成 绩：

江 西 科 技 师 范 大 学

毕业设计（论文）

题目（中文）：基于 Web 客户端技术的个性化 UI 设计和实现

题目（外文）：Web client based customized UI design and Programming

院（系）：元宇宙产业学院

专　　业：计算机科学与技术

学生姓名：彭丽娟

学　　号：20213628

指导教师：李健宏

2024 年 6 月 18 日

目录

基于 Web 客户端技术的个性化 UI 设计和实现

摘要：在最近十年的发展中，基于 HTML5 的 Web 标准软件开发技术因其跨平台兼容性和开源性，已经在众多软件开发领域中获得了广泛的应用。本项目以 Web 客户端技术作为研究和实践的重点，对程序设计和软件开发进行了深入的研究。我们通过深入分析技术文档、参与开发者社区的讨论和阅读专业书籍，开发了一个个性化的用户界面（UI）应用程序。在开发过程中，我们直接使用了 Web 客户端的基础 API，并利用 HTML 来构建网页结构，CSS 来设计界面外观，JavaScript 来实现用户交互功能。项目还实施了响应式设计，以适应移动互联网时代用户设备的多样化需求。此外，项目采用了面向对象的设计原则，开发了一个统一处理鼠标和触屏交互的通用指针模型，这显著提高了代码的质量和效率。在项目管理上，我们采取了增量开发策略，通过六轮的敏捷开发和代码重构，成功完成了设计和开发，并确保了项目的测试通过。在代码共享方面，我们使用 Git 进行版本控制，经过六次代码重构和正式提交，以及两次在测试阶段的代码修改，最终使用 Git Bash 将代码库推送至 GitHub。通过 GitHub 的 HTTP 服务，我们的 UI 应用得以在全球范围内部署，用户现在可以通过网络链接或二维码在各种设备上便捷地访问。

关键字：Web 客户端；响应式设计；面向对象；触屏交互；

Personalized UI design and implementation based on Web client technology

Abstract: In the last decade of development, the Web standard software development technology based on HTML 5 has been widely used in many software development fields due to its cross-platform compatibility and open source nature. This project takes Web client technology as the focus of research and practice, and studies program design and software development. We have developed a personalized user interface (UI) application through in-depth analysis of technical documentation, participating in discussions in the developer community, and reading professional books. During the development process, we directly used the basic API of the Web client, and used HTML to build the web page structure, CSS to design the interface appearance, and JavaScript to realize the user interaction function. The project has also implemented a responsive design to meet the diverse needs of users devices in the mobile Internet era. In addition, the project adopts object-oriented design principles to develop a universal pointer model that uniformly handles mouse and touch-screen interactions,which significantly improves the quality and efficiency of the code. In terms of project management, we adopted an incremental development strategy, through six rounds of agile development and code refactoring, successfully completed the design and development, and ensured that the test of the project passed. In terms of code sharing, we used Git for version control, after six code refactoring and formal submission, and two code modifications in the test phase, and finally pushed the code base to GitHub using Git Bash. With GitHub's HTTP service, our UI applications have been deployed worldwide, and users can now easily access them on a variety of devices through network links or QR codes.

Keywords: Web client; responsive design; object-oriented; touch-screen interaction;

1.前言

1.1 毕业任务分析

本设计分为两个主要步骤，首先是挑选一条自己感兴趣的技术实践路径，重点学习核心技术，参考导师的案例项目，理解技术间的联系、各自在项目中的角色和职责，同时在执行项目中提高编写高质量代码的能力。一旦模仿导师案例的技术实现基本完成，就说明理论与实践已经结合，可以进入第二阶段，开始独立设计软件。第二阶段遵循软件工程规范进行开发：1. 针对个人问题进行定义和分析；2. 制定一套合适的技术解决方案；3. 根据方案设计流程，编写代码并部署；4. 调试代码，测试软件，进行性能优化。步骤 3 和 4 可能会发现前序步骤的问题，导致需要在步骤 2、3、4 之间多次迭代，修正设计缺陷或代码错误。大部分工作集中在步骤 3，即构建代码体系和实施软件架构的细节。学习 Web 标准和相关技术，编写 Web 程序，使用相关工具，最终构建一套能在多平台上运行的高质量代码应用，这构成了所谓的 Web 应用开发技术栈。

1.2 研究方法

研究方法采用模型法，论文设计中采用了 UML 进行问题建模。UML 较为抽象，设计精确度较高，我建议先编写代码建立模型，待代码运行成功后，再使用 UML 语言绘制模型，作为文档资料，这样更为合理。

2.Web 平台和客户端技术概述

网络的创始人 Tim Berners-Lee 在创造互联网的基础技术后，建立了 W3C 组织[1]。该组织在 2010 年之后推出了 HTML5 这一国际标准，与欧洲 ECMA 组织维护的 ECMAScript 标准相结合，为全球开发者提供了一个统一的开发平台，这一理想至今仍在不断被科学家和网络行业所追求和完善。我的毕业设计项目采用了学习 Web 标准和相关技术，编写 Web 程序，应用相关工具的技术路径，目标是构建一套能够在不同平台上运行的高质量代码应用。

2.1 Web 技术的历史背景

1989 年，蒂姆·伯纳斯-李爵士在 CERN 提出了万维网概念，目的是通过互联网自由共享信息。1990 年 10 月，他开发了首个万维网服务器"httpd"和集成浏览器与编辑器的客户端程序"WorldWideWeb"，奠定了万维网技术的基础。伯纳斯-李爵士创造了"万维网

″术语，并编写了 HTML 初版，引入超链接功能，成为网页设计的主要语言。他还规范了 URI 和 HTTP，确立了网络通信的基础。随着技术发展，这些规范得到全球开发者的优化和扩展，推动了万维网功能的增强和互联网的普及。伯纳斯-李爵士被誉为″万维网之父″，其贡献对社会发展产生了深远影响，将被永久铭记[2]。

2.2 万维网互联

万维网（World Wide Web，简称 WWW 或 Web）的互联是指通过互联网技术，将全球范围内的计算机和服务器连接起来，形成一个庞大的、分布式的、可交互的信息网络。这个网络允许用户通过浏览器等客户端软件访问、浏览、发布和共享各种信息资源。万维网互联的基础是互联网（Internet），它是一个全球性的、开放的网络，由各种不同类型的网络、计算机、服务器和通信设备组成。互联网使用 TCP/IP 协议族进行通信，这些协议定义了数据如何在网络中传输、路由和寻址。在万维网中，网页（Web page）是基本的信息单元，它们使用 HTML（超文本标记语言）编写，可以包含文本、图像、音频、视频等多种媒体形式。网页通过超链接（hyperlink）相互连接，形成一个庞大的、复杂的网络结构。用户可以通过点击超链接来访问不同的网页，从而实现信息的浏览和导航。Web 服务器是万维网互联中的关键组成部分，它们负责存储、处理和传输网页文件。当用户请求访问某个网页时，Web 服务器会接收请求，并根据请求的内容将相应的网页文件发送给用户的浏览器。浏览器则负责解析和渲染网页文件，将其呈现给用户。除了 Web 服务器和浏览器之外，万维网互联还涉及许多其他的技术和工具，如搜索引擎、电子邮件、社交媒体等。这些技术和工具为用户提供了更加便捷、高效的信息获取和交流方式，使得万维网成为了现代社会中不可或缺的一部分[3]。

