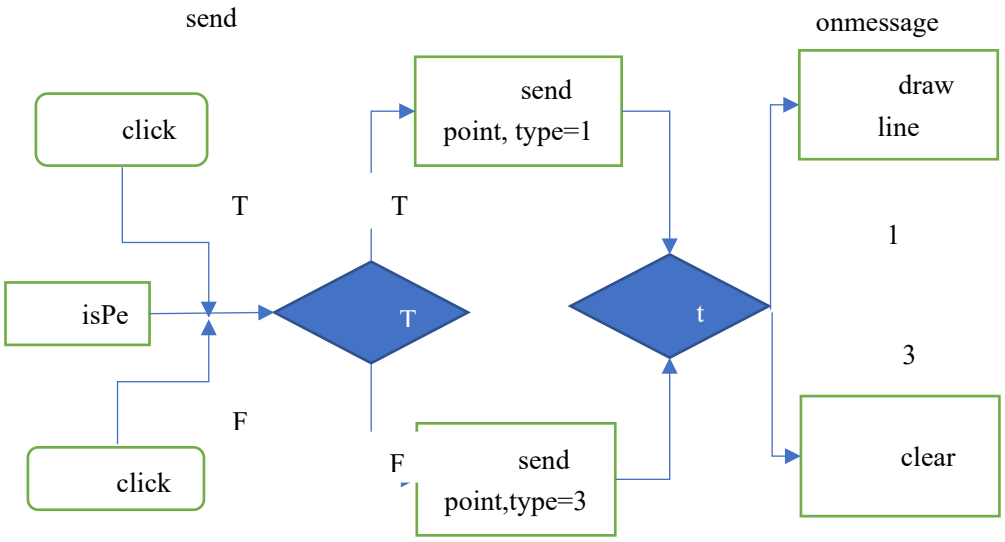

图 7：橡皮功能流程图

由该流程图可知，isPencil 的值决定了是否使用橡皮功能代码，最终在传输数据时其传输的 type 将改变，使得接受数据时能分别发送端使用的数据类型。

2.5.2 调色器功能与清空功能

调色器功能的实现与橡皮功能的实现大体上相同，通过设置调色按钮后，每当用户选择颜色即可通过 ctx.strokeStyle 函数设置当前画笔颜色，然后将该数据使用 websocket.send 发出即可完成同步调色功能。

清空功能与调色器功能的实现结构完全相同，只需在按下清空画板“clear”按钮后使用 ctx.clearRect(0,0,画板长，画板宽)即可实现全屏清空，然后发送特殊 type=4 的数据即可实现同步。

3. 组件连接的设计与实现

3.1 http 服务器详解

超文本传输协议（HTTP）是一个用于传输超媒体文档（例如 HTML）的应用层协议。它是为 Web 浏览器与 Web 服务器之间的通信而设计的，但也可以用于其他目的。HTTP 遵循经典的客户端-服务端模型，客户端打开一个连接以发出请求，然后等待直到收到服务器端响应。在本项目中，浏览器作为客户端发出 http 请求，需要我们自行搭建一个 http 服务器响应客户端的请求，传输画板所需要的 html 文件。

3.1.1 http 服务器搭建环境

在 java 环境中搭建 http 服务器，使用 eclipse IDE，创建项目后导入需要的依赖包，本项目采用较为基础的 socket API，将从头实现一个服务器。

套接字使用 TCP 提供了两台计算机之间的通信机制。 客户端程序创建一个套接字，并尝试连接服务器的套接字。

当连接建立时，服务器会创建一个 Socket 对象。客户端和服务器现在可以通过对 Socket 对象的写入和读取来进行通信。

java.net.Socket 类代表一个套接字，并且 java.net.ServerSocket 类为服务器程序提供了一种来监听客户端，并与他们建立连接的机制。

