

成 绩:

江西科技师范大学

毕业论文（设计）

题目（中文）：基于 Web 客户端技术的个性化 UI 设计和实现

（外文）：Web client based UI design and code

院（系）：元宇宙产业学院

专 业：计算机科学与技术

学生姓名：欧阳朵朵

学 号：20213597

指导教师：

年 月 日

目录

基于 Web 客户端技术的个性化 UI 的设计和编程	1
(Customized UI design and Programming based on Web client technology)	1
1. 前言	1
1.1 项目概要	1
1.2 研学计划	2
1.3 研究方法	3
2. 技术总结和文献综述	3
2.1 Web 平台和客户端技术概述	3
2.2 项目的增量式迭代开发模式	5
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	8
3.4 项目的代码提交和版本管理	9
4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计	10
4.3 项目的运行和测试	13
5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI	15
5.2 项目的实现和编程	16
5.3 项目的运行和测试	19
6. 个性化 UI 设计中对鼠标交互的设计开发	20
6.1 分析与设计	20
6.2 项目的实现和编程	21
6.3 项目的运行和测试	25
7. 对触屏和鼠标的通用交互操作的设计开发	27
7.1 分析和设计	27
7.2 项目的实现和编程	27
7.3 项目的运行和测试	29
8. UI 的个性化键盘交互控制的设计开发	31
8.1 分析与设计	31
8.2 项目的实现和编程	31
8.3 项目的运行和测试	35
9. 谈谈本项目中的高质量代码	36
9.1 编程的作用	36
9.2 项目的高质量代码	37
10. 用 gitBash 工具管理项目的代码仓库和 http 服务器	38
10.1 经典 Bash 工具介绍	38
10.2 通过 GitHub 平台实现本项目的全球域名	39
10.3 创建一个空的远程代码仓库	40
10.4 设置本地仓库和远程代码仓库的链接	40
参考文献	44

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 欧阳朵朵

摘要: 在过去的十年里, 以 HTML5 为核心的 Web 标准软件开发技术, 凭借其跨平台兼容性和开源特性, 在多个领域的应用软件开发中得到了广泛应用。针对本次毕业设计任务, 我们选择了 HTML5 的 Web 客户端技术作为技术路径, 深入研究和实践了程序设计与软件开发的过程。通过系统查阅相关技术文献、开发者论坛和资料, 我们成功设计并开发了一款个性化用户界面 (UI) 的应用程序。在开发过程中, 我们综合应用了 HTML 语言进行内容构建, CSS 语言进行 UI 外观设计, 以及 JavaScript 语言实现 UI 的交互功能。此外, 我们采用了响应式设计编程, 以适应移动互联网时代用户屏幕多样化的需求。同时, 大量运用面向对象的程序设计思想, 如构建了一个通用的 pointer 模型, 该模型通过一套代码实现了对鼠标和触屏的控制, 从而保证了代码的高质量, 这也是本项目的显著特点。本项目采用了增量式开发模式, 通过逐步求精的方式进行了六次代码的重构 (包括分析、设计、实现和测试), 有效地推进了项目的设计、开发和测试工作。我们借助 Git 工具进行了版本管理, 在开发过程中重构代码六次并提交, 同时在测试阶段进行了两次代码修改和提交。利用 GitBash 工具将本项目的代码仓库上传至 GitHub 平台, 并通过 GitHub 提供的 HTTP 服务器, 实现了 UI 应用在全球互联网的部署。用户可通过地址或二维码方便地跨平台高效访问该程序。

关键词: Web 平台; github; JavaScript; HTML; CSS

1. 前言

1.1 项目概要

在软件开发与程序设计的深入研究中, 本项目选定 HTML5 的 Web 客户端技术作为核心实施路径。经过细致查阅相关领域的权威书籍、开发者交流平台和学术文献, 特别是 Mozilla 组织 MDN 社区内的技术实践指导, 我们深入掌握了 HTML 内容建模、CSS 样式编排和 JavaScript 功能编程的基础理论与应用技巧。基于这些技术, 我们成功设计并开发了一款具备个性化用户界面的 (UI) 应用程序。在开发过程中, 我们充分利用 HTML 进行页面内容的结构化建模, 通过 CSS 精细雕琢 UI 的外观呈现, 并结合 JavaScript 编程技术赋予 UI 丰富的交互功能。整个项目的代码均为手工编写, 未引入任何外部代码库或框架, 确保了代码的原生性和可维护性。此外, 项目还融入了响应式设计理念, 使得该 UI 能够灵活适应不同屏幕尺寸, 无论是 PC 端还是移动端设备, 都能获得最佳的用户体验。在功

能实现上，项目采用了 DOM 技术和事件驱动模式，实现了对鼠标、触屏、键盘等底层事件的快速响应和流畅支持。我们创新性地设计了一个指向性设备模拟模型，通过代码实现了对鼠标和触屏的精准控制，这不仅是本项目在模型研究法上的一次成功实践，也是其显著亮点之一。此外，项目还广泛运用了面向对象的程序设计思想，通过构建通用的 pointer 模型，实现了对鼠标和触屏的高效管理，进一步提升了代码的质量和可重用性。在代码管理和分享方面，本项目采用了 Git 工具进行版本控制，并在开发过程中进行了六次代码重构和正式提交。经过严格的测试与调整，额外提交了两次修改后的代码。最终，利用 Git Bash 工具，我们将项目的代码仓库上传至知名的 GitHub 平台，并成功设置为 HTTP 服务器，为全球用户提供了便捷的访问途径。

1.2 研学计划

本项目精心规划了六个迭代递增的开发阶段，旨在通过 Git Bash 工具进行项目提交，并将代码存储于 GitHub 仓库中，以开源的方式与全球开发者共享项目进展。这六个阶段涵盖了从内容构思到移动互联功能的实现，再到响应式 UI 设计的全方位开发流程。在第一阶段，我们利用 HTML 和 CSS 技术，初步构建了简洁的内容展示框架。第二阶段在此基础上，通过运用 CSS 的高级语法和 JavaScript 技术，丰富了内容的展示效果，并添加了导航区域，便于用户快速定位所需内容。进入第三阶段，我们着手实现移动互联的响应式 UI 设计。这一阶段专注于通过键盘和鼠标的响应，展示 PC 端和移动端的个性化 UI 设计，并解决了 iOS 和 Android 两大主流移动操作系统中内容展示的差异问题。接下来的三个阶段，我们专注于优化个性化 UI 设计的键鼠响应和移动互联时代的宽窄屏适配。我们不断完善和丰富 UI 设计，以满足更多用户的需求和偏好。最终利用 HTTP 服务实现全球范围内对本项目 UI 应用的便捷访问，为用户提供了更为流畅和个性化的使用体验。

1.3 研究方法

在深入研究 Web 客户端技术背景下个性化 UI 设计与实现的文献后，我们明确了用户对移动互联时代个性化 UI 设计的具体需求和期望。进而，我们对项目的可行性进行了深入的分析，并采纳了软件工程中的经典开发方法——迭代增量式开发模型。此模型将复杂的软件开发过程拆解为一系列的小步骤（迭代），每个迭代均产生具备执行能力的成果（增量），并在后续的迭代中持续优化和扩展。基于定性与定量研究收集的数据，我们进行了详尽的统计分析和数据挖掘，根据所得数据，通过迭代增量式开发模型逐步构建移动互联时代的个性化 UI 设计。最终，我们总结并归纳了 Web 客户端技术中个性化 UI 设计与实现的关键问题，并展望了未来的发展方向和创新路径，旨在为相关领域的研究和实践提供有价值的启示和指导。

