目录

基于Web客户端技术的个性化UI的设计和编程 2
（Customized UI design and Programming based on Web client technology） 2
1. 前言 2
1.1毕设任务分析 3
1.2 研学计划 3
1.3研究方法 3
2. 技术总结和文献综述 4
2.1 Web平台和客户端技术概述 4
2.2 项目的增量式迭代开发模式 4
3. 内容设计概要 6
3.1 分析和设计 6
3.2 项目的实现和编程 6
3.3 项目的运行和测试 7
3.4 项目的代码提交和版本管理 8
4. 移动互联时代的UI开发初步——窄屏终端的响应式设计 9
5. 应用响应式设计技术开发可适配窄屏和宽屏的UI 10
6. 个性化UI设计中对鼠标交互的设计开发 11
7. 对触屏和鼠标的通用交互操作的设计开发 13
8. UI的个性化键盘交互控制的设计开发 16
9. 谈谈本项目中的高质量代码 19
10. 用gitBash工具管理项目的代码仓库和http服务器 19
10.1 Bash介绍 19
10.2 通过gitHub平台实现本项目的全球域名 20
10.3 创建一个空的远程代码仓库 20
10.4 设置本地仓库和远程代码仓库的链接 21
参考文献： 24

基于 Web 客户端技术的个性化 UI 的设计和编程

（Customized UI design and Programming based on Web client technology）

科师大元宇宙产业学院 2021 级 黄文胜

【摘要】

Web 技术以其跨操作系统平台的优势成为了广泛流行的网页前端开发手段，为了适应移动互联网时代软件项目的前端需求，本项目以 Web 客户端技术为研究学习内容，广泛查阅了技术资料与相关文献，集合了 HTML 内容建模、CSS 样式设计和 JavaScript 功能编程的基本技术和技巧。通过集成上述技术，再应用本科的相关课程的知识，实现了一个性化的用户界面的项目，该用户界面以响应式技术为支撑做到了最佳适配用户屏幕，程序可以动态适用于当前 PC 端和移动端设备；在功能上以 DOM 技术和事件驱动模式的程序为支撑实现了对鼠标、触屏、键盘的底层事件响应和流畅支持。为了便于后续项目的再次开发，项目用了工程思想管理，使用了软件工程的增强式开发模式，一共做了 6 次项目迭代开发，每次迭代都包含软件的 4 个经典开发过程（A:Analysis，D:Design，I：Implementation，T:Testing），逐步编写了 UI 应用。同时为实现项目开源，本项目使用 git 工具进行代码和开发过程日志记录，详细记录和展现了开发思路和代码优化的过程，并通过 gitbash 把项目上传到 github 上，建立自己的代码仓库，供全球开发者使用，最后将该代码仓库设置成为了 http 服务器，实现了本 UI 应用的全球便捷访问。

1. 前言

在本项目前期为更好的完成项目的搭建，我根据自身实力指定了每周计划，总共耗时四周完成将整个项目从最开始的设计，到最后的部署。第一周凭借给定的范围信息，我将项目可能设计的技术进行总结，如最基础的 html、css、javascript 技术，以及设计思想进行了整理，如采用增强式开发模式，以及模块化设计思想。第二周我开始逐步搭建最基础的 html 代码，为方便后续对项目再次开发，在代码搭建过程中，我严格遵守注释原则，以及对疑问产生到就地解决的方针，虽然本周进度因此进度较慢，但是对第一周总结后的技术学习却产生

了更加良性的影响，如对 html 的元素属性，以及 css 对 id 和 class 属性操作的区别等都进行了较为详细的比较和学习，为本人薄弱的前端知识丰富了一大波实践经验。第三周我对上周搭建的基础项目进行深层次开发，通过第一周总结出的开发思想，我利用搭建模型的思想方法完成了对鼠标和键盘反应等较为复杂的功能模块的设计于代码编写。第四周通过利用网络中对 github 详细的教程，我正式开始了对个人项目的部署。本次项目共有 6 次开发历程，同时每次开发的结果都经本人使用 git 提交到 github 中，以便后续追踪项目的开发过程。

根据个人对毕业设计项目和毕业论文的理解，分别阐述：学习计划、项目的技术路线、参考资料和研究方法。
建议参考：https://masterlijh.github.io/testHttp.html 的第一点和第二点。

1.1 毕设任务分析

本次毕设旨在与考察本学期学生对上课听讲过后，是否在课余时间对知识进行温习和实践，以及学生是否可以独立完成简单软件的设计，开发以及部署。

1.2 研学计划

第一周	第二周	第三周	第四周
前三天完成对本项目所需求的技术和开发方法进行总结	前三天使用前端基础技术 html 实现网页各模块的内容显示	前三天实现对后续功能的开发，如键盘响应和鼠标移动等功能的开发	前三天详细了解 github 网上部署网页的流程，以及相关资源的上传
后四天将个人总结过后的技术以及开发模式进行简单实践操作，进行简单掌握	后四天对前端时间的知识进行总结和温习，以及对前端技术进行深层次学习	后四天分析功能开发过程中产生的问题，以及将结果进行整理记录，同时做好对新知识的学习工作	后四天完成项目的部署，以及对项目搭建过程中对知识学习的总结，以及开发经验的总结

1.3 研究方法

本次项目我主要采用的式文献综述法进行项目的研究，对于我而言，书本文献等都是知识来源的最大途径，其中随着时代的发展，网络更是我学习的最大法宝，通过对前人留下的宝贵经验以及详细的知识点，使我本次项目搭建非常流畅。