2.3 Web 平台

Web 平台是一种新兴的应用程序、信息和服务等的集成平台。Web 平台提供了一个完备的基础架构，使多个系统媒介连接，实现跨越不同系统之间进行数据交换、访问资源等高效信息交换；用户可以通过 Web 浏览器访问 Web 平台上的网络应用程序、数据、服务等，无需安装其他客户端软件；Web 平台提供了一个高度可维护的环境，降低系统维护和更新的技术难度，提高系统可靠性；采用先进的数据安全技术，对 Web 应用程序上的数据和用户信息进行安全可靠的管理，保证用户隐私的安全；Web 平台支持多媒体文件，为 Web 用户提供更加丰富的信息和服务；Web 平台可以提供多种应用程序、服务和数据，用户可按需组合多种应用程序和服务，获得更完美的系统服务[4]。Web 平台在

现代信息化系统中应用广泛，如电子商务、网上书店、团购网站、社交网络、在线学习等领域。

2.4 Web 编程

Web 编程是指使用各种编程语言和技术来开发和实现基于互联网的应用程序的过程。它涉及到前端开发、后端开发和数据库开发等多个层面。前端开发：主要负责网页用户界面的设计和实现，包含 HTML、CSS 和 JavaScript 等技术的应用。HTML 用于定义网页的结构，CSS 用于控制网页的布局和样式，而 JavaScript 用于实现网页的交互效果。后端开发：主要负责服务器端的逻辑实现和数据处理，可以使用多种语言和框架，如 PHP、Java、Python 等。后端开发的任务包括处理用户请求、访问数据库、生成动态页面等。数据库开发：主要负责设计和维护数据库系统，如 MySQL、Oracle、SQL Server 等。数据库是 Web 应用程序的核心，用于存储和管理大量的数据。Web 编程的目标是创建具有丰富功能和良好用户体验的网站和应用程序，其核心技术包括 HTML、CSS、JavaScript、PHP、Java 等。开发人员根据需求选择适合的编程语言和技术进行开发，实现网页的动态化和交互性[5]。

2.5 项目的增量式迭代开发模式

本项目作为本科学生的毕业设计软件作品，相较于单一功能的程序，其复杂性和手写代码量显著增大，远超简单项目的数量级。从问题的分析到代码的初步编写，这一过程并非短期内能够完成，而是需要系统工程的视角来规范和管理整个编写流程。在软件工程的开发过程中，我们考虑了两种经典的管理模式：瀑布模型和增量式迭代模型。尽管每种模式都涵盖了分析、设计、实施和测试这四个阶段，但它们的执行方式有所不同。

瀑布模型强调各阶段之间的严格顺序和依赖关系，要求前一阶段完美完成后才能进入下一阶段[6]。然而，对于许多小型开发者或身兼数职的开发者来说，很难保证每次都能一次性完美完成所有工作。在实施过程中，开发者可能会发现设计上的不足，需要在后续的迭代中进行改进。在当今开源的软件开发环境中，持续优化设计和重构代码已成为常态。因此，在本项目的开发中，我们选择了增量模型的开发模式。通过六次项目的开发迭代，我们不断完善和改进项目的各个方面，确保了项目的顺利进行和高质量交付。

3.内容设计概要

3.1 分析和设计

本设计采用用户熟悉的“三步法”来构建内容，首先通过标题信息展示品牌标识或文字标题以吸引目光，接着是核心内容区，遵循“内容至上”的原则，这是项目的核心理念和 UI 设计的核心主题，最后是底部的补充信息，用于展示用户可能感兴趣的细节。如下图 3-1 所示：

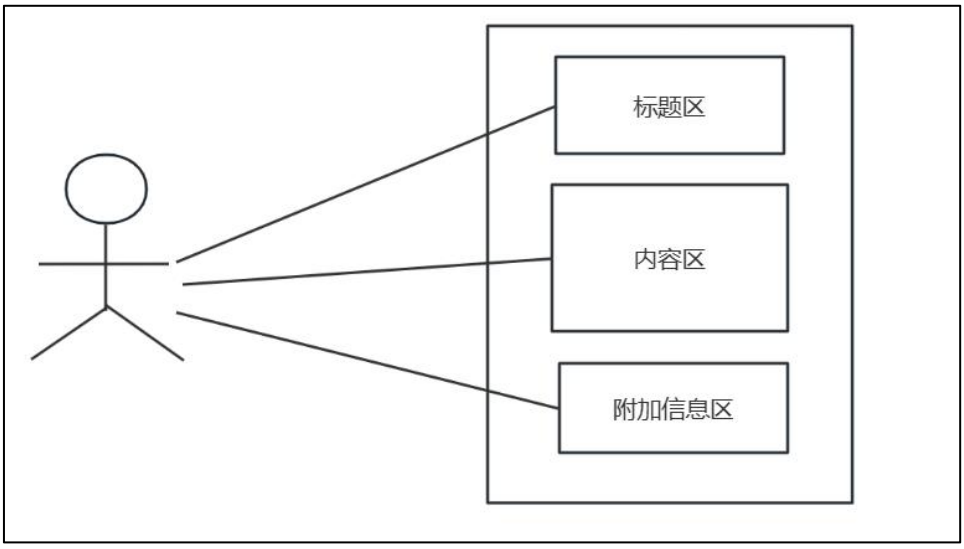


图 3-1 用例图

3.2 项目的实现和编程

一、HTML 代码

下面这段代码块是该项目的 HTML 部分，该部分需要定义了一个包含页眉、导航和页脚的简单网页结构，同时页眉部分包含了一个标识为"book"的段落，而导航部分包含了项目描述的段落，页脚部分包含了版权和作者信息。

```
<header>
  <p id="book">
   基于 Web 客户端技术的个性化 UI 设计和实现@studentPenglj
  </p>
 </header>
 <nav>
  在过去十年，HTML5 技术因其跨平台和开源优势，在软件开发中广泛应用。本项目专注于 Web 客户端技术，深入研究程序设计和软件开发。我们通过分析文档、社区讨论和阅读书籍，开发了一个个性化 UI 应用。开发中使用了 WebAPI、HTML、CSS 和 JavaScript，实现了响应式设计和面向对象设计原则，提高了代码质量和效率。项目采用增量开发策略和敏捷开发流程，通过六轮开发和代码重构，保证了测试通过。使用 Git 进行版本控制，最终通过 GitHub 将应用部署，用户可通过网络链接或二维码在不同设备上访问。
 </nav>
 <footer>
  <p id="statusInfo">
   CopyRight from 彭丽娟 江西科技师范大学 2021--2025
```

```
 </p>
</footer>
```