2. 技术总结和文献综述

2.1 Web 平台和客户端技术概述

自 Web 的奠基人 Tim Berners-Lee 创建 Web 的基础技术架构后，他成立了 W3C 组织，该组织在 2010 年后推出的 HTML5 国际标准与欧洲 ECMA 组织维护的 ECMAScript 标准相结合，为全球开发者迈向统一开发平台奠定了坚实基础。这一伟大的目标至今仍激励着科学家与 Web 行业不断探索和完善^[1]。我的毕业设计项目便是基于学习 Web 标准与技术，通过编写 Web 程序与相关工具，致力于构建一套高质量、跨平台运行的代码应用。

回溯至 1989 年，为了促进 CERN（位于瑞士日内瓦的欧洲粒子物理实验室）的合作研究，蒂姆·伯纳斯-李提出了在论文中嵌入“超文本链接”的构想，使得文献间的引用可以通过点击链接快速跳转。从 1989 年至 1991 年，伯纳斯-李取得了多项重要成果：他设计了 HTML，其中超文本链接成为核心特性；他构思了万维网背后的概念，包括 HTTP 协议；他还开发了一个基于 HTML 的互联网浏览器原型。1993 年，他与丹康·诺利共同向互联网工程任务组 (IETF) 递交了首个

关于 HTML 的正式提案。随后在 1994 年，蒂姆·伯纳斯-李在麻省理工学院创建了万维网联盟(W3C)，并承担起 HTML 标准的管理工作。他编写了 HTML 的第一个版本，这种具备超文本链接功能的文档格式语言，成为了 Web 的主要发布格式。随着 Web 技术的推广，他对 URI、HTTP 和 HTML 的原始规范进行了深入的改进和讨论。W3C 的成立标志着 Web 正式进入规范化发展阶段，其宗旨在于推动网络技术的统一和标准化，确保网络信息的广泛可访问性和兼容性。早期，W3C 主要聚焦于 HTML 等网页基础技术的标准制定，随着互联网的飞速发展，其工作范畴逐渐扩展至 CSS、XML、DOM、SVG、网页无障碍性、网络服务以及 HTML5 等众多领域。迄今为止，W3C 已发布数百项具有深远影响的 Web 技术标准与实施指南，极大地推动了互联网技术的进步与应用的普及。W3C 的成员组成也日益国际化，由最初的几家创始机构扩展到涵盖全球 40 多个国家的 400 多个会员组织，并在全球多地设立办事处，展现了其作为国际性标准组织的广泛影响力和中立性。W3C 通过开放的标准制定过程，鼓励多方参与，确保网络技术的持续创新与健康发展。

Web，即万维网的缩写，通常简称为“Web”。它是文档的集合，称为网页，由世界各地的计算机用户共享。这些网页类型各异，功能多样，但共同之处在于它们都在电脑屏幕上展示内容。这些“内容”包括文本、图片以及用户输入机制，如文本框和按钮^[2]。处理这些内容的技术多种多样，如 HTML（用于定义网页结构和意义）、CSS（用于描述网页外观和布局）、JavaScript（用于实现网页的动态功能和交互性）、DOM（使 JavaScript 能够操作网页内容、结构和样式）以及 Web APIs（如 Fetch API 用于数据获取、Web Storage 用于客户端存储、Web Workers 用于后台处理等），这些技术极大地丰富了 Web 应用的功能。Web 编程是一个广阔的领域，涵盖了多种类型的编程工具。这些工具都以 HTML 为核语言，因此几乎所有的 Web 编程书籍都会在一定程度上介绍 HTML。这些书籍通常深入讲解 HTML5、CSS 和 JavaScript，这三种技术被认为是客户端 Web 编程的基石。在客户端 Web 编程中，所有的计算都在最终用户的计算机（即客户端计算机）上执行

^[3]。

2.2 项目的增量式迭代开发模式

本项目作为一个本科阶段毕业设计的软件作品，其复杂程度远超单一用途的程序，其涉及的手写代码量显著超越了一两个数量级的范畴。从问题的深入分析到代码的初步编写，这一过程无法在短暂数日内仓促完成，它实际上是一个系统工程的体现。因此，我们需要从软件工程的管理角度出发，严谨地规划和执行项目的编写过程。

针对本项目的特点，我们考量了两种经典的软件工程开发管理模式：瀑布模型和增量式迭代模型。不论选择何种模式，都需历经分析、设计、实施和测试这四个核心阶段^[4]。瀑布模型强调专业团队的高度协作，要求每一阶段的开始都必须基于前一阶段的完美结束。然而，对于多数普通开发者，特别是身兼数职的小微开发者而言，这种一次性完美完成各阶段工作的要求显得不切实际。在实施过程中，开发者可能会发现前期设计存在不足，需要在后续迭代中进行优化。

在当今开源软件开发的大环境下，开发者通常会在开发过程中持续优化设计、重构代码，以不断改进程序的功能和代码质量。因此，在本项目的实际开发中，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 所示：

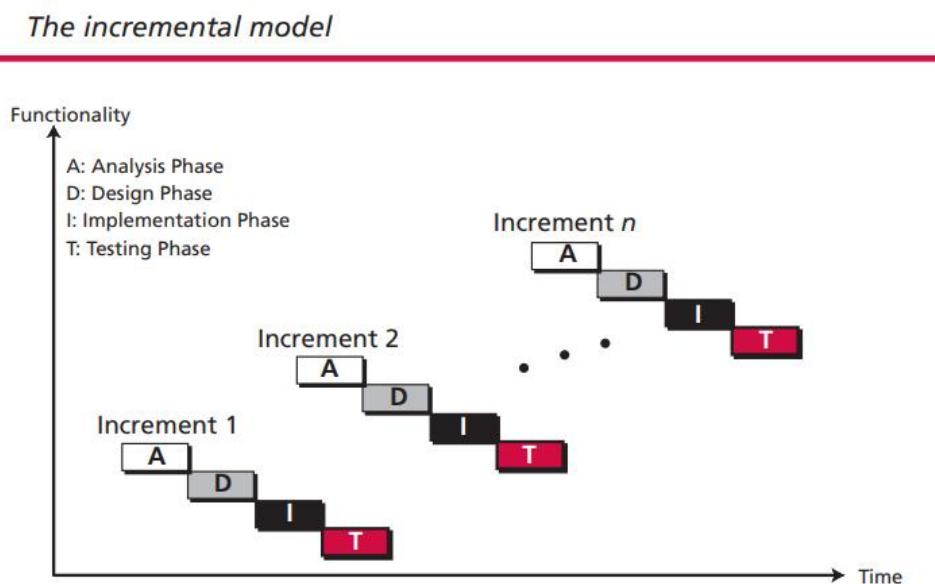


图 2-1 增量迭代开发模式

在软件开发过程中，开发人员遵循一系列精心设计的步骤。首先，开发人员构建了一个系统的简化版本，这个版本大致勾勒出整体架构，但尚未包含详尽的

细节。这一简化的版本旨在直观地展示增量模型的基本概念。随着开发进程的推进，开发者进入了第二个版本的开发阶段，此时加入了更多的功能细节，并对尚未完善的部分进行了补充。在每次系统更新后，都会再次进行全面测试。若发现问题，开发人员能够迅速定位到新增功能中的潜在问题。在现有系统稳定运行之前，开发者不会急于添加新的功能。这一迭代过程将持续进行，直至所有预期的功能均被完整且稳定地集成到系统中。

在如今高度开源的软件开发环境中，优化设计和重构代码成为了常态，开发者们始终致力于提升程序的功能性和代码质量。因此，在本项目的开发过程中，采用了增量模型的开发模式。经过六次精心的迭代和优化，本项目最终得以完成。