2. 技术总结和文献综述

2.1 Web 平台和客户端技术概述

2.1.1web 平台的发展历程

Web 之父 Tim Berners-Lee 在发明 Web 的基本技术架构以后，就成立了 W3C 组织，该组织在 010 年后推出的 HTML5 国际标准，结合欧洲 ECMA 组织维护的 ECMAScript 国际标准，几乎完美缔造了全球开发者实现开发平台统一的理想，直到今天，科学家与 Web 行业也还一直在致力于完善这个伟大而光荣的理想[1]。学习 Web 标准和 Web 技术，学习编写 Web 程序和应用有关工具，最终架构一套高质量代码的跨平台运行的应用，是我的毕设项目应用的技术路线[2]。

2.1.2web 客户端技术概述

Web 应用的程序设计体系由三大语言有机组成：HTML, CSS, JavaScript。这三大语言的组合也体现了人类社会化大生产分工的智慧，可以看作用三套相对独立体系实现了对一个信息系统的描述和控制，可以总结为：HTML 用来描述结构（Structure）、CSS 用来描述外表（presentation）、Javascript 用来描述行为（Behavior）[3]；这也可以用经典的 MVC 设计模式来理解 Web 平台架构的三大基石，Model 可以理解为 HTML 标记语言建模，View 可以理解为用 CSS 语言来实现外观，Controller 则可理解为用 JavaScript 结合前面二个层次，实现了在微观和功能层面的代码控制[3]。

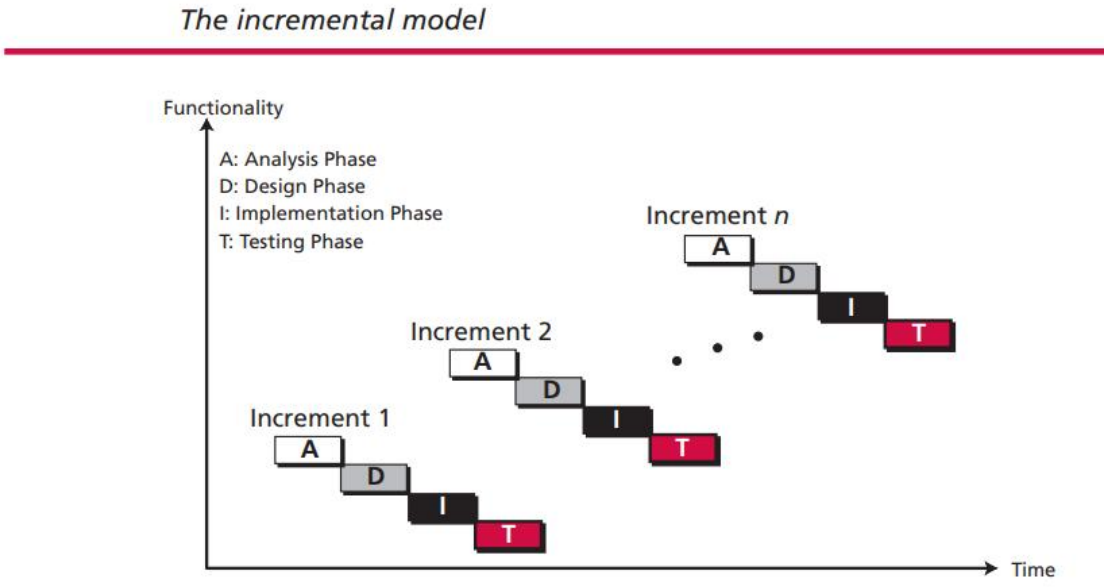
2.2 项目的增量式迭代开发模式

本项目作为一个本科专业学生毕业设计的软件作品，与单一用途的程序相比较为复杂，本项目所涉及的手写代码量远超过简单一二个数量级以上，从分析问题的到初步尝试写代码也不是能在几天内能落实的，可以说本项目是一个系统工程，因此需要从软件工程的管理视角来看待和规范项目的编写过程[4]。

而本项目考虑选择的软件工程开发过程管理模式有两种经典模型：瀑布模型（The waterfall model）和增量式迭代模型(The incremental model)。[5]而任何开发模式则都必须同样经历四个阶段：分析（Analysis）、设计（Design）、实施（Implementation）、测试（test）[6]。

瀑布模型需要专业团队完美的配合，从分析、设计到实施，最后到测试，任何阶段的开始必须基于上一阶段的完美结束。而这对于我们大多数普通开发者是不太现实的，作为小微开发者由于身兼数职，其实无法 1 次就能完美完成任何阶段的工作，比如在实施过程中，开发者会发现前面的设计存在问题，则必须在下一次

迭代项目时改良设计。在当今开源的软件开发环境中，开发者在软件的开发中总是在不断地优化设计、重构代码，持续改进程序的功能和代码质量。因此在本项目的开发中，也采用了增量模型的开发模式[5]。本项目中我一共做了六次项目的开发迭代，如下图 2-1[7]所示：



3. 内容设计概要

3.1 分析和设计

这一步是项目的初次开发，本项目最初使用人们习惯的“三段论”式简洁方式开展内容设计，首先用一个标题性信息展示 logo 或文字标题，吸引用户的注意力，迅速表达主题；然后展现主要区域，也就是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的重点；最后则是足部的附加信息，用来显示一些用户可能关心的细节变化。如图 4-1 用例图所示：