二、CSS 代码

下面这段 CSS 样式代码块设置了 HTML 中<body>、<header>、<footer>和<nav>元素的样式。它规定了<body>中的字体大小，以及<header>、<footer>和<nav>的边框和高度，从而改变 HTML 中对应部分的外观。

```
<style>
 body {
  font-size: 20px;
  font-family: "楷体";
 }
 header {
  border: 2px solid blue;
  height: 150px;
 }
 footer {
  border: 2px solid blue;
  height: 150px;
 }
 nav {
  border: 2px solid blue;
  height: 400px;
 }
</style>
```

3.3 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 Edge 浏览器打开项目的结果，如下图 3-2 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 3-3 的二维码，运行测试本项目的第一次开发的阶段性效果。

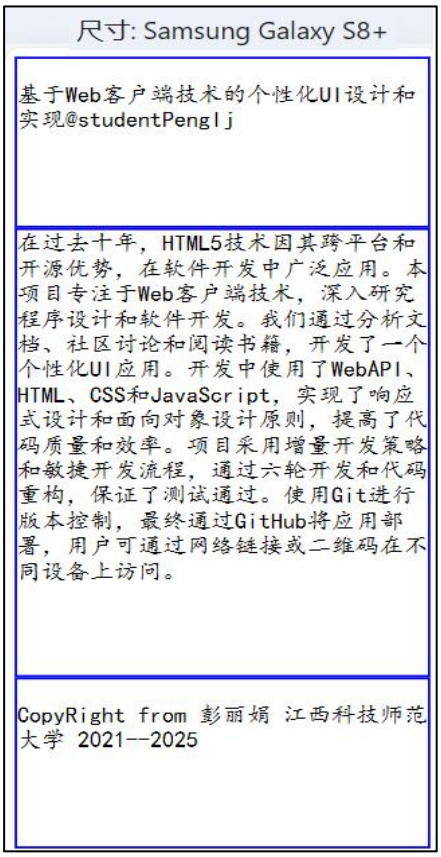


图 3-2 PC 端运行效果截图

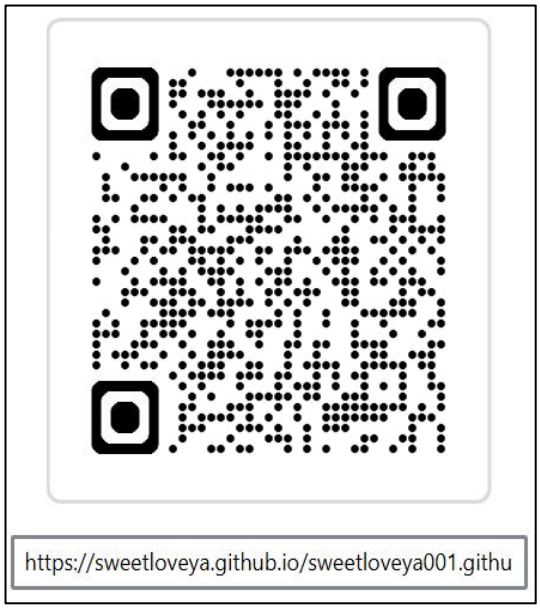


图 3-3 移动端二维码

3.4 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。

成功提交代码后，gitbash 的反馈如下所示：

```
27862@sweetloveya MINGW32 /abc (master)
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
[master (root-commit) 33afd11] 项目第一版：“三段论”式的内容设计概要开发
 1 file changed, 47 insertions(+)
 create mode 100644 index.html
```

gitbash 反馈代码的仓库日志如下所示：

```
27862@sweetloveya MINGW32 /abc (master)
$ git log
commit 33afd110c6cfd6eea9c878aa744bcb0880ca61b2 (HEAD -> master)
Author: 彭丽娟 <2786219140@qq.com>
Date:   Tue Jun 4 11:46:41 2024 +0800

    项目第一版：“三段论”式的内容设计概要开发
```

4. 移动互联时代的响应式设计和窄屏代码实现

4.1 响应式设计与实现

计算机显示器的规格多种多样，其尺寸和清晰度通常受价格因素影响。网页设计者并不为每个网页定制特定版本，而是制定一套通用的布局规范，让浏览器根据具体的硬件环境来决定页面的展示方式。因此，网页设计通常不会涉及过多细节。例如，设计者可以决定哪些句子构成一个段落，但无法指定具体的行长或段落缩进等细节。这种由浏览器决定具体展示效果的方式，导致同一网页在不同浏览器或不同硬件上可能会呈现出不同的外观。如果屏幕宽度不同，那么文本的行长或图像大小也会有所差异。核心在于：网页提供了展示的框架性指导，而具体的展示细节则由浏览器根据设备特性来确定。因此，同一网页在不同设备或浏览器上可能会有轻微的展示差异。下图是该部分的大致结构。