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。

3.2 项目的实现和编程

本项目第一阶段以“三段论”方式展开的内容代码在第一部分标题区中使用 HTML+CSS 来达到项目的实现，因为这是项目的初始阶段，我们使用简单的语句来初步介绍我们想要表达的内容，同时作为移动移动互联设计的第一阶段，我们将在 pc 端和移动端同步实现项目的初始展示，其次在项目的代码提交和版本管理上，我们利用 gitBash 工具进行项目的管理。

HTML 代码编写如代码截图 3-1：

```
<body>
  <header>
    《 欧阳朵朵的毕设题目 》
  </header>
  <main>
    我的主题内容：‘读好书、练思维、勤编程’@masterLijh 计算思维系列课程
  </main>
  <footer>
    CopyRight from 欧阳朵朵 江西科技师范大学 2024--2025
  </footer>
</body>
```

代码截图 3-1

CSS 代码编写如代码截图 3-2：

```
1 *{
2   margin: 10px;
3   text-align: center;
4   font-size: 30px ;
5 }
6 header{
7   border: 2px solid blue;
8   height: 200px;
9 }
10
11 main{
12   border: 2px solid blue;
13   height: 400px;
14 }
15
16 footer{
17   border: 2px solid blue;
18   height: 100px;
19 }
20 a{
21   display: inline-block ;
22   padding: 10px ;
23   color: white;
24   background-color: blue;
25   text-decoration: none ;
26 }
```

代码截图 3-2

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 Chrome 浏览器打开项目的结果，如下图 3-3 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 3-4 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 3-3 PC 端运行效果图



图 3-4 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：如开发者的姓名和电子邮件。进入 gitBash 命令行后，按次序输入以下命令：

```
HP@C MINGW64 /webUI (master)
$ git config user.name 科师大欧阳朵朵

HP@C MINGW64 /webUI (master)
$ git config user.email 2794024923@qq.com

HP@C MINGW64 /webUI (master)
$ touch index.html myCss.css
```

测试运行成功后，执行下面命令提交代码：

```
HP@C MINGW64 /webUI (master)
$ git add index.html myCss.css

HP@C MINGW64 /webUI (master)
$ git commit -m 项目第一版：“三段式”的内容设计概要开发
```

成功提交代码后，gitbash 的反馈如下所示：

```
HP@C MINGW64 /webUI (master)
$ git commit -m 项目第一版：“三段式”的内容设计概要开发
[master (root-commit) 6834c32] 项目第一版：“三段式”的内容设计概要开发
 2 files changed, 0 insertions(+), 0 deletions(-)
  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：

```
HP@C MINGW64 /webUI (master)
$ git log
```

gitbash 反馈代码的仓库日志如下所示：

```
HP@C MINGW64 /webUI (master)
$ git log
commit 6834c32d74f48fc05eab7938efca38efa49743a2 (HEAD -> master)
)
Author: 科师大欧阳朵朵 <2794024923@qq.com>
Date:   Tue Jun 18 17:45:49 2024 +0800
```

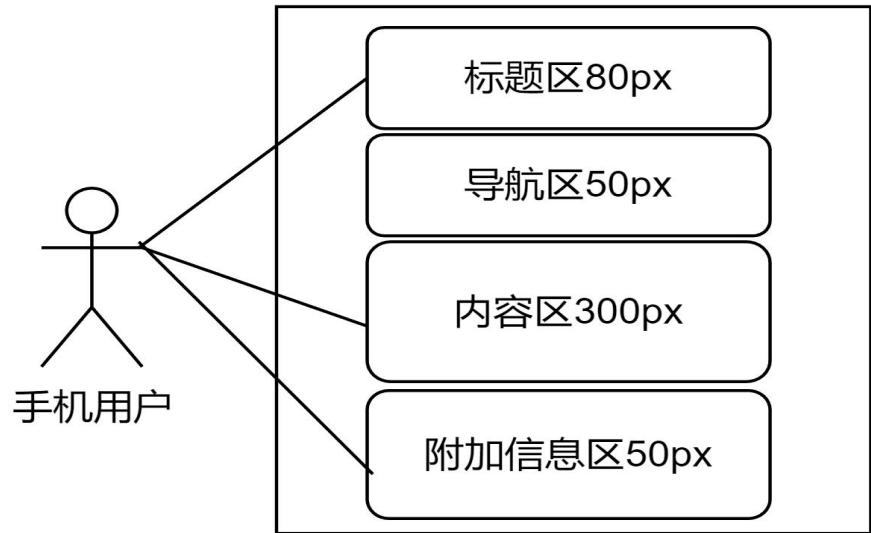
项目第一版：“三段式”的内容设计概要开发

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 分析和设计

鉴于计算机显示器硬件的显著多样性，品牌各异或类型不同的显示器在尺寸和分辨率上均存在显著差异。因此设计师在网页布局上并未采用固定版本，而是采取了一种灵活的策略，即提供总体布局指南，同时浏览器可以根据特定计算机硬件自主调整页面。例如，网页作者可以定义一组句子构成一个段落，但具体到诸如行宽或段落首行缩进等细节，则由浏览器根据具体情况自行判断。然而这种浏览器自主调整布局的做法也可能带来一个有趣的现象：当同一网页在不同浏览器或硬件各异的计算机上查看时，其外观可能呈现差异。若屏幕宽度不同，文本行长或图像显示尺寸便会有所变化。关键在于网页为所需呈现的内容提供了大致的框架和指引，而浏览器则在实际展示时根据具体情况进行微调。因此当同一网页在不同计算机上呈现或经由不同显示设备浏览时，用户可能会观察到细微的视觉效果差异。

基于这一认识，我们在设计的第二阶段进行了进一步的优化。我们利用 JavaScript 动态读取显示设备的参数信息，并结合 CSS 技术精准地部署适配当前设备的代码以调整页面宽度和布局，从而实现对不同设备的完美适配。这种设置不仅适用于 PC 端，同样能够确保移动端设备获得理想的页面展示效果。无论用户使用的是何种型号、何种品牌的手机或电脑，都能确保页面呈现出合理且理想的视觉效果。用例图如下：



4.2 项目的实现和编程

HTML 代码编写如代码截图 4-1：

```

<header>
  <p id="book">
    我的毕设题目</p>
</header>
<nav>
  <button>导航一</button>
  <button>导航二</button>
  <button>导航三</button>
</nav>
<main id = 'main'>
  软件内容区域
</main>
<footer>
  <p id="statusInfo">
    欧阳朵朵 江西科技师范大学 2025</p>
</footer>
```

代码截图 4-1

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 css 代码 4-2 所示：

```
28 <style>
29 *{
30   margin: 10px;
31   text-align: center;
32 }
33
34 header{
35   border: 2px solid blue;
36   height: 15%;
37   font-size: 1.66em;
38 }
39 }
40 main{
41   border: 2px solid blue;
42   height: 70%;
43   font-size: 1.2em;
44 }
45 }
46 nav{
47   border: 2px solid blue;
48   height: 10%;
49 }
50 nav button{
51   font-size: 1.1em;
52 }
53 footer{
54   border: 2px solid blue;
55   height: 5%;
56 }
57 </style>
```

代码截图 4-2

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^[6]。如 js 代码截图 4-3 所示：