以下步骤在两台计算机之间使用套接字建立 TCP 连接时会出现：

服务器实例化一个 ServerSocket 对象，表示通过服务器上的端口通信。

服务器调用 ServerSocket 类的 accept() 方法，该方法将一直等待，直到客户端连接到服务器上给定的端口。

服务器正在等待时，一个客户端实例化一个 Socket 对象，指定服务器名称和端口号来请求连接。

Socket 类的构造函数试图将客户端连接到指定的服务器和端口号。如果通信被建立，则在客户端创建一个 Socket 对象能够与服务器进行通信。

在服务器端，accept() 方法返回服务器上一个新的 socket 引用，该 socket 连接到客户端的 socket。

图 8：http 服务器流程

3.1.2 实现 Runnable 接口

使用 Runnable 接口

public class App implements Runnable

在 **public void run()**这个函数中实现整个服务器的主线程

1. 首先创建客户端 socket 并监听客户端的请求调用 accept()会自动开始监听服务器端口，如果收到请求将会自动连接。
2. 客户端和服务器建立连接之后，需要读取客户端发出的 http 请求内容来确定传输的文件
3. http 服务器根据客户端请求的方式和需要的文件传输给浏览器。

在本项目的设计中只需要实现对 GET 请求的回应即可。如果浏览器用 GET 请求空则返回用户界面，如果请求不为空则截取请求内容，然后将所请求的内容发送回浏览器。设计了处理 html 文件和 js 文件的逻辑。

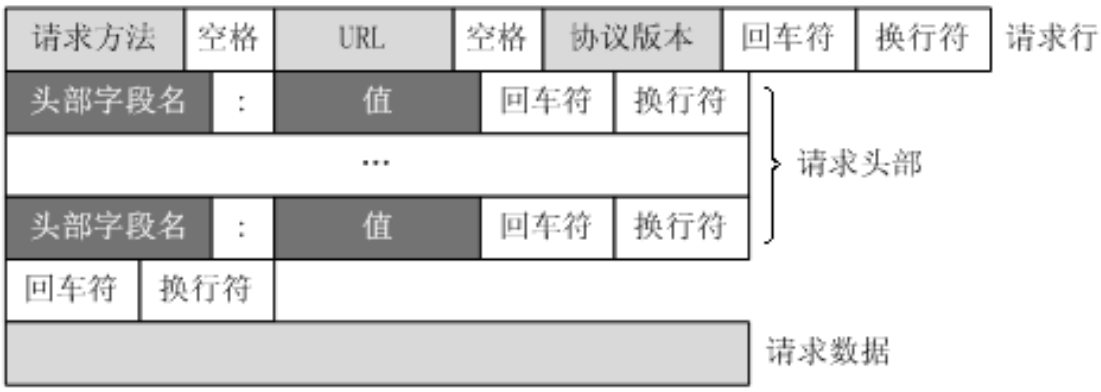

图 9：http 请求格式

其中调用了传输文件的函数 fileService()，这个函数负责将参数名对应的文件根据 ConTentType 发送到 client，需要注意的是 http 应答报文的格式。