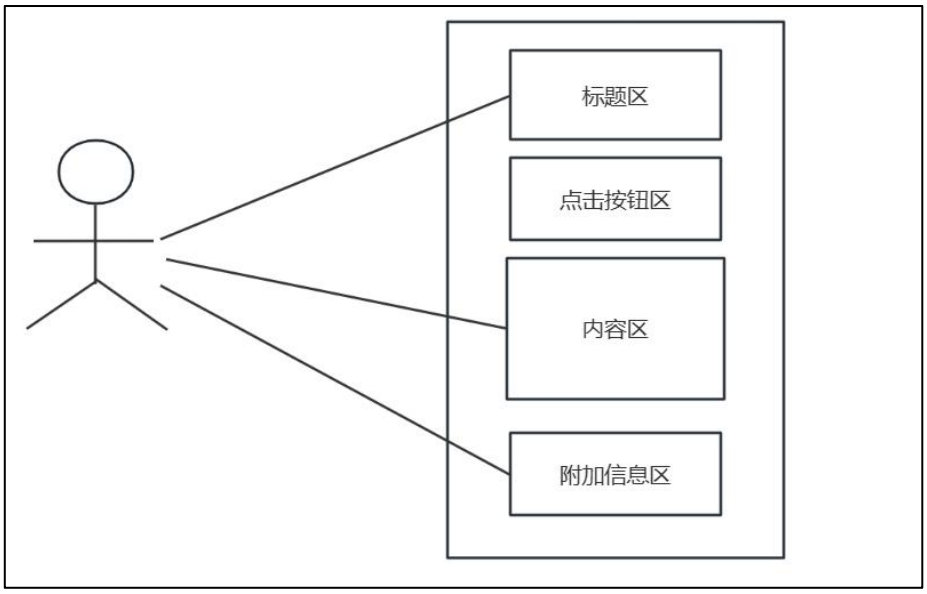


图 4-1 用例图

4.2 编辑与实现

本次代码进行了应用程序的响应式设计:1.为四个区域设置了占整个用户界面的比例（以高度为准）;2.用 js 动态获取了用户屏幕的宽度和高度,并依据我们的需要进行了计算,设置了基础字体的大小;3.在不同的区域根据内容的重要程度设定了该区域字体的相对大小;4.改变了 UI 的颜色和背景色，对主区域的背景图做了优化外观的设计；总之，初步为用户的应用程序做了响应式设计并部署了代码。JavaScript 部分的代码如下图所示:

```
<script>
  var UI = {};
  UI.deviceWidth = window.innerWidth;
  UI.deviceHeight = window.innerHeight;
  document.body.style.height = UI.deviceHeight + "px";
  document.body.style.fontSize = UI.deviceWidth / 36 + "px";
</script>
```

4.3 运行和测试

该项目在 PC 端用 Edge 浏览器打开项目的结果，如下图 4-2 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 4-3 的二维码，运行测试本项目的第二次开发的阶段性效果。

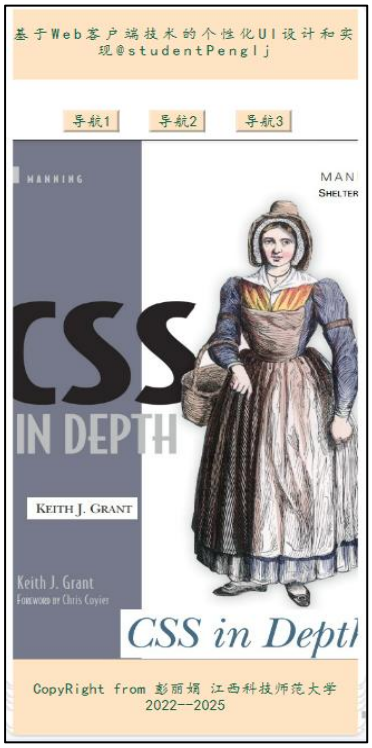


图 4-2 PC 端运行效果截图

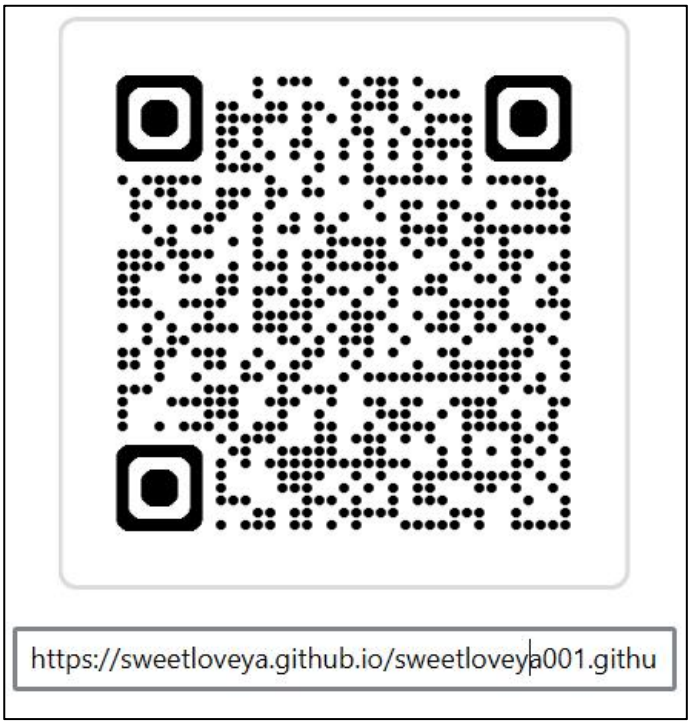


图 4-3 移动端二维码

4.4 代码提交和版本管理

成功提交代码后，gitbash 的反馈如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git commit -m 项目第二版：移动互联时代的响应式设计和窄屏代码实现
[main 3dd8016] 项目第二版：移动互联时代的响应式设计和窄屏代码实现
 1 file changed, 97 insertions(+)
 create mode 100644 1.2.html
```

gitbash 反馈代码的仓库日志如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git log
commit 3dd8016f32731639af48b5a7801d50cd4e9bab0a (HEAD -> main)
Author: masterLijh <maste_rlee.qq.com>
Date:   Wed Jun 5 09:45:53 2024 +0800

    项目第二版：移动互联时代的响应式设计和窄屏代码实现
```