```
1 <script>
2     var UI = {};
3     UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;
4     UI.appHeight = window.innerHeight;
5     const LETTERS = 22 ;
6     const baseFont = UI.appWidth / LETTERS;
7
8 //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
9     document.body.style.fontSize = baseFont + "px";
10 //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。
11 //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。
12     document.body.style.width = UI.appWidth - 2*baseFont + "px" ;
13     document.body.style.height = UI.appHeight - 4*baseFont + "px";
14 </script>
```

代码截图 4-3

4.3 项目的运行和测试

本次阶段的项目运行和测试是测试在移动端设备的显示，以确保可以有效地满足移动互联时代的响应式设计的需求，如图 4-4 移动端展示图如下图所示：

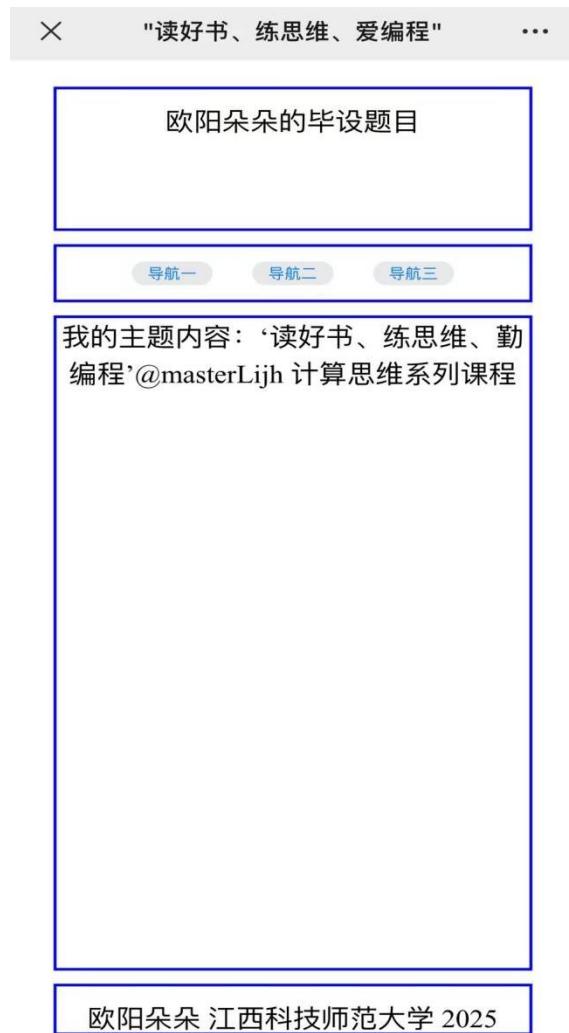


图 4-4 移动端展示图

移动端用户可以通过扫描图 4-5 的二维码，运行测试本项目的第二次开发的阶段性效果。



<https://rabaesther.github.io/RabaEsther/1-2.html>

图 4-5 二维码

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.1 分析与设计

在移动互联时代，用户终端的多样性显著，涵盖了多样化的设备类型、屏幕尺寸、分辨率、操作系统及浏览器等。用户可能通过智能手机、平板电脑、笔记本电脑、台式电脑乃至智能手表等设备访问网站或应用程序。为应对这种多样性，本次项目更新特别引入了鼠标和键盘的交互支持。为了确保用户在不同终端设备上均能享受到优质的用户体验，响应式设计成为了一项关键技术。以下是对响应式设计的详细分析与设计阐述，最开始我们根据当前浏览器窗口的宽度来动态调整应用界面的宽度。若窗口宽度超过 600 像素，则应用界面将自动调整为 600 像素的宽度；反之，则保持与窗口宽度一致。接下来我们获取当前窗口的高度，并据此计算出基础字体大小。基于这一基础字体大小，我们利用 JavaScript 来动态调整页面的 CSS 样式。页面的字体大小将直接设置为计算得到的基础字体大小；页面的宽度将设定为应用界面宽度减去两倍的基础字体大小；而页面的高度则减去 8 倍的基础字体大小。通过这一系列的调整，我们能够确保页面在不同设备上呈现出最佳的视觉效果和用户体验。

如图 5-1 用例图所示：

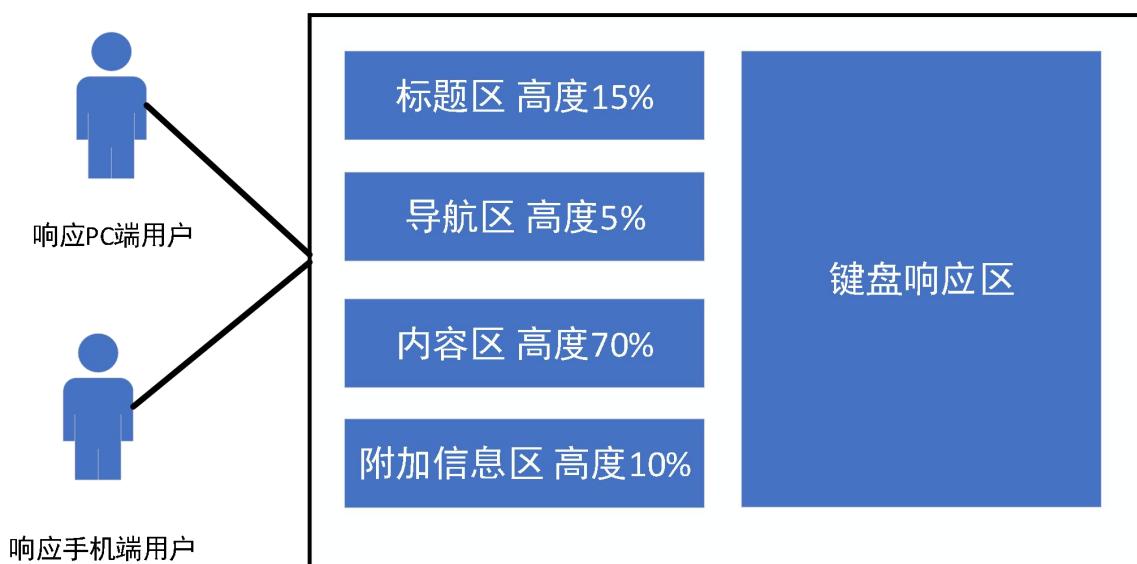


图 5-1 用例图

5.2 项目的实现和编程

HTML 代码截图如下代码截图 5-2 所示:

```
<body>
  <header>
    <p id="book">
      欧阳朵朵的毕设题目</p>
  </header>
  <nav>
    <button>导航1</button>
    <button>导航2</button>
    <button>导航3</button></nav>
  <main id="main">
    <div id="bookface" >
      我的主题内容: ‘读好书、练思维、勤编程’@masterLijh 计算思维系列课程
    </div>
  </main>
  <footer>
    CopyRight 欧阳朵朵 江西科技师范大学 2024--2025</footer>
```

代码截图 5-2

Css 代码截图下代码截图 5-2 所示:

```
1 <style>
2   *{
3     text-align: center;
4     box-sizing: border-box ;
5   }
6   header,main,div#bookface,nav,footer{
7     margin:1em;
8   }
9   header{
10    border: 2px solid blue;
11    height: 15%;
12    font-size: 1.66em;
13  }
14  main{
15    border: 2px solid blue;
16    height: 70%;
17    font-size: 1.2em;
18  }
19  }
20  nav{
21    border: 2px solid blue;
22    height: 10%;
23  }
24  }
25  nav button{
26    font-size: 1.1em;
27  }
28  footer{
29    border: 2px solid blue;
30    height: 5%;
31  }
32  body{
33    position:relative ;
34  }
35  #aid{
36    position: absolute;
37    border: 3px solid blue;
38    top: 0.5em;
39    left: 600px;
40  }
41  #bookface{
42    width: 80%;
43    height: 80%;
44    border:1px solid red;
45    background-color: blanchedalmond;
46    margin:auto;
47  }
48 </style>
49
```

代码截图 5-2

与上一阶段比较，本阶段使用了 JavaScript，最新创建了一个 mouse 对象以及 keypress 对象，可以初步地接收鼠标按下地坐标以及接收用户键盘按下的按键及其对应的编码，最后再利用动态 CSS，实现了软件界面的全屏设置，如 js 代码截图 5-3 所示：