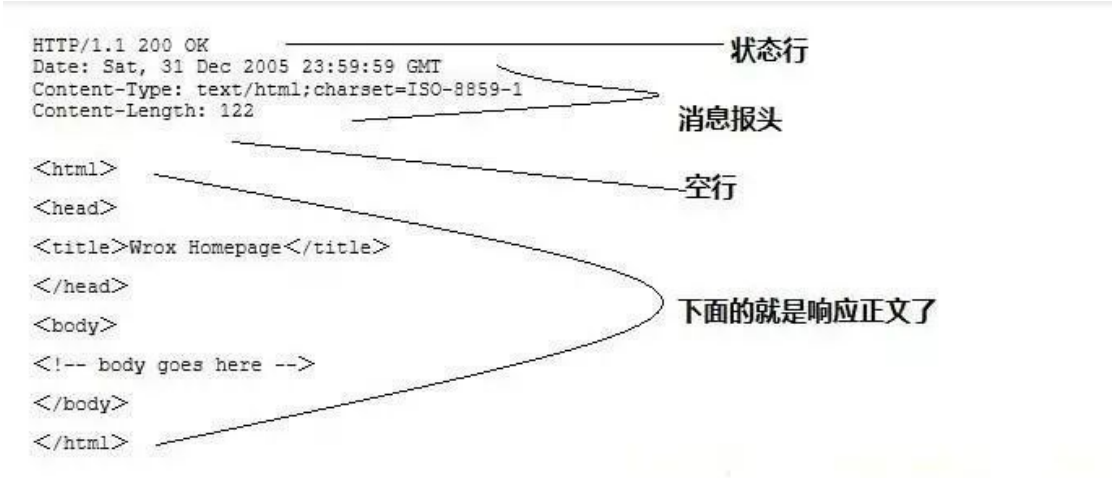

图 10：http 应答报文

按照正确的应答格式发送 http 应答报文，浏览器就可以正确处理，进而解析 html 文件。

3.2 UI 界面设计

调用 java 原生界面库 Swing 和 Awt 设计 UI 界面。

AWT（Abstract Window Toolkit，抽象窗口工具）是一套早期的 Java GUI 开发工具，Swing 也是在 AWT 的基础上发展起来的。

Swing 是一个用于 Java GUI 编程（图形界面设计）的工具包（类库）；换句话说，Java 可以用来开发带界面的 PC 软件，使用到的工具就是 Swing。

Swing 使用纯粹的 Java 代码来模拟各种控件（使用 Java 自带的作图函数绘制出各种控件），没有使用本地操作系统的内在方法，所以 Swing 是跨平台的。也正是因为 Swing 的这种特性，人们通常把 Swing 控件称为轻量级控件。

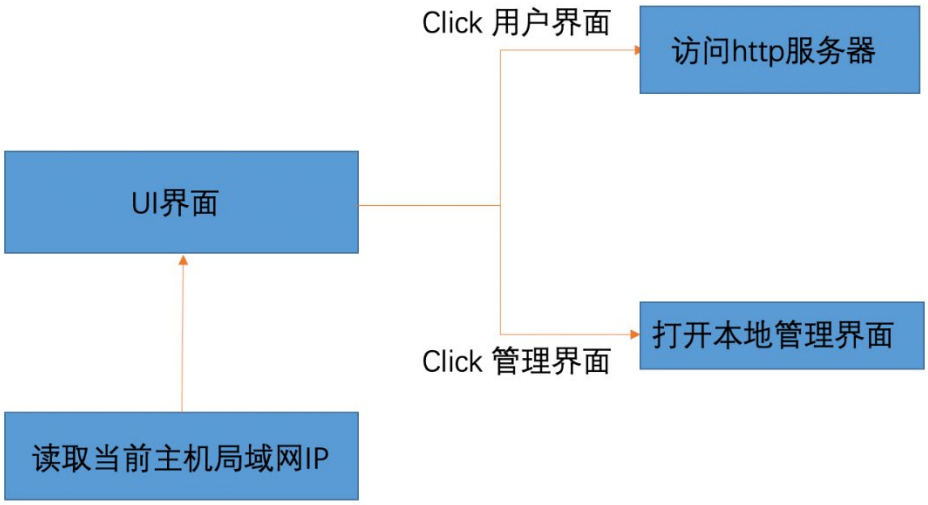

图 11：UI 设计流程

调用的组件解析如下：

概念解析：

JFrame – java 的 GUI 程序的基本思路是以 JFrame 为基础，它是屏幕上 window 的对象，能够最大化、最小化、关闭。

JPanel – Java 图形用户界面(GUI)工具包 swing 中的面板容器类，包含在 javax.swing 包中，可以进行嵌套，功能是对窗体中具有相同逻辑功能的组件进行组合，是一种轻量级容器，可以加入到 JFrame 窗体中。。