5. 适用移动互联时代的响应式设计

5.1 响应式设计与实现

该部分项目主要实现了对鼠标点击事件和敲击键盘事件的捕捉，具体结构如下图所示：

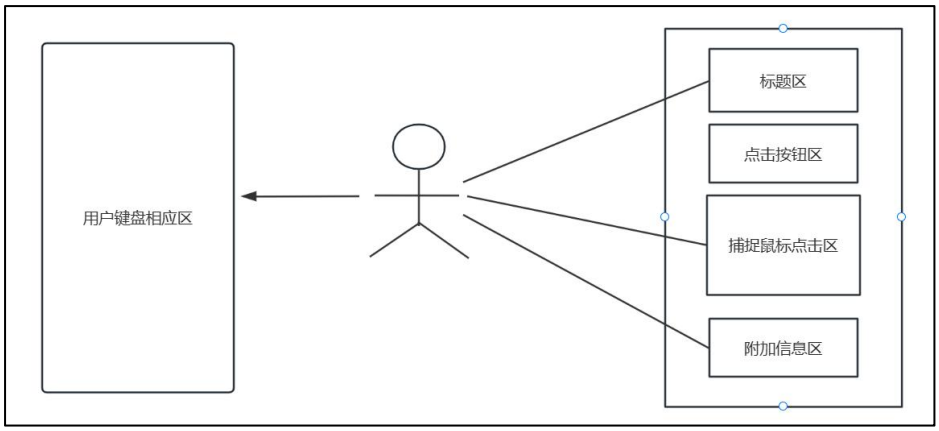


图 5-1 用例图

5.2 编辑与实现

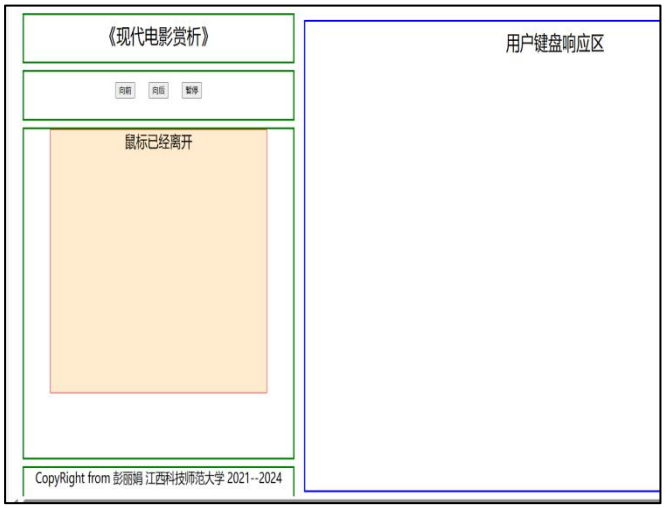
该项目定义了一个 UI 对象，用于存储应用的宽度和高度，并根据屏幕宽度设置字体大小和页面尺寸，通过改变 body 的 fontSize 属性，实现响应式设计。根据屏幕宽度，设置辅助区域 aid 的显示和尺寸。定义了一个 mouse 对象，用于跟踪鼠标的状态和位置。

为 bookface 元素添加了鼠标事件监听器，包括鼠标按下、移动和离开事件，以更新页面内容。为 body 元素添加了键盘事件监听器，用于响应按键事件，并更新辅助区域 aid 的内容。定义了一个自定义的$函数，用于简化 DOM 元素的选择过程，支持通过 ID 或 CSS 选择器获取元素。JS 部分代码如下：

```
$("bookface").addEventListener("mousedown", function (ev) {
     let x = ev.pageX;let y = ev.pageY;
     console.log("鼠标按下了，坐标为：" + "(" + x + "," + y + ")");
     $("bookface").textContent = "鼠标按下了，坐标为：" + "(" + x + "," + y +
")";});
   $("bookface").addEventListener("mousemove", function (ev) {
     let x = ev.pageX;let y = ev.pageY;
     console.log("鼠标正在移动，坐标为：" + "(" + x + "," + y + ")");
   $("bookface").textContent = "鼠标正在移动，坐标为：" + "(" + x + "," + y +
")";});
   $("bookface").addEventListener("mouseout", function (ev) {
   $('bookface').textContent = "鼠标已经离开";});
   $("body").addEventListener("keypress", function (ev) {
      let k = ev.key;
      let c = ev.keyCode;
      let s1 = "按键是：";
      let s2 = "编码是：";
   $("keyboard").textContent = s1 + k + " " + s2 + c;
       })
```

5.3 运行和测试

该项目在 PC 端用 Edge 浏览器打开项目的结果，如下图 5-2 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 5-3 的二维码，运行测试本项目的第三次开发的阶段性效果。



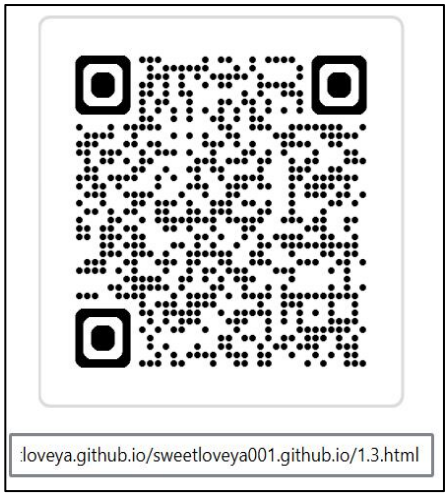


图 5-2 PC 端运行效果截图　　图 5-3 移动端二维码

5.4 代码提交和版本管理

成功提交代码后，gitbash 的反馈如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git commit -m 移动互联时代的响应式设计对鼠标点击事件的捕捉
[main 843f5ef] 移动互联时代的响应式设计对鼠标点击事件的捕捉
 1 file changed, 183 insertions(+)
 create mode 100644 1.3.html
```

gitbash 反馈代码的仓库日志如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git log
commit 3dd8016f32731639af48b5a7801d50cd4e9bab0a (HEAD -> main)
Author: masterLijh <maste_rlee.qq.com>
Date:   Wed Jun 5 09:45:53 2024 +0800

    项目第二版：移动互联时代的响应式设计和窄屏代码实现
```

6. 个性化 UI 设计中鼠标模型

6.1 设计与实现

该部分项目主要是实现对鼠标事件的监听，具体结构如下图所示：

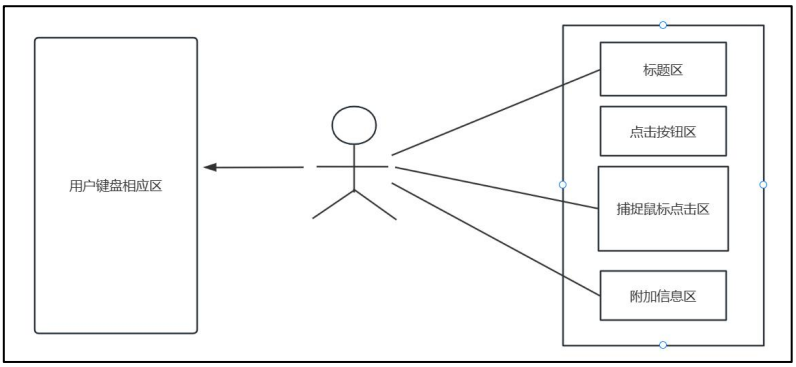


图 6-1 用例图

6.2 编程与实现

该部分项目主要是实现对鼠标事件的监听，其中 mousedown 是记录当鼠标在 bookface 元素上按下时，记录鼠标按下的位置，并将状态标记为 isDown；而 mousemove 主要记录当鼠标移动时，如果已经按下，则计算鼠标的移动距离，并实时更新 bookface 元素的位置和文本内容；mouseup 和 mouseout 主要作用是当鼠标松开或离开 bookface