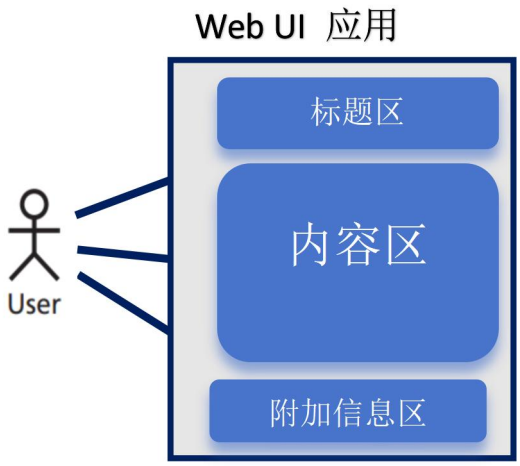


图 4-1 用例图

3.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
    《 健康的重要性 》
</header>
 <main>
     我的主题内容： 健康最重要，财富，名气再重要也比不上一个健康的身体，如果一个人的身体垮了，那么一切都使浮云
 </main>
<footer>
  CopyRight 黄先生  江西科技师范大学 2024-2025
</footer>
```

二、CSS 代码编写如下：

```
 *{
```

```
  margin: 10px;
  text-align: center;
 font-size:30px ;
}
 header{
   border: 2px solid blue;
   height: 200px;
    }

 main{
   border: 2px solid blue;
   height: 400px;
 }
 footer{
   border: 2px solid blue;
   height: 100px;
 }
a{
 display: inline-block ;
 padding:10px ;
 color: white;
 background-color: blue;
 text-decoration: none ;
}
```

3.3 项目的运行和测试

项目的运行和测试至少要通过二类终端，本文此处仅给出 PC 端用 microsoft 浏览器打开项目的结果，如下图 3.1 所示。由于本项目的阶段性文件已经上传 github 网站，移动端用户可以通过扫描图 3.2 的二维码，运行测试本项目的第一次开发的阶段性效果。

图 3.1 第一阶段页面

图 3.2 移动端二维码

3.4 项目的代码提交和版本管理

本项目的文件通过 gitBash 工具管理，作为项目的第一次迭代，在代码提交和版本管理环节，我们的目标是建立项目的基本文件结构，还有设置好代码仓库的基本信息：如开发者的名字和电子邮件。
进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ mkdir webUI
$ cd webUI
$ git init
$ git config user.name JXSD
$ git config user.email JXSD.QQ.COM
$ touch index.html myCss.css
```

编写好 index.html 和 myCss.css 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add index.html myCss.css
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
```

成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
[master (root-commit) 32de024] 项目第一版：“三段论”式的内容设计概
要开发
 2 files changed, 46 insertions(+)
 create mode 100644 index.html
 create mode 100644 myCss.css
```

项目代码仓库自此也开启了严肃的历史记录，我们可以输入日志命令查看，

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：

```
$ git log
commit 32de0243f71ec4396f54175d83c805c999d473ca (HEAD -> master)
Author: 江科师大李健宏 <marsterlijh@jxstnu.edu.cn>
Date:   Wed May 29 15:26:56 2024 +0800

    项目第一版：“三段论”式的内容设计概要开发
```

4. 移动互联时代的UI开发初步——窄屏终端的响应式设计

4.1 分析移动互联时代的多样化屏幕的需求

用 JavaScript 开动态读取显示设备的信息，然后按设计，使用 js+css 来部署适配当前设备的显示的代码。

实现代码

用汉语言来描述我们是如何实现的，与上一阶段比较，本阶段初次引入了 em 和 % ，这是 CSS 语言中比较高阶的语法，可以有效地实现我们的响应式设计 。如代码块 4-1 所示：

```
<style>
 *{
  margin: 10px;
  text-align: center;
 }

  header{
    border: 2px solid blue;
    height: 15%;
    font-size: 1.66em;

  }
  main{
    border: 2px solid blue;
    height: 70%;
    font-size: 1.2em;

  }
```

```
    nav{
      border: 2px solid blue;
      height: 10%;
         }
    nav button{
     font-size: 1.1em;
    }
    footer{
      border: 2px solid blue;
      height: 5%;
    }
  </style>
```

代码块 4-1

用汉语言来描述我们是如何实现的：与上一阶段比较，本阶段首次使用了 JavaScript ，首先创建了一个 UI 对象，然后把系统的宽度和高度记录在 UI 对象中，又计算了默认字体的大小，最后再利用动态 CSS，实现了软件界面的全屏设置。如代码块 4-2 所示：

```
<script>
    var UI = {};
    UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;
    UI.appHeight = window.innerHeight;
   const LETTERS = 22 ;
   const baseFont = UI.appWidth / LETTERS;

   //通过更改 body 对象的字体大小，这个属性能够遗传其子子孙孙
    document.body.style.fontSize = baseFont + "px";
    //通过把 body 对象的宽度和高度设置为设备/屏幕的宽度和高度，实现全屏。
    //通过 CSS 对子对象百分比（纵向）的配合，从而实现响应式设计的目标。
    document.body.style.width = UI.appWidth - 2*baseFont + "px" ;
    document.body.style.height = UI.appHeight - 4*baseFont + "px";
 </script>
```

代码块 4-2

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI

本阶段为实现宽屏和窄屏的响应式设计，我利用 JavaScript 技术先实现对设备宽度的获取，进而实现确定设备是宽屏或是窄屏，其中识别方法为： 利用 window.innerWidth
获取现在设备的宽度。关于该阶段详细代码如下：

```
var UI = {};
      // 设置宽度    如果宽度大于 600，就只显示 600 宽度
      UI.appWidth = window.innerWidth >= 600 ? 600 : window.innerWidth;
      // 设置高度
      UI.appHeight = window.innerHeight;
      // 基础字体的大小
      let baseFont = parseInt(UI.appWidth / 20);

      // 设置宽高
      setBodyStyle(baseFont, UI.appHeight, UI.appWidth);
      setAidStyle(UI.appWidth, UI.appHeight);
```