JLabel – JLabel 对象可以显示文本、图像或同时显示二者。可以通过设置垂直和水平对齐方式，指定标签显示区中标签内容在何处对齐。默认情况下，标签在其显示区内垂直居中对齐。默认情况下，只显示文本的标签是开始边对齐；而只显示图像的标签则水平居中对齐。

JTextField –一个轻量级组件，它允许编辑单行文本。

JButton – JButton 类的实例。用于创建按钮类似实例中的 "Login"。

创建一个 JFrame 作为主要窗口，创建两个 JButton，一个负责提供用户界面，另一个负责提供管理界面，这两个都要设置事件监听函数来控制点击之后跳转。

事件处理模型：

若使图形界面能够接收用户的操作，必须给各个组件加上事件处理机制。在事件处理的过程中，主要涉及三类对象。

Event（事件）：用户对组件的一次操作称为一个事件，以类的形式出现。例如，键盘操作对应的事件类是 KeyEvent。

Event Source（事件源）：事件发生的场所，通常就是各个组件，例如按钮 Button。

Event Handler（事件处理者）：接收事件对象并对其进行处理的对象事件处理器，通常就是某个 Java 类中负责处理事件的成员方法。

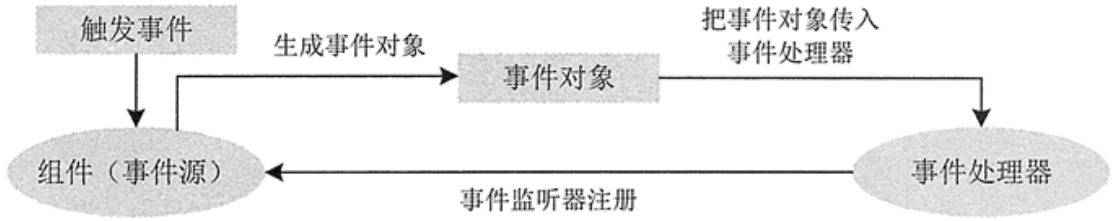

图 12：事件处理模型

动作事件监听器：

动作事件监听器是 Swing 中比较常用的事件监听器，很多组件的动作都会使用它监听，像按钮被里击、列表框中选择一项等。与动作事件监听器有关的信息如下。

事件名称：ActionEvent。

事件监听接口: ActionListener。

事件相关方法：addActionListener() 添加监听，removeActionListener() 删除监听。

涉及事件源：JButton、JList、JTextField 等。

三、 效果展示

1. 后端服务器运行

由于后端服务器使用 Workereman 框架进行实现，因此需要提前安装好 php 并配置好环境变量，这里直接展示运行步骤。

进入后端代码文件目录中，使用指令 `php ./server.php start` 启动服务器。

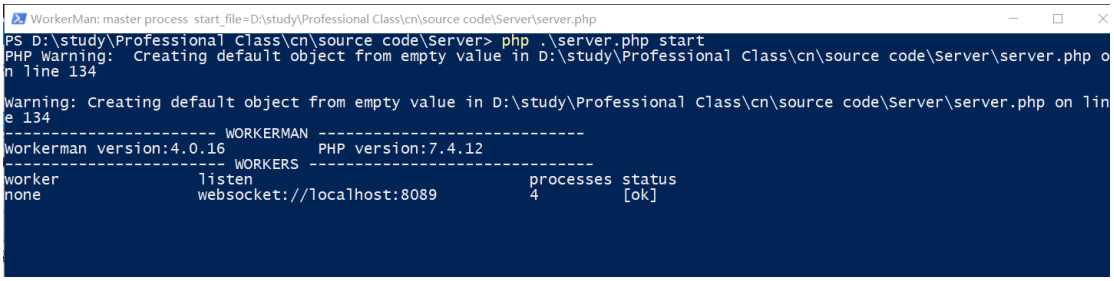

图 13：后端服务器启动

2. 用户登录

由于连接部分使用 java 实现，因此需要提前安装并配置好 java 环境，之后运行 app.har 文件即可，运行效果如下：

图 14：管理界面

通过点击管理界面按钮可以进入管理画板界面，通过点击用户界面按钮，可以进入普通用户界面，同时还提供二维码和 IP 两种方式供其余用户获取用户界面（此时 IP 为当前测试环境 IP，会随网络环境变动而改变）。管理用户自动登录，普通用户需要自行输入用户名进行登录。