元素时，根据鼠标移动的距离判断是否为有效拖动，并更新文本内容。如果移动距离小于 100 像素，则重置 bookface 元素的位置。JS 部分代码如下：

```
  $("bookface").addEventListener("mouseup", function (ev) {
       mouse.isDown = false;
  $("bookface").textContent = "鼠标松开!";
       if (Math.abs(mouse.deltaX) > 100) {
      $("bookface").textContent += "，这是有效拖动！";
} else {
        $("bookface").textContent += " 本次算无效拖动！";
        $("bookface").style.left = '7%';
}});
   $("bookface").addEventListener("mouseout", function (ev) {
      ev.preventDefault();
      mouse.isDown = false;

  $("bookface").textContent = "鼠标松开!";
    if (Math.abs(mouse.deltaX) > 100) {
       $("bookface").textContent += " 这次是有效拖动！";
 }
   else {
       $("bookface").textContent += " 本次算无效拖动！";
       $("bookface").style.left = '7%';}});
       $("bookface").addEventListener("mousemove", function (ev) {
       ev.preventDefault();
      if (mouse.isDown) {
      console.log("mouse isDown and moving");
       mouse.deltaX = parseInt(ev.pageX - mouse.x);
    $("bookface").textContent = "正在拖动鼠标，距离：" + mouse.deltaX + "px ";
     $('bookface').style.left = mouse.deltaX + 'px';
 }
 });
```

6.3 运行和测试

该项目在 PC 端用 Edge 浏览器打开项目的结果，如下图 6-2 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 6-3 的二维码，运行测试本项目的第四次开发的阶段性效果。

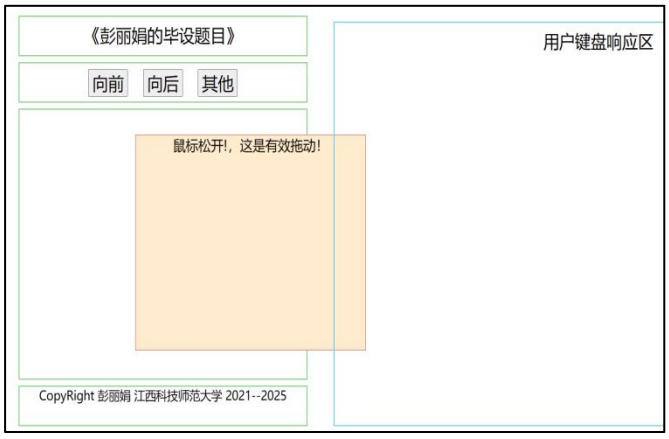


图 6-2 PC 端运行效果截图

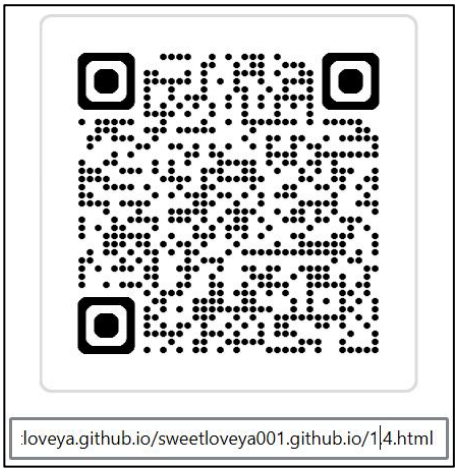


图 6-3 移动端二维码

6.4 代码提交和版本管理

成功提交代码后，gitbash 的反馈如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git commit -m 项目第四版：实现对鼠标事件的监听
[main 8555326] 项目第四版：实现对鼠标事件的监听
 1 file changed, 199 insertions(+)
 create mode 100644 1.4.html
```

gitbash 反馈代码的仓库日志如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git log
commit 85553269dd91c0930f20a16db7dd6bb43eaebf0c (HEAD -> main)
Author: masterLijh <maste_rlee.qq.com>
Date:   Wed Jun 5 11:28:10 2024 +0800

    项目第四版：实现对鼠标事件的监听
```

7. 对触屏和鼠标的通用交互操作的设计开发

7.1 设计与实现

该部分项目是同时适用于触屏和鼠标的通用交互操作，不仅能够提升用户体验，还能增强应用的可访问性和灵活性。具体结构如下图 7-1 所示：

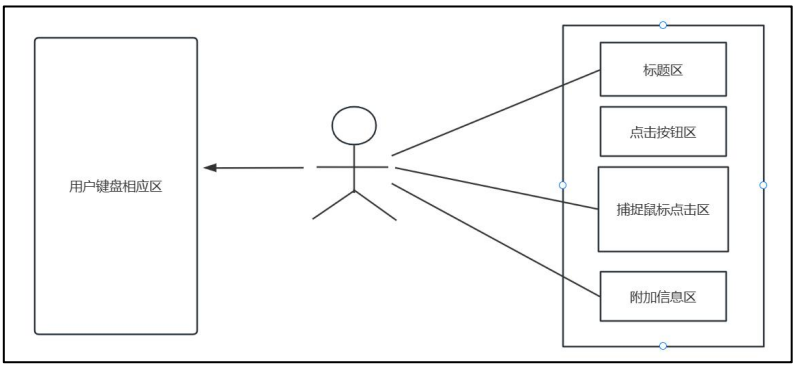


图 7-1 用例图

7.2 编辑和实现

该部分项目主要是创建一个交互式的拖动或滑动效果，无论是通过鼠标还是触屏，用户都可以在网页上拖动一个元素，并且可以看到拖动的距离和动态更新的文本内容。具体 JS 代码部分如下：

```
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;
    });
```

7.3 运行和测试

该项目在 PC 端用 Edge 浏览器打开项目的结果，如下图 7-2 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 7-3 的二维码，运行测试本项目的第五次开发的阶段性效果。