通过此段代码不仅可以在确认设备为窄屏还是宽屏的同时将网页内内容的字体大小同时实现响应式变化，控制字体大小为设备宽度的二十分之一。

6. 个性化 UI 设计中对鼠标交互的设计开发

本阶段为实现对鼠标交互设计功能，在该阶段我采用的方法为首先建立一个鼠标模型--Mouse，该模型用于捕捉鼠标在指定区域内抬起、移动，点击等状态，并将状态显示在指定区域。捕捉的方法为 addEventListener，该方法能够实现对点击等事件的监听，当事件发生时，执行后续方法。在 Mouse 模型中 isDown 元素的布尔值表示鼠标为点击还是抬起的状态。该功能实现截图如 6.1 所示。

```
// 尝试对鼠标设计 UI 控制
      var Mouse = {
         isDown: false,
         x: 0,
         deltaX: 0
      }

      $('bookface').addEventListener('mousedown', function (ev) {
         Mouse.isDown = true;
         let x = ev.pageX;
         let y = ev.pageY;
         console.log('鼠标按下了，坐标：(' + x + ',' + y + ')')
         $('bookface').textContent = '鼠标按下了，坐标：(' + x + ',' +
y + ')'
      })

      $('bookface').addEventListener('mousemove', function (ev) {
         let x = ev.pageX;
         let y = ev.pageY;
         console.log('鼠标正在移动')
```

```
        $('bookface').textContent = '鼠标正在移动，坐标：(' + x + ',' + y + ')'
    })

    $('bookface').addEventListener('mouseup', function (ev) {
        let x = ev.pageX;
        let y = ev.pageY;
        console.log('鼠标抬起了，坐标：(' + x + ',' + y + ')')
        $('bookface').textContent = '鼠标抬起了，坐标：(' + x + ',' + y + ')'
    })
```