```

1 <script>
2   var UI = {};
3   UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;
4   UI.appHeight = window.innerHeight;
5   const LETTERS = 22 ;
6   const baseFont = UI.appWidth / LETTERS;
7
8 //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
9   document.body.style.fontSize = baseFont + "px";
10 //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。
11 //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。
12   document.body.style.width = UI.appWidth - 2*baseFont + "px" ;
13   document.body.style.height = UI.appHeight - 8*baseFont + "px";
14
15 if(window.innerWidth < 900){
16   $('#aid').style.display='none';
17 }
18 $('#aid').style.width=window.innerWidth - UI.appWidth - 2*baseFont + 'px';
19 $('#aid').style.height= document.body.clientHeight + 'px';
20
21 //尝试对鼠标设计UI控制
22 var mouse={};
23 mouse.isDown= false;
24 mouse.x= 0;
25 mouse.deltaX=0;
26 $('#bookface').addEventListener("mousedown",function(ev){
27   let x= ev.pageX;
28   let y= ev.pageY;
29
30   console.log("鼠标按下了，坐标为: "+("+"x+","+"y+"));
31   $('#bookface').textContent= "鼠标按下了，坐标为: "+("+"x+","+"y+");
32 });
33
34 $('#bookface').addEventListener("mousemove",function(ev){
35   let x= ev.pageX;
36   let y= ev.pageY;
37
38   console.log("鼠标正在移动，坐标为: "+("+"x+","+"y+"));
39   $('#bookface').textContent= "鼠标正在移动，坐标为: "+("+"x+","+"y+");
40 });
41 $('#bookface').addEventListener("mouseout",function(ev){
42   let x= ev.pageX;
43   let y= ev.pageY;
44
45   console.log("鼠标按下了，坐标为: "+("+"x+","+"y+"));
46   $('#bookface').textContent= "鼠标离开了，坐标为: "+("+"x+","+"y+");
47   $('#bookface').textContent="鼠标已经离开";
48 });
49 $('body').addEventListener("keypress",function(ev){
50   let k = ev.key;
51   let c = ev.keyCode;
52   $('#keyboard').textContent = "您的按键 : " + k + " , "+ "字符编码 : " + c;
53 });
54
55 function $(ele){
56   if (typeof ele !== 'string'){
57     throw("自定义的$函数参数的数据类型错误，实参必须是字符串！");
58     return
59   }
60   let dom = document.getElementById(ele) ;
61   if(dom){
62     return dom ;
63   }else{
64     dom = document.querySelector(ele) ;
65     if (dom) {
66       return dom ;
67     }else{
68       throw("执行$函数未能在页面上获取任何元素，请自查问题！");
69       return ;
70     }
71   }
72 } //end of $
73

```

代码截图 5-3

5.3 项目的运行和测试

此次测试主要针对 PC 端设备和手机端进行鼠标交互以及键盘交互的功能性测试，但由于手机端屏幕大小没有超过 600，所以无法显示出手机端的键盘交互区页面，PC 端如下图 5-4 所示，手机端如下图 5-5 所示，移动端用户可以通过扫描图 5-6 的二维码，运行测试本项目的第三次开发的阶段性效果。

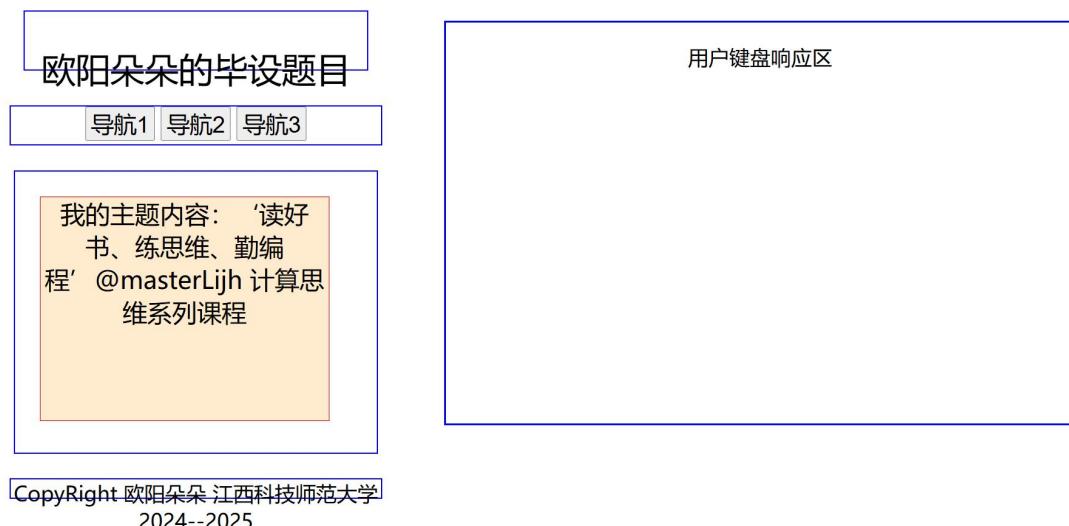


图 5-4 PC 端测试图

欧阳朵朵的毕设题目

导航1 导航2 导航3

我的主题内容：‘读好书、
练思维、勤编
程’@masterLjh 计算思维
系列课程

CopyRight 欧阳朵朵 江西科技师范大学
2024--2025

<https://rabaesther.github.io/RabaEsther/1-3.html>

图 5-5 手机端界面图

图 5-6 移动端二维码

6. 个性化 UI 设计中对鼠标交互的设计开发

6.1 分析与设计

在 UI 中尝试对鼠标设计控制，设计模拟手机端的触屏效果，设置触屏条件横向移动 100px 为有效拖动，否则为无效拖动。使用鼠标按下、松开、移动模拟手指横向滑动屏幕的操作，添加 Eventlistener 实现 mouseup、mousedown、mouseout 以及 mousemove 的功能。鼠标移动时会显示正在拖动鼠标和拖动距离，当鼠标横

向拖动距离大于 100px 显示“鼠标松开！这次是有效拖动！”，拖动距离小于 100px 显示“鼠标松开！这次是无效拖动！”。

6.2 项目的实现和编程

以下代码截图是对 mouseup 和 mousedown 的实现，当鼠标按下时，在控制台处显示鼠标按下位置的坐标，当鼠标松开时，通过判断鼠标移动距离，确认鼠标拖动是否有效。

HTML 代码编写如代码截图 6-1：

```
<body>
  <header>
    <p id="book">
      《我的毕设题目》</p>
  </header>
  <nav>
    <button>向前</button>
    <button>向后</button>
    <button>其他</button>
  </nav>
  <main id="main">
    <div id="bookface">
      这是书的封面图<br>
      在此对象范围拖动鼠标(本例触屏无效)
    </div>
  </main>
  <footer>
    CopyRight 欧阳朵朵 江西科技师范大学 2024--2025
  </footer>
  <div id="aid">
    <p>用户键盘响应区</p>
    <p id="keyboard"></p>
  </div>
```

截图 6-1

Css 代码截图 6-2 所示：

```
<style>
*{
    margin: 10px;
    text-align: center;
}
header{
    border: 3px solid green;
    height: 10%;
    font-size: 1em;
}
nav{
    border: 3px solid green;
    height: 10%;
}
main{
    border: 3px solid green;
    height: 70%;
    font-size: 0.8em;
    position: relative;
}
#box{
    position: absolute;
    right: 0;
    width: 100px;
}
footer{
    border: 3px solid green;
    height:10%;
    font-size: 0.7em;
}
#aid{
    position: absolute;
    border: 3px solid blue;
    top:0px;
    left:600px;
}
#bookface{
    position: absolute;
    width: 80%;
    height: 80%;
    border:1px solid red;
    background-color: blanchedalmond;
    left:7% ;
    top: 7% ;
}
</style>
```

代码截图 6-2

js 代码截图 6-3 所示：