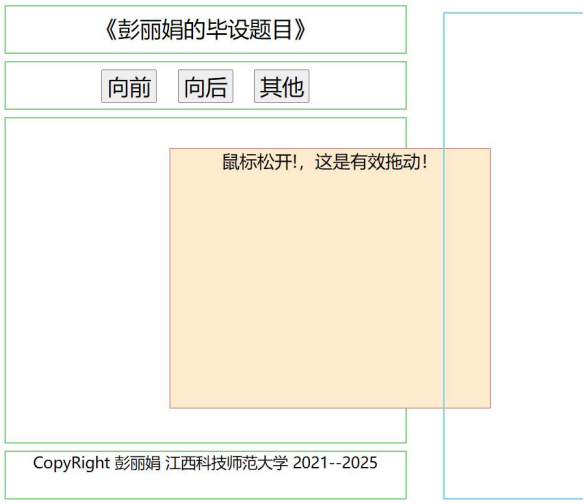


图 7-2 PC 端运行效果截图

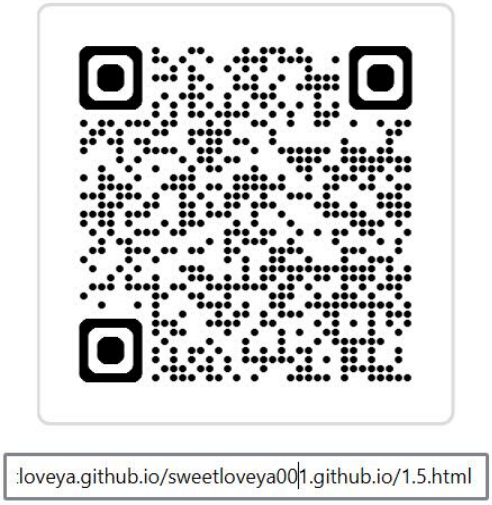


图 7-3 移动端二维码

7.4 代码提交和版本管理

成功提交代码后，gitbash 的反馈如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git commit -m 项目第五版：对触屏和鼠标的通用交互操作的设计开发
[main 2cafb45] 项目第五版：对触屏和鼠标的通用交互操作的设计开发
 1 file changed, 226 insertions(+)
 create mode 100644 1.5.html
```

gitbash 反馈代码的仓库日志如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git log
commit 2cafb457172ca7e426e385fc66b67c85402fe083 (HEAD -> main)
Author: masterLijh <maste_rlee.qq.com>
Date:   Wed Jun 5 15:43:52 2024 +0800

    项目第五版：对触屏和鼠标的通用交互操作的设计开发
```

8.UI 的个性化键盘交互控制的设计开发

8.1 设计与实现

该部分项目是实现对键盘交互控制，采用 keydown 和 keyup 键盘底层事件，为未来的键盘功能提供更大空间，具体结构如下图 8-1 所示：

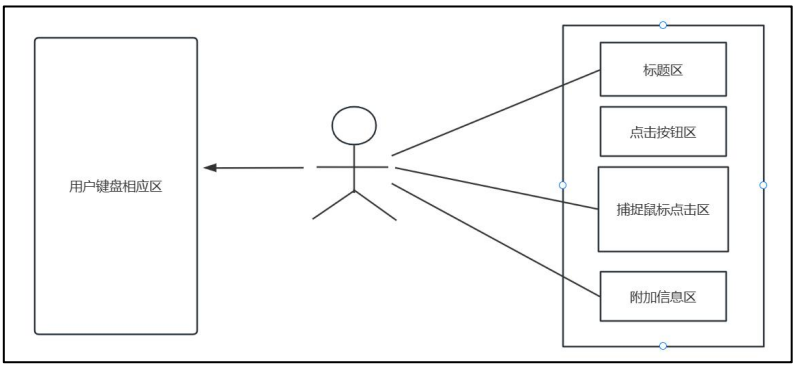


图 8-1 用例图

8.2 编辑与实现

keydown 事件：当用户按下键盘上的任意键时，会触发此事件监听器。ev.key: 事件对象的 key 属性包含了被按下的键的值，通常是按键的字符表示，如"a"、"1"等。ev.keyCode:事件对象的 keyCode 属性包含了被按下的键的编码。更新页面上 id 为 keyboard 的元素的文本内容，显示按下的键和对应的编码。当用户释放键盘上的任意键时，会触发此事件监听器。

keyup 事件与键盘按下事件类似，这里也获取了 ev.key 和 ev.keyCode。更新页面上 id 为 keyboard 的元素的文本内容，显示释放的键和对应的编码。printLetter 函数用于判断释放的键是否是字母、数字或标点符号。如果是，则允许将该键的字符追加到页面上 id 为 typeText 的元素的文本内容中。

```
$("body").addEventListener("keydown", function (ev) {
        let k = ev.key;
        let c = ev.keyCode;
      $("keyboard").textContent = "您已按键 ：" + k + " ，" + "字符编码 ：" + c;
      });
      $("body").addEventListener("keyup", function (ev) {
        let key = ev.key;
```

```
        let code = ev.keyCode;
        $("keyboard").textContent = "松开按键 ： " + key + " ， " + "
字符编码 ： " + code;
        if (printLetter(key)) {
          $("typeText").textContent += key;}
        function printLetter(k) {
          if (k.length > 1) { //学生必须研究这个逻辑的作用
            return false;}
          let puncs = ['~', '`', '!', '@', '#', '$', '%', '^', '&', '*',
'(', ')', '-', '_', '+', '=', ',', '.', '<', '>', '?', '/', ' '];
          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;} });
```

8.3 运行与测试

该项目在 PC 端用 Edge 浏览器打开项目的结果，如下图 8-2 所示。同时，该项目需要测试手机端的界面如下图 8-3 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 8-4 的二维码，运行测试本项目的第五次开发的阶段性效果。

图 8-2 PC 端运行效果截图

图 8-3 手机端截图

图 8-4 移动端二维码

8.4 代码提交与版本管理

成功提交代码后，gitbash 的反馈如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git commit -m 项目第六版：UI的个性化键盘交互控制的设计开发~
[main 471ced8] 项目第六版：UI的个性化键盘交互控制的设计开发~
 1 file changed, 267 insertions(+)
 create mode 100644 1.6.html
```

gitbash 反馈代码的仓库日志如下所示：

```
27862@sweetloveya MINGW32 /d/abc (main)
$ git log
commit 471ced87d4e09e1fe67e17032c465b36d3bcb444 (HEAD -> main)
Author: masterLijh <maste_rlee.qq.com>
Date:   Wed Jun 5 16:19:37 2024 +0800

    项目第六版：UI的个性化键盘交互控制的设计开发~
```

9. 本项目中的高质量代码

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，是一项将面向对象编程原则与现代编程范式相结合的挑战。在实现过程中，我们将使用局部变量来减少全局变量的使用，这有助于避免潜在的命名冲突和提高代码的可维护性。同时，我们也会采用函数式编程的原则，例如使用高阶函数和不可变数据结构，来提高代码的可读性和可测试性。逻辑上，我们将确保 Pointer 类的移动方法能够正确处理坐标的更新，并在每次移动后触发视图的更新。此外，我们还需要确保控制器能够正确地将输入事件映射到相应的 Pointer 子类上，以便进行适当的处理。通过这种设计，我们不仅能够实现对鼠标和触屏的统一控制，还能够保持代码的模块化和可扩展性。这种设计模式的应用，使得未来添加新的输入设备或修改现有逻辑变得更加简单和直接。