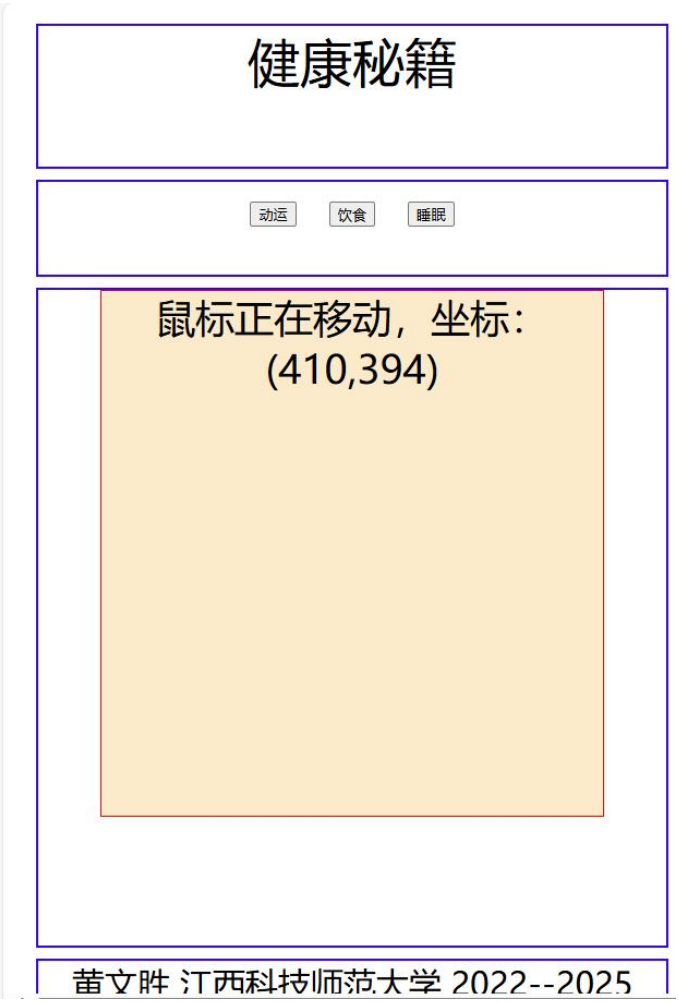


图 6.1 鼠标功能界面

下图为该阶段功能线上查看方式（二维码）：

7. 对触屏和鼠标的通用交互操作的设计开发

本阶段为实现触屏功能的开发，值得注意的是鼠标的移动与触摸时的移动在使用 JavaScript 实现时代码是可以重复使用的。为实现该功能的开发，我们可以借鉴鼠标相关功能开发的经验，首先建立触屏模型 Pointer ，该模型与鼠标模型 Mouse 相同。同时添加 touch 事件进行鼠标拖动和触摸移动行为的判断。在此次开发我们实现了对鼠标移动距离和触屏移动距离是否为有效移动的判断，以及将移动是否为有效反映在指定区域，该功能实现截图为 7.1。

```
//尝试对鼠标和触屏设计一套代码实现 UI 控制
      var Pointer = {};
      Pointer.isDown = false;
      Pointer.x = 0;
      Pointer.deltaX = 0;
      { //Code Block begin
         let handleBegin = function (ev) {
            Pointer.isDown = true;

            if (ev.touches) {
               console.log("touches1" + ev.touches);
               Pointer.x = ev.touches[0].pageX;
               Pointer.y = ev.touches[0].pageY;
               console.log("Touch begin : " + "(" + Pointer.x + "," +
Pointer.y + ")");
               $("bookface").textContent = "触屏事件开始，坐标：" + "("
+ Pointer.x + "," + Pointer.y + ")";
            } else {
```

```
                Pointer.x = ev.pageX;
                Pointer.y = ev.pageY;
                console.log("PointerDown at x: " + "(" + Pointer.x + ","
+ Pointer.y + ")");
                $("bookface").textContent = "鼠标按下，坐标：" + "(" +
Pointer.x + "," + Pointer.y + ")";
            }
        };
        let handleEnd = function (ev) {
            Pointer.isDown = false;
            ev.preventDefault()
            //console.log(ev.touches)
            if (ev.touches) {
                $("bookface").textContent = "触屏事件结束!";
                if (Math.abs(Pointer.deltaX) > 100) {
                    $("bookface").textContent += "，这是有效触屏滑动！";
                } else {
                    $("bookface").textContent += " 本次算无效触屏滑动!
";
                    $("bookface").style.left = '7%';
                }
            } else {

                $("bookface").textContent = "鼠标松开!";
                if (Math.abs(Pointer.deltaX) > 100) {
                    $("bookface").textContent += "，这是有效拖动！";
                } else {
                    $("bookface").textContent += " 本次算无效拖动！";
                    $("bookface").style.left = '7%';
                }
            }
        };
        let handleMoving = function (ev) {
            ev.preventDefault();
            if (ev.touches) {
                if (Pointer.isDown) {
                    console.log("Touch is moving");
                    Pointer.deltaX = parseInt(ev.touches[0].pageX -
Pointer.x);
                    $("bookface").textContent = "正在滑动触屏，滑动距离：
" + Pointer.deltaX + "px 。";
                    $('bookface').style.left = Pointer.deltaX + 'px';
                }
```

```
        } else {
            if (Pointer.isDown) {
                console.log("Pointer isDown and moving");
                Pointer.deltaX = parseInt(ev.pageX - Pointer.x);
                $("bookface").textContent = "正在拖动鼠标，距离：" +
Pointer.deltaX + "px 。";
                $('bookface').style.left = Pointer.deltaX + 'px';
            }
        }
    };

    $("bookface").addEventListener("mousedown", handleBegin);
    $("bookface").addEventListener("touchstart", handleBegin);
    $("bookface").addEventListener("mouseup", handleEnd);
    $("bookface").addEventListener("touchend", handleEnd);
    $("bookface").addEventListener("mouseout", handleEnd);
    $("bookface").addEventListener("mousemove", handleMoving);
    $("bookface").addEventListener("touchmove", handleMoving);
    $("body").addEventListener("keypress", function (ev) {
        $("aid").textContent += ev.key;
    });
```

图 7.1 触屏界面

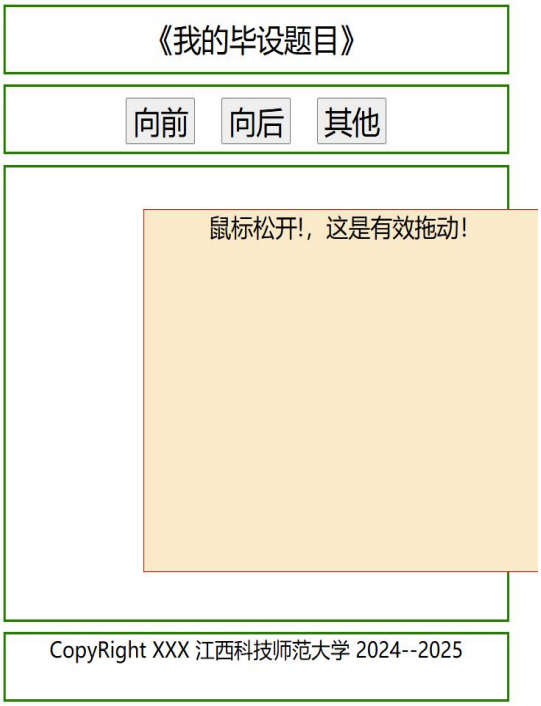


图 7.2 鼠标界面

下图为该阶段成果展示二维码：

8. UI 的个性化键盘交互控制的设计开发

Keyup 为释放键盘键，keydown 为按下键盘，用户在按下非字符建时最先触发的事件为 keydown 事件，最后触发的是 keyup 事件，而按下字符键时，在该事件期间还会触发 keypress 事件，最后再触发 keyup 事件，理解这些事件触发顺序以及对不同按键触发规则后，编写代码，实现对按下按键值的获取与显示，以及对发生事件的显示，实现截图为 8.1。深入探索和有效利用 keydown 和 keyup 事件，