登陆界面如下：

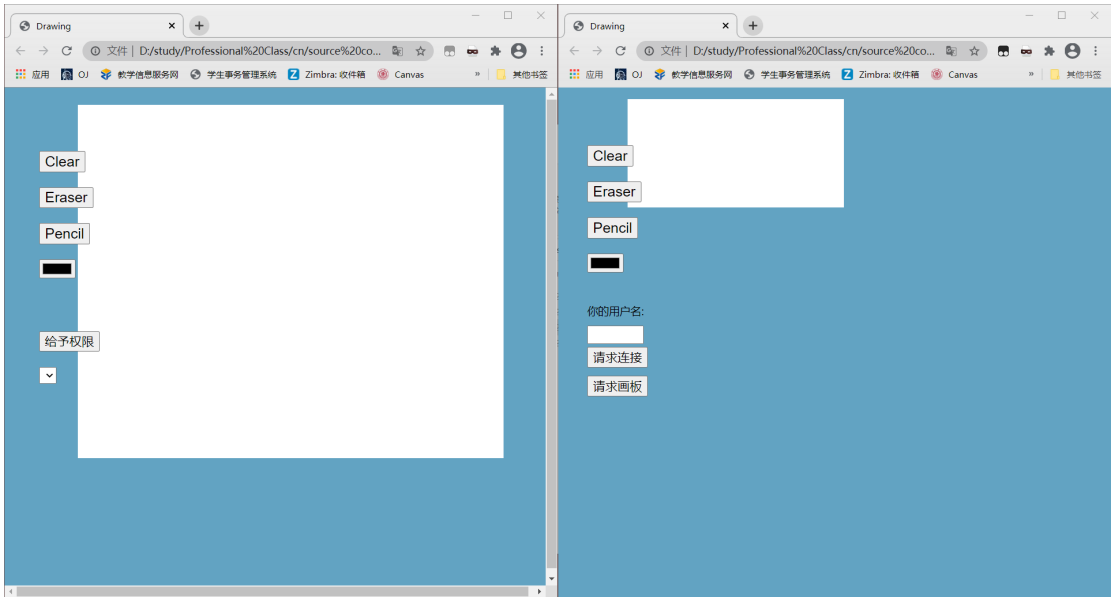

图 15：登陆界面（左为权限用户自动登录，右为普通用户输入用户名登陆）

3. 权限申请及绘画

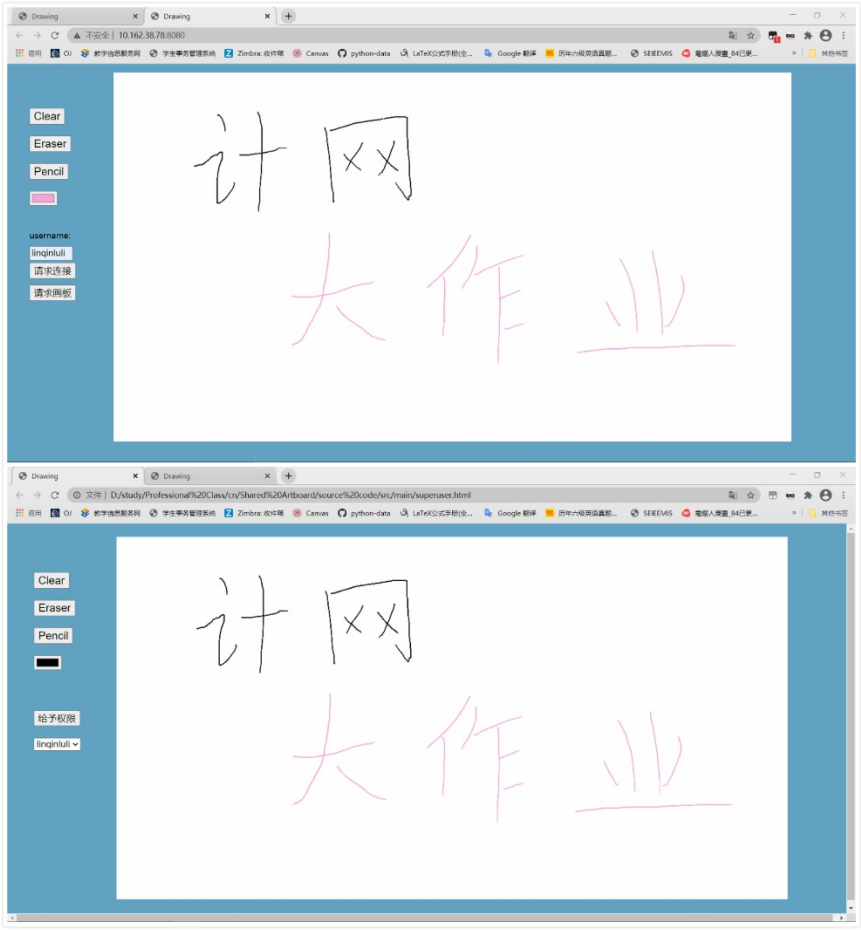

图 16：绘画界面（下为权限用户，上为普通用户）

图中普通用户登陆之后仍不能绘画，需要通过点击请求画板按钮，向权限用户提交申请，之后权限用户进行处理，将权限交给普通用户。图中“计网”二字为权限用户书写，“大作业”为获得绘画权限之后的用户书写，可以看到同步效果良好。

四、 项目亮点及创新

1. **跨平台，无需客户端软件。**由于采用 B/S 架构进行开发，相较于传统 C/S 架构，对于客户端环境要求较低，仅需要浏览器接入服务器即可，可以运行在各种操作系统上，无需下载专用的客户端软件。而现今大多数移动互联网平台均支持浏览器软件的使用，因此跨平台性能较好，即使在多种设备之间也能有着较好的表现。
2. **使用便捷。**将用户界面 html 放在管理用户的主机上并用 http 服务器传输非常便捷，不同于向固定网址寻求 html，搭建 http 服务器可以使得任何主机充当资源站，从而实现了完整性和便捷性。随时随地在任何主机都可以搭建并共享文件，同时在一个局域网中发送 http 请求大大降低了响应延时和传输延时。