```
 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';
                }}};
```

10. 用 git 工具展开代码的版本管理和发布

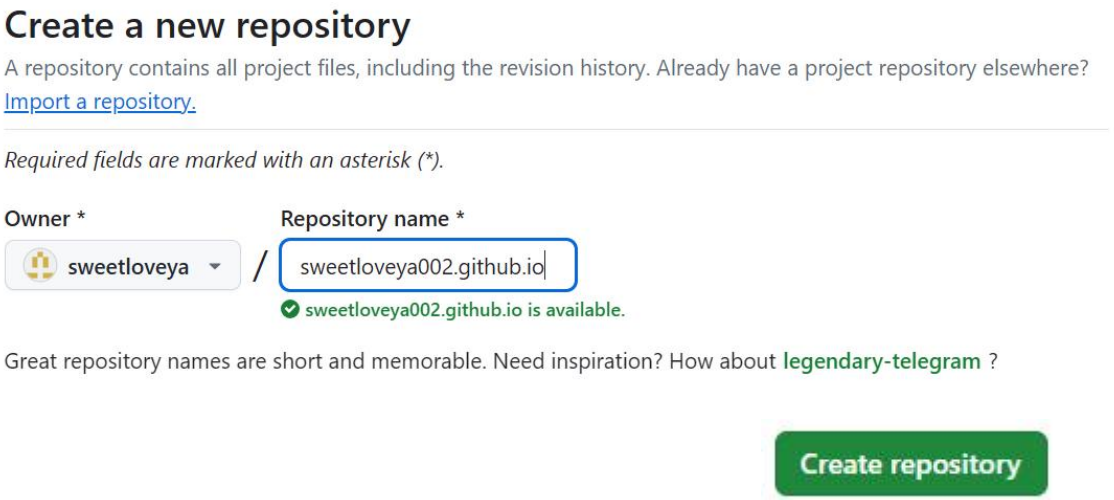
10.1 Bash 工具介绍

当我们谈到命令行时，我们实际上指的是 shell。shell 是一个程序，并将它们传递给操作系统执行。几乎所有的 Linux 发行版都提供了一个来自 GNU 项目的 shell 程序。这个名字是伯恩-再次外壳的首字母缩写，指的是 bash 是 shell 的增强替代品，最初的 Unix 外壳程序由史蒂夫·伯恩写[7]。和 Windows 一样，类似 inix 的 Linux 操作系统以

分层目录结构组织文件。这意味着它们被组织成一个树状的目录模式（在其他系统中有时称为文件夹），其中可能包含文件和其他目录。文件系统中的第一个目录称为根目录。根目录包含文件和子目录，其中包含更多的文件和子目录，等等[7]。

10.2 通过 github 平台实现本项目的全球域名

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。

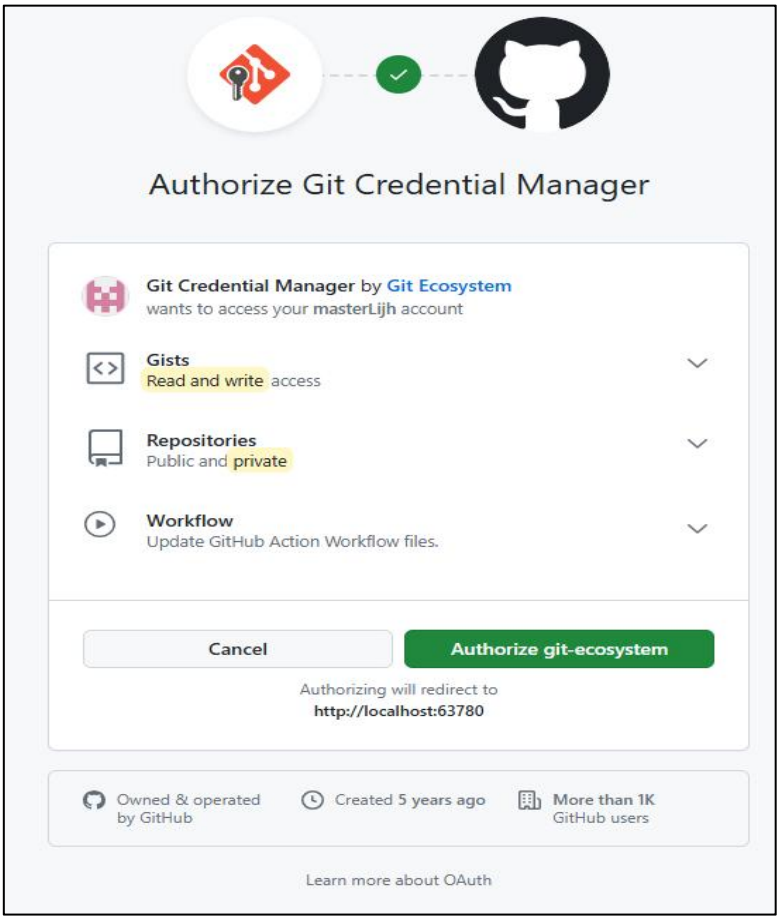
10.4 设置本地仓库和远程代码仓库的链接

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

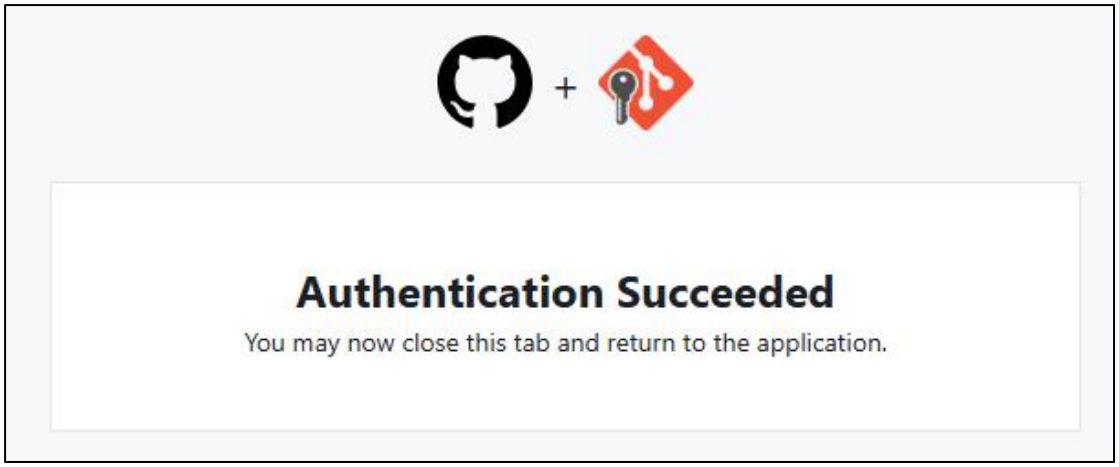
```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md
$ git init
$ git add README.md
$ git commit -m "这是我第一次把代码仓库上传至 gitHub 平台"
$ git branch -M main
$ git remote add origin
  https://sweetloveya.github.io/sweetloveya001.github.io.git
$ git push -u origin main
```

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：gitBush，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：

参考文献

[1] W3C.W3C'shistory.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