不仅可以丰富 UI 的交互方式，还能提升应用的响应速度、可用性和可访问性，为用户提供更加流畅和高效的键盘操作体验，并且对键盘事件的有效利用，可以显著提升无障碍访问性，确保没有鼠标也能顺畅操作。
因为系统中只有一个键盘，所以我们在部署代码时，把键盘事件的监听设置在 DOM 文档最大的可视对象——body 上，通过测试，不宜把键盘事件注册在 body 内部的子对象中。代码如下所示：

```
    $("body").addEventListener("keydown",function(ev){
   ev.preventDefault() ; //增加“阻止事件对象的默认事件后”，不仅 keypress 事件将不再响应，而且系统的热键，如“F5 刷新页面/Ctrl+R ”、“F12 打开开发者面板”等也不再被响应
    let k = ev.key;
    let c = ev.keyCode;
    $("keyStatus").textContent = "按下键 ：" + k + " ，"+ "编码 ：" + c;
   });
```

```
$("body").addEventListener("keyup",function(ev){
   ev.preventDefault() ;
    let key = ev.key;
    $("keyStatus").textContent =  key + " 键已弹起" ;
    if (printLetter(key)){
       $("typeText").textContent += key ;
    }
```

```
    function printLetter(k){
     if (k.length > 1){ //学生须研究这个逻辑的作用
        return false ;
     }
     let puncs =
['~','`','!','@','#','$','%','^','&','*','(',')','-','_','+','=',',',
'.',';',':','<','>','?','/',' ','\'','\"'] ;
      if ( (k >= 'a' && k <= 'z') || (k >= 'A' && k <= 'Z')
|| (k >= '0' && k <= '9'))  {
        console.log("letters") ;
        return true ;
      }
     for (let p of puncs ){
      if (p === k) {
          console.log("puncs") ;
          return true ;
        }
     }
```

```
    return false ;
     //提出更高阶的问题，如何处理连续空格和制表键 tab?
   }  //function printLetter(k)
});
```

图 8.1 键盘 UI 界面

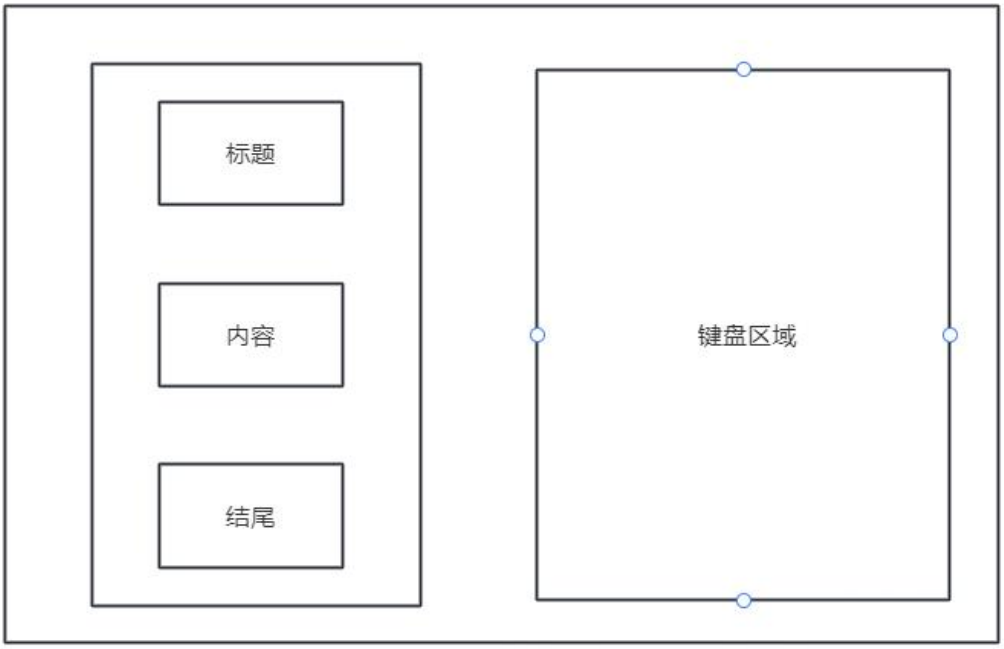


图 8.2 宽屏界面设计图

该页面二维码如下：

9. 谈谈本项目中的高质量代码

本次项目高质量代码有对元素 id 属性获取使用抽象函数$()进行获取，避免在获取元素 id 属性时需要重复编写固定的代码，减轻代码量的同时，使 id 属性的获取更加便捷和快速。其次较好的代码有创建一个 Pointer 对象，践行 MVC 设计模式，设计一套代码同时对鼠标和触屏实现控制。通过 pointer 对象模型可以减轻局部变量的定义，提高代码的利用率，以及在代码编写时通过对 touch 事件发生的判断，实现将鼠标拖动和触屏移动事件的区分，并同时利用同一套代码实现触屏移动和鼠标移动距离的确定和显示，极大提高了代码的利用率和个人开发者的工作量，而且逻辑清晰的代码结构为之后的维护和更新迭代提供了更好的代码环境。

10. 用 gitBash 工具管理项目的代码仓库和 http 服务器

10.1 Bash 介绍

Bash，全称 Bourne-Again SHell，是一种 Unix shell，由布莱恩·福克斯在 1987 年为 GNU 项目编写。它是 Bourne shell（sh）的后继版本和开源实现，名称来源于对 Bourne shell 的双关语（“Bourne again”/“born again”）。Bash 在 1989 年发布了第一个正式版本，最初设计用于 GNU 操作系统，但因其高度兼容性和扩展性，它能够运行在大多数类 Unix 系统上，包括 Linux 和旧版的 Mac OS X（如 v10.4）中，这些系统通常将其设为默认的 shell。

作为命令处理器，Bash 的主要职责是接收用户输入的命令，执行这些命令，并提供一个与操作系统交互的界面。它不仅可以在终端窗口中直接接受用户的即时输入，还可以读取并执行脚本文件中的命令序列，这些脚本文件包含了一系列预定义的 Bash 命令和控制结构，用于自动化复杂的任务。