3. **自行设计数据协议。**由于数据传输过程使用 WebSocket 协议，因此需要提前自行设计数据类型及其协议。虽然使得开发过程较为复杂，但在实现过程中设计数据的识别过程类似于多层协议之间的解码识别，在开发过程中能够更加深入的对数据传输过程有

了一定的了解。

4. **节省服务器资源和带宽，能够更实时地进行通讯。**很多网站为了实现推送技术，所用的技术都是 Ajax 轮询。轮询是在特定的的时间间隔（如每 1 秒），由浏览器对服务器发出 HTTP 请求，然后由服务器返回最新的数据给客户端的浏览器。这种传统的模式带来很明显的缺点，即浏览器需要不断的向服务器发出请求，然而 HTTP 请求可能包含较长的头部，其中真正有效的数据可能只是很小的一部分，显然这样会浪费很多的带宽等资源。HTML5 定义的 WebSocket 协议，能更好的节省服务器资源和带宽，并且能够更实时地进行通讯。

5. **自行设计用户权限管理。**为了方便绘画数据共享，本组成员自行设计权限管理系统，通过使用 php 开发的基于 WebSocket 协议的 Workerman 架构进行多用户管理，使得绘画数据共享流程清晰，准确无误。同时在 html 中内置刷新机制，可以随着用户的加入和退出实时更新当前请求权限用户。