```

1 <script>
2   var UI = {};
3   if(window.innerWidth>600){
4     UI.appWidth=600;
5   }else{
6     UI.appWidth = window.innerWidth;
7   }
8
9   UI.appHeight = window.innerHeight;
10
11 let baseFont = UI.appWidth /20;
12 //通过改变body对象的字体大小，这个属性可以影响其后代
13 document.body.style.fontSize = baseFont + "px";
14 //通过把body的高度设置为设备屏幕的高度，从而实现纵向全屏
15 //通过CSS对子对象百分比（纵向）的配合，从而达到我们响应式设计的目标
16 document.body.style.width = UI.appWidth - baseFont + "px";
17 document.body.style.height = UI.appHeight - baseFont*4 + "px";
18 if(window.innerWidth<1000){
19   $("aid").style.display='none';
20 }
21 $("aid").style.width=window.innerWidth-UI.appWidth - baseFont*3 +'px';
22 $("aid").style.height= UI.appHeight - baseFont*3 +'px';
23
24 //尝试对鼠标设计UI控制
25 var mouse={};
26 mouse.isDown= false;
27 mouse.x= 0;
28 mouse.y= 0;
29 mouse.deltaX=0;
30 $("bookface").addEventListener("mousedown",function(ev){
31   mouse.isDown=true;
32   mouse.x= ev.pageX;
33   mouse.y= ev.pageY;
34   console.log("mouseDown at x: "+("+"( "+mouse.x +"," +mouse.y +"") ) ;
35   $("bookface").textContent= "鼠标按下，坐标: "+("+"( "+mouse.x+"," +mouse.y+"") );
36 });

```

```

$( "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' ;
    }
});

70
71 });
72
73     function $(ele){
74         if (typeof ele !== 'string'){
75             throw("自定义$函数参数的数据类型错误，实参必须是字符串！");
76             return
77         }
78         let dom = document.getElementById(ele) ;
79         if(dom){
80             return dom ;
81         }else{
82             dom = document.querySelector(ele) ;
83             if (dom) {
84                 return dom ;
85             }else{
86                 throw("执行$函数未能在页面上获取任何元素，请自查问题！");
87                 return ;
88             }
89         }
90     }
91 //end of $
92 </script>

```

代码截图 6-3

6.3 项目的运行和测试

本次测试主要测试 PC 端的封面移动用例，如下图 6-4 所示：

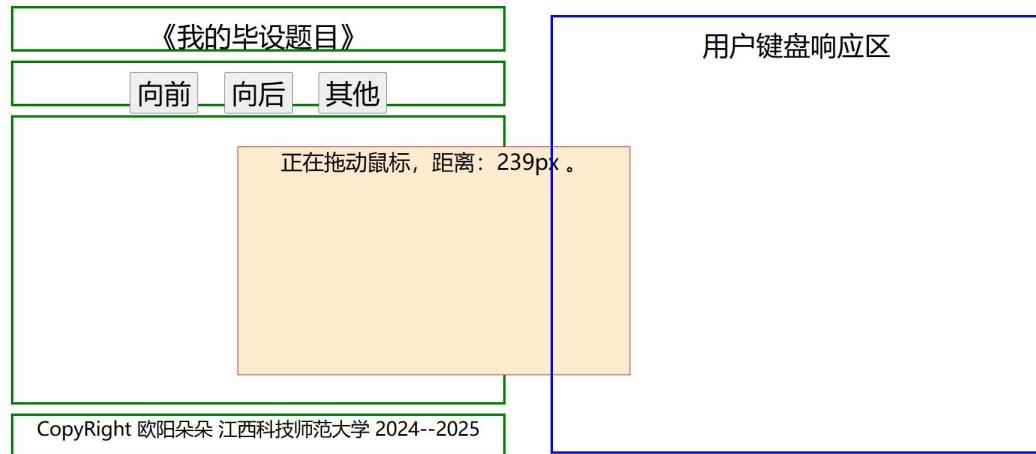


图 6-4 鼠标模型拖动示例图

移动端用户示例如图 6-5 所示：



图 6-5

可见，由于该次项目针对 pc 端，可见在移动端并不能有效拖动。

二维码如图 6-6 所示：



<https://rabaesther.github.io/RabaEsther/1-4.html>

图 6-6 项目二维码

7. 对触屏和鼠标的通用交互操作的设计开发

7.1 分析和设计

当点击鼠标或触屏时显示鼠标在方框内的具体坐标，并在鼠标或触屏拖动时显示拖动是否有效，当拖动有效时显示鼠标或触屏的拖动到拖动距。

创建 Pointer 实现对鼠标和触屏设计一套代码实现 UI 控制。在 bookface 中添加 handleBegin, handleEnd, handleMoving 三个 EventListener。首先判断操作为触屏还是鼠标操作，并进行相应的操作反馈。

7.2 项目的实现和编程

添加 EventListener，对 handleBegin 的实现，当点击鼠标或触屏时，显示对应的坐标。对 handleEnd 功能的实现，显示鼠标或触屏的拖动操作，当拖动距离超过 100 时，拖动无效。对 handleMoving 功能的实现，显示鼠标及触屏拖动操作时的移动距离。因为本次迭代没有对 HTML 代码和 CSS 代码进行修改，所以

此次代码展示只提交 js 代码，js 代码如代码截图 7-1 所示：

```
<script>
    var UI = {};
    if(window.innerWidth>600){
        UI.appWidth=600;
    }else{
        UI.appWidth = window.innerWidth;
    }

    UI.appHeight = window.innerHeight;

    let baseFont = UI.appWidth /20;
    //通过改变body对象的字体大小，这个属性可以影响其后代
    document.body.style.fontSize = baseFont +"px";
    //通过把body的高度设置为设备屏幕的高度，从而实现纵向全屏
    //通过CSS对子对象百分比（纵向）的配合，从而达到我们响应式设计的目标
    document.body.style.width = UI.appWidth - baseFont + "px";
    document.body.style.height = UI.appHeight - baseFont*4 + "px";
    if(window.innerWidth<1000){
        $("#aid").style.display='none';
    }
    $("#aid").style.width=window.innerWidth-UI.appWidth - baseFont*3 +'px';
    $("#aid").style.height= UI.appHeight - baseFont*3 +'px';

    //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 ;
});

} //Code Block end
function $(ele){
    if (typeof ele !== 'string'){
        throw("自定义的$函数参数的数据类型错误，实参必须是字符串！");
        return
    }
    let dom = document.getElementById(ele) ;
    if(dom){
        return dom ;
    }else{
        dom = document.querySelector(ele) ;
        if (dom) {
            return dom ;
        }else{
            throw("执行$函数未能在页面上获取任何元素，请自查问题！");
            return ;
        }
    }
} //end of $