Bash 支持的功能包括但不限于：

(1) 文件名替换（通配符匹配）：使用星号(*)、问号(?)等通配符匹配文件名。

(2) 管道：通过管道符号(|)将一个命令的输出作为另一个命令的输入，实现命令的串联。

(3) Here 文档：构造多行字符串，常用于向命令或脚本传递多行输入。

(4) 命令替换：使用反引号(```)或$(...)结构执行命令并将其输出嵌入到另一个命令中。

(5) 变量管理：支持定义、赋值和引用变量，包括环境变量和自定义变量。

(6) 条件判断与循环：提供 if、case 语句进行条件分支，以及 for、while、until 循环控制结构。

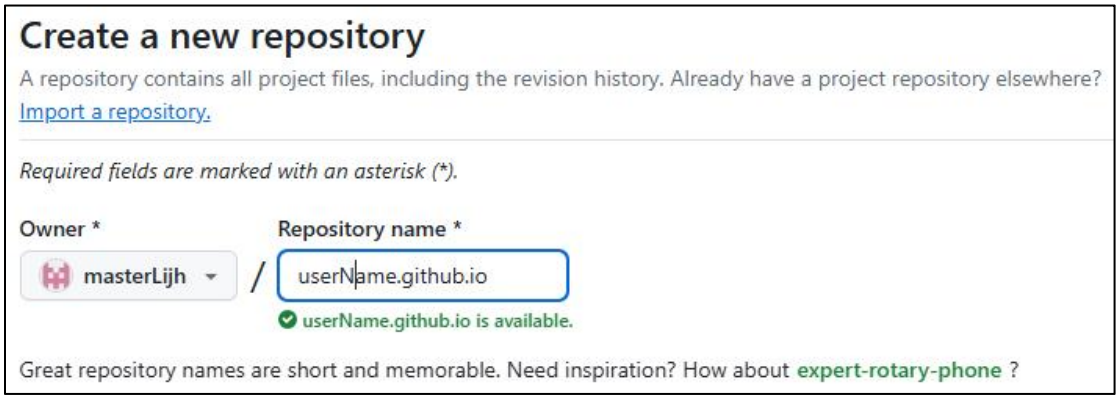
(7) 历史命令：记录用户输入的历史命令，方便回顾和重用。

函数：支持用户定义函数，以模块化的方式组织代码。

10.2 通过 gitHub 平台实现本项目的全球域名

https://jxsd.github.io/jiangxishida.github.io/

10.3 创建一个空的远程代码仓库



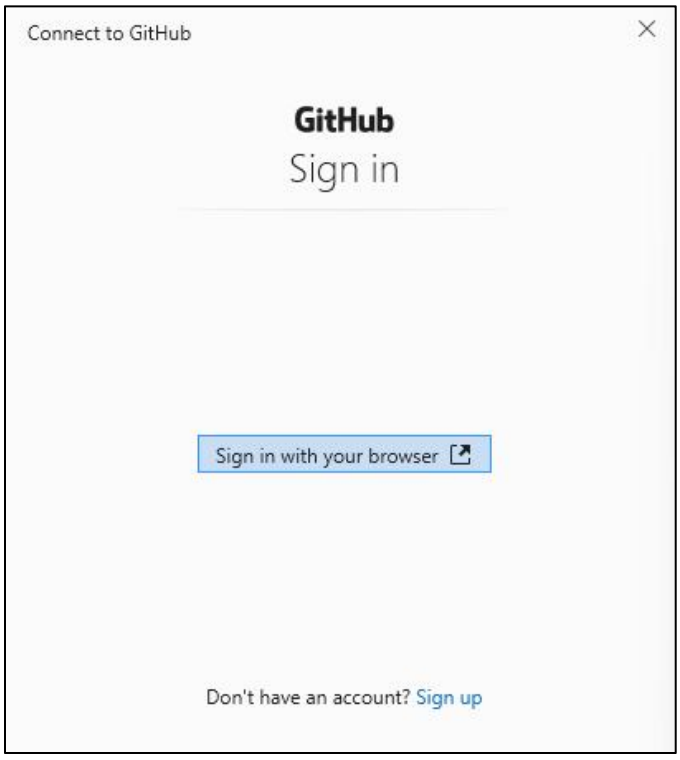
点击窗口右下角的绿色“Create repository”，则可创建一个空的远程代码仓库。