</script>
```

代码截图 7-1

7.3 项目的运行和测试

此次功能测试主要使正对手机端用户，可以进行横向触屏操作功能，显示触屏坐标，如下图 7-1、7-2 所示：



图 7-1 显示触屏坐标测试图

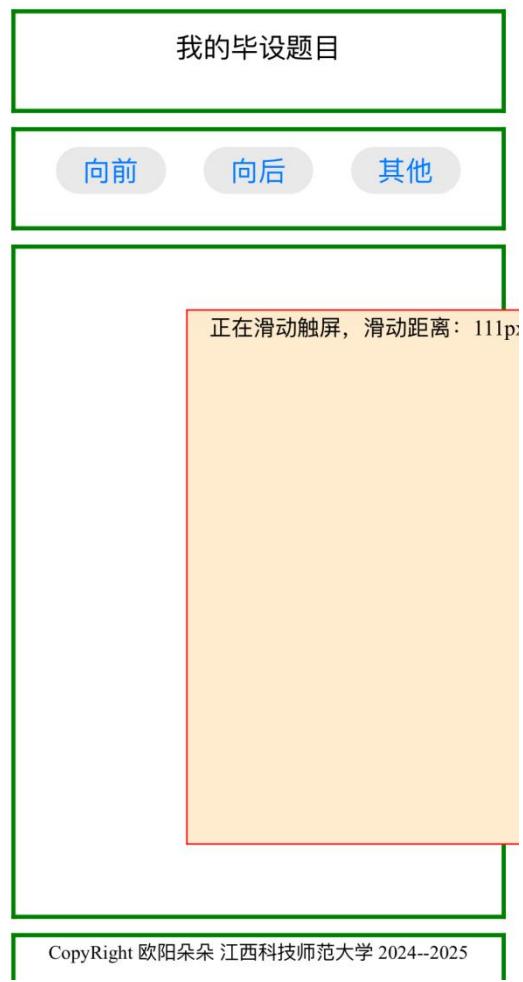


图 7-2 显示触屏拖动距离测试图

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 7-3 移动端二维码

8. UI 的个性化键盘交互控制的设计开发

8.1 分析与设计

当敲击键盘和松开键盘时对所按按键的键值和对应代码输出出来。添加两个 EventListener，利用 keydown 和 keyup 两个底层事件，实现同时输出按键状态和文本内容。

8.2 项目的实现和编程

由于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。

HTML 代码编写如代码截图 8-1 所示：

```
<body>
  <header>
    <p id="book">
      我的毕设题目</p></header>
  <nav>
    <button>向前</button>
    <button>向后</button>
    <button>其他</button> </nav>
  <main id="main">
    <div id="bookface">
      这是书的封面图<br>
      在此对象范围拖动鼠标/滑动触屏<br>
      拖动/滑动超过100像素，视为有效UI互动！ </div></main>
  <footer>
    CopyRight 欧阳朵朵 江西科技师范大学 2024--2025</footer>
  <div id="aid">
    <div id="aid">
      用户键盘响应区
      <p id="typeText"></p>
      <hr>
      <p id="keyboard"></p>
    </div>
  
```

代码截图 8-1

Css 代码截图下代码截图 8-2 所示：

```

1 <style>
2 *{
3   margin: 10px;
4   text-align: center;
5 }
6 header{
7   border: 3px solid green;
8   height: 10%;
9   font-size: 1em;
10}
11}
12 nav{
13   border: 3px solid green;
14   height: 10%;
15}
16 main{
17   border: 3px solid green;
18   height: 70%;
19   font-size: 0.8em;
20   position: relative;
21}
22
23 #box{
24   position: absolute;
25   right: 0;
26   width: 100px;
27}
28
29 footer{
30   border: 3px solid green;
31   height:10%;
32   font-size: 0.7em;
33}
34
35 body{
36   position: relative;
37 }
38 button{
39   font-size:1em;
40 }
41 #aid{
42   position: absolute;
43   border: 3px solid blue;
44   top:0px;
45   left:600px;
46 }
47 #bookface{
48   position: absolute;
49   width: 80%;
50   height: 80%;
51   border:1px solid red;
52   background-color: blanchedalmond;
53   left:7% ;
54   top: 7% ;
55 }
56

```

代码截图 8-2

添加两个 EventListener，keydown 显示按键是的键值和字符编码，keyup 显示松开按键时的键值和字符编码，添加功能 printLetter(k) 确保只有一个按键。js 代码截图如代码截图 8-3 所示：

```

<script>
    var UI = {};
    if(window.innerWidth>600){
        UI.appWidth=600;
    }else{
        UI.appWidth = window.innerWidth;
    }

    UI.appHeight = window.innerHeight;

    let baseFont = UI.appWidth /20;
    //通过改变body对象的字体大小，这个属性可以影响其后代
    document.body.style.fontSize = baseFont +"px";
    //通过把body的高度设置为设备屏幕的高度，从而实现纵向全屏
    //通过CSS对子对象百分比（纵向）的配合，从而达到我们响应式设计的目标
    document.body.style.width = UI.appWidth - baseFont + "px";
    document.body.style.height = UI.appHeight - baseFont*5 + "px";
    if(window.innerWidth<1000){
        $("aid").style.display='none';
    }
    $("aid").style.width=window.innerWidth-UI.appWidth - baseFont*3 +'px';
    $("aid").style.height= UI.appHeight - baseFont*3 +'px';

    //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("keydown",function(ev){
ev.preventDefault() ;
    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){
ev.preventDefault() ;
    let key = ev.key;
    let code = ev.keyCode;
    $("keyboard").textContent = "松开按键 : " + key + " , "+ "字符编码 : " + code;
    if (printLetter(key)){
        $("typeText").textContent += key ;
    }
    function printLetter(k){
if (k.length > 1){
    return false ;
}
let puncs = ['~','`','!','@','#','$','%','^','&','*','(',')','-','_','+','=','`','`','<','>','?','/','`'];
    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 ;
}
});
}
}

```

```

        }
    function $(ele){
        if (typeof ele !== 'string'){
            throw("自定义的$函数参数的数据类型错误，实参必须是字符串！");
            return
        }
        let dom = document.getElementById(ele) ;
        if(dom){
            return dom ;
        }else{
            dom = document.querySelector(ele) ;
            if (dom) {
                return dom ;
            }else{
                throw("执行$函数未能在页面上获取任何元素，请自查问题！");
                return ;
            }
        }
    }

</script>

```

代码截图 8-3

8.3 项目的运行和测试

本次项目测试主要就是测试 keydown 和 keyup 的功能，测试图如下图 8-4 所示。



图 8-4 keydown 功能测试图

移动端用户可以通过扫描图 8-5 的二维码，运行测试本项目的第六次开发的阶段性效果。



<https://rabaesther.github.io/RabaEsther/1-6.html>

图 8-5 手机端二维码

9. 谈谈本项目中的高质量代码

9.1 编程的作用

在当下社会，电脑已如同螺丝刀般普及，但其内在机制却相当复杂，使得指挥其执行特定任务并非总是易事。当面对的是日常且易于理解的计算机任务，如查阅电子邮件或使用计算器功能时，用户可以直接启动相应的应用程序来轻松完成。然而，当面临独特或开放性的任务时，可能并无现成的应用程序可供使用，这时编程的作用便显得尤为重要。编程本质上即是创建程序的过程，它涉及一组明确的指令，用以指导计算机执行特定操作。由于计算机本身缺乏自主思考和创新能力，编程工作往往显得单调乏味且挑战重重。然而，若能克服这些困难，并享受其中蕴含的逻辑推理过程，编程的价值便得以体现。编程能够极大地提升工作效率，使原本需要人工长时间处理的任务在数秒内即可完成。此外，它还能使电脑执行之前无法完成的任务，为用户提供更为丰富的功能体验。更重要的是，编程为锻炼抽象思维能力提供了绝佳的平台。

9.2 项目的高质量代码

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。面向对象思想，封装，抽象，局部变量，函数式编程，逻辑。

以下是实现代码：

```
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+")");
    }
};
```

图 9-1 用 Pointer 对象实现鼠标模型

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

图 9-2 鼠标拖动以及触屏操作代码详情图

```

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' ;
        }
    }
};

```

图 9-3 鼠标拖动或触屏滑动距离计算代码详情图

10. 用 gitBash 工具管理项目的代码仓库和 http 服务器

10.1 经典 Bash 工具介绍

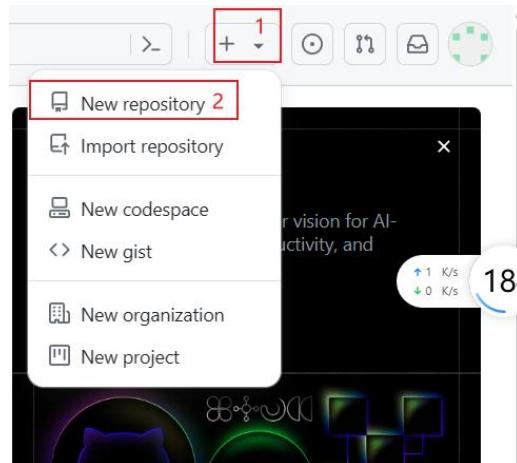
在探讨命令行界面时，我们实际上是在提及 shell。shell 是一个程序，负责接收用户通过键盘输入的命令，并将其传递给操作系统以执行。在 Linux 生态系统中，几乎所有的发行版都采用了来自 GNU 项目的 bash 作为默认的 shell。bash，作为 Bourne-again Shell 的首字母缩写，是对 Steve Bourne 最初在 Unix 上开发的 shell（通常称为 sh）的一种功能增强的替代方案^[7]。

类似于 Windows，类 Unix 操作系统如 Linux 也采用了层次化的目录结构来组织文件。这种结构被形象地比喻为树状，有时在其他系统中也被称为文件夹结构。在此结构中，文件被分类并放置在各级目录中，这些目录可以进一步包含文件和其他子目录。文件系统的起点是根目录，它不仅是所有文件和目录的起点，还包含了各种关键文件和子目录，这些子目录又可能包含更多的文件和子目录，以此类推，形成了一个完整的文件系统树^[7]。

10.2 通过 GitHub 平台实现本项目的全球域名

10.2.1 创建一个空的远程代码仓库

注册并登录 GitHub 网站后，创建仓库，步骤如下图所示。



The screenshot shows the 'Create a new repository' form on GitHub. At the top, it says 'Create a new repository'. Below that, a note says 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)' A note at the bottom of the form says 'Great repository names are short and memorable. Need inspiration? How about [bug-free-octo-disco](#) ?'

Form fields include:

- Owner ***: Rrreal-love
- Repository name ***: (empty input field)
- Description (optional)**: (empty input field)
- Visibility**:
 - Public**: Anyone on the internet can see this repository. You choose who can commit.
 - Private**: You choose who can see and commit to this repository.
- Initialize this repository with:**
 - Add a README file

A large green 'Create repository' button is at the bottom right.

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。

10.2.2 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。

10.2.3 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

```
HP@C MINGW64 /webUI (master)
$ echo "WebUI应用的远程http服务器设置" >> README.md

HP@C MINGW64 /webUI (master)
$ git init

HP@C MINGW64 /webUI (master)
$ git add README.md

HP@C MINGW64 /webUI (master)
$ git commit -m "这是我第一次把代码仓库上传至GitHub平台"
[master adea7b0] 这是我第一次把代码仓库上传至GitHub平台
 1 file changed, 1 insertion(+)

HP@C MINGW64 /webUI (master)
$ git branch -M main

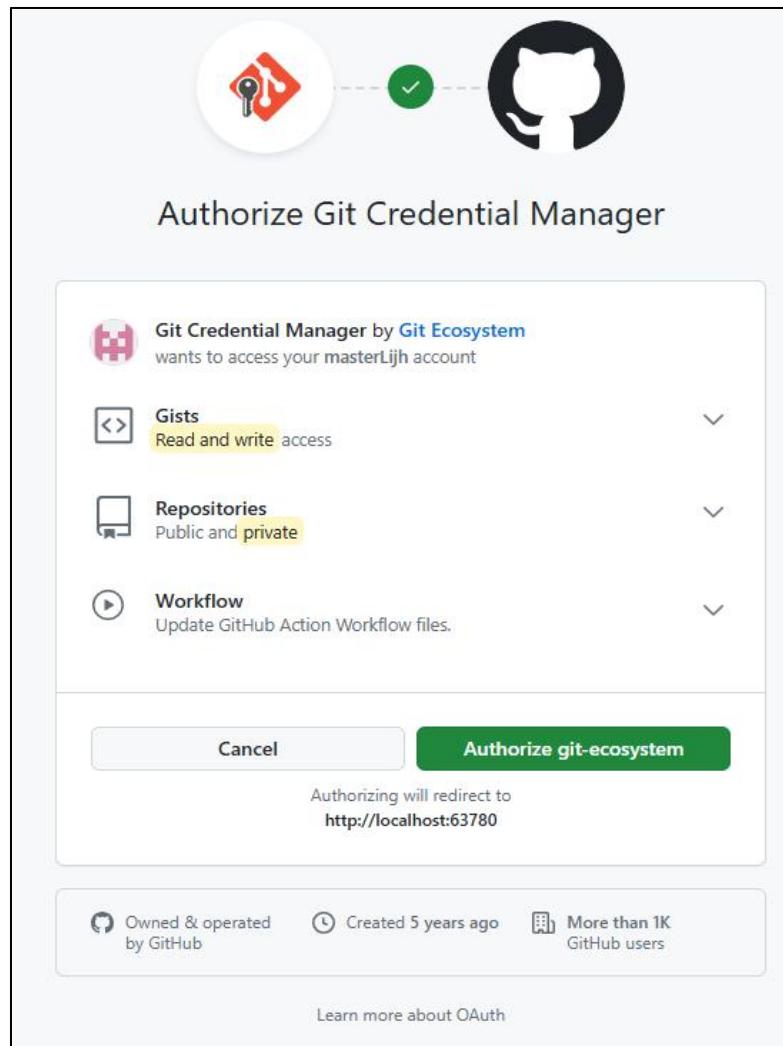
HP@C MINGW64 /webUI (main)
$ git remote add origin http://rabaesther.github.io/RabaEsther/

HP@C MINGW64 /webUI (main)
$ git push -u origin main
```

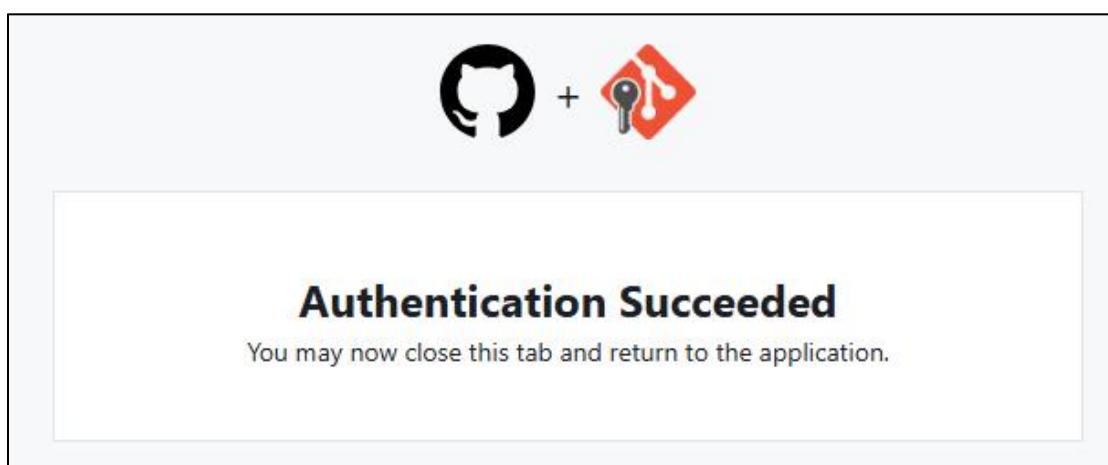
本项目使用 window 平台， gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



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：

The screenshot shows a Microsoft Edge browser window with the title bar "web作业". The main content area displays a pink-themed web application for managing assignments. At the top, it says "20213597web" and "作业". Below this is a table with five rows, each representing an assignment submission:

日期	作业
20230924	第一次作业
20231009	第二次作业
20231011	第三次作业
20231014	第四次作业
20231018	第五次作业

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