10.4 设置本地仓库和远程代码仓库的链接

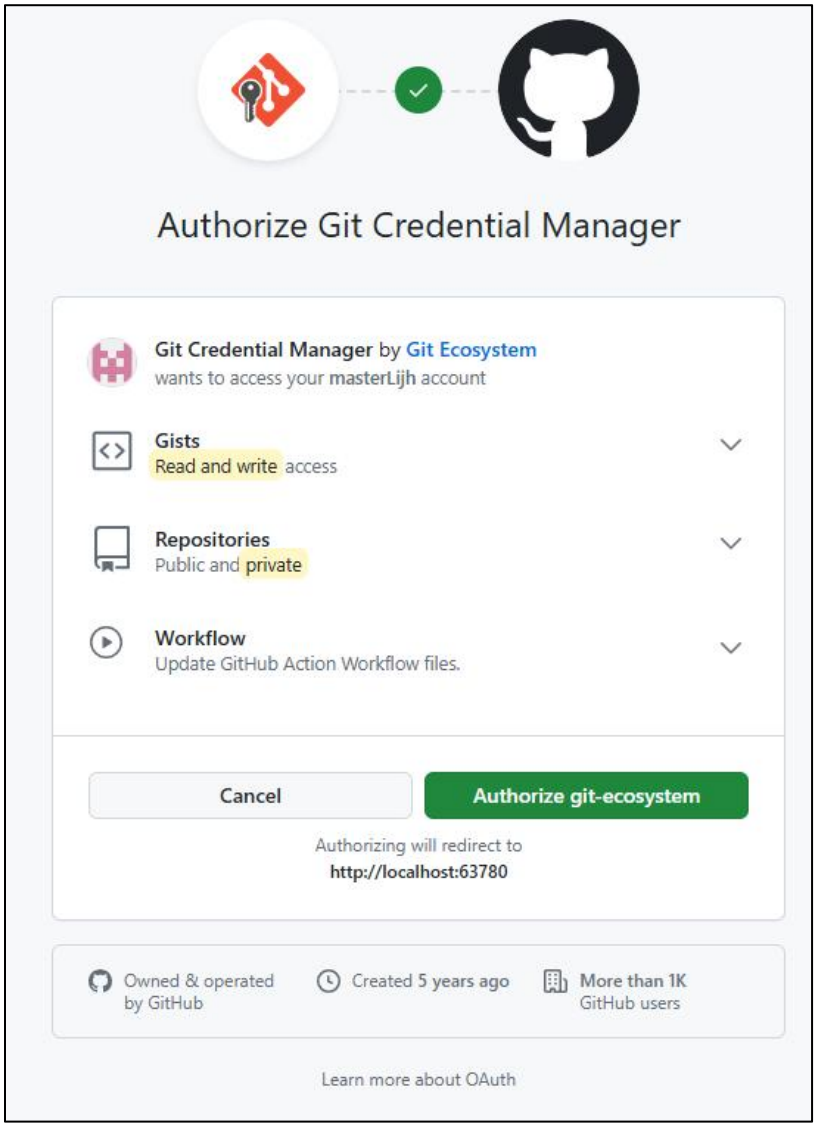
进入本地 webUI 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接

```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md
$ git init
$ git add README.md
$ git commit -m "这是我第一次把代码仓库上传至 gitHub 平台"
$ git branch -M main
$ git remote add origin
        https://github.com/jxsd/jiangxishida.github.io.git
$ git push -u origin main
```

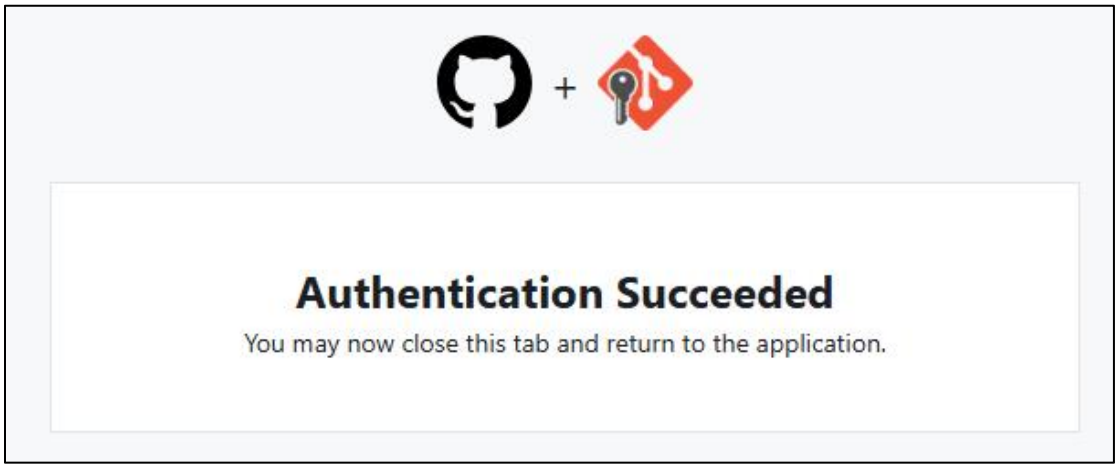
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



再次确认授权 gitBash 拥有访问改动远程代码的权限，如下图所示：



最后，GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。



从此，我们无论在本地做了任何多次代码修改，也无论提交了多少次，上传远程时都会把这些代码和修改的历史记录全部上传 github 平台，而远程上传命令则

可简化为一条：git push ，极大地方便了本 Web 应用的互联网发布。

远程代码上传后，项目可以说免费便捷地实现了在互联网的部署，用户可以通过域名或二维码打开，本次使用 PC 的微软 Edge 浏览器打开，本文截取操作中间的效果图，如下所示：

全文完成，谢谢！

参考文献：

[1] W3C. W3C's history. W3C Community. [EB/OL]. https://www.w3.org/about/. https://www.w3.org/about/history/. 2023.12.20

[2] Douglas E. Comer. The Internet Book [M] (Fifth Edition). CRC Press Taylor & Francis Group, 2019: 217-218

[3] John Dean,PhD. Web programming with HTML5,CSS,and JavaScript[M]. Jones & Bartlett Learning,LLC. 2019: 2

[4] John Dean,PhD. Web programming with HTML5,CSS,and JavaScript[M]. Jones & Bartlett Learning,LLC. 2019: xi

[5] Behrouz Forouzan. Foundations of Computer Science[M](4th Edition). Cengage Learning EMEA,2018: 274--275

[6] Marijn Haverbeke. Eloquent JavaScript 3rd edition. No Starch Press,Inc, 2019.

[7] William Shotts. The Linux Command Line, 2nd Edition [M]. No Starch Press, Inc, 245 8th Street, San Francisco, CA 94103, 2019：3-7