

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 设计与实现

（外文）：Personalized UI design and implementation

院（系）：元宇宙产业学院

专 业：计算机科学与技术

学生姓名：刘可汉

学 号：20213613

指导教师：李建宏

2024 年 6 月 12 日

目录

1. 引言.....	4
1.1 项目概述	4
1.2 研学计划	4
1.3 研究方法	5
2. 技术总结及文献综述	6
2.1 Web 平台和客户端技术概述	6
2.2 增量式迭代开发模式的应用.....	8
2.3 高质量代码的开发与维护.....	8
2.4 文献综述.....	9
3. 内容设计概要	10
3.1 分析和设计.....	10
3.2 项目的实现和编程.....	11
3.3 项目的运行和测试.....	12
3.4 项目的代码提交和版本管理.....	13
4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计.....	15
4.1 分析和设计	15
4.2 项目的实现和编程	16
4.3 项目的运行和测试	17
5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI	20
5.1 分析和设计	20
5.2 项目的实现和编程	20
5.3 项目的运行和测试	24
6. 个性化 UI 设计中对鼠标交互的设计开发.....	26
6.1 分析和设计	26
6.2 项目的实现和编程	26
6.3 项目的运行和测试	30
7. 对触屏和鼠标的通用交互操作的设计开发	33
7.1 分析和设计	33
7.2 项目的实现和编程	33

7.3 项目的运行和测试	37
8. UI 的个性化键盘交互控制的设计开发.....	39
8.1 分析和设计	39
8.2 项目的实现和编程	39
8.3 项目的运行和测试	42
9. 谈谈本项目中的高质量代码	44
9.1 响应式 UI 设计	44
9.2 动态设置字体和元素尺寸.....	44
9.3 处理鼠标和触屏事件.....	45
10. 用 gitBash 工具管理项目的代码仓库和 http 服务器	46
10.1 经典 Bash 工具介绍	46
10.2 创建 github 代码仓库	46
10.3 建立连接	47
10.4 本地仓库与 github 远端仓库连接.....	49
参考文献	52

Web 技术的个性化 UI 设计

摘要: 本论文旨在通过 Web 平台和客户端技术, 设计和开发一系列具有响应式 UI 和高质量代码的交互操作功能。通过增量式迭代开发模式, 我们逐步实现了项目的核心功能, 并进行了详细的测试以确保其稳定性和可用性。在内容设计方面, 我们注重了对窄屏和宽屏终端的适配, 采用响应式设计技术, 使 UI 能够根据不同设备的屏幕尺寸自动调整布局和元素大小。同时, 我们还针对鼠标和触屏设备进行了通用交互操作的设计开发, 提升了用户体验。在个性化 UI 设计方面, 我们特别关注了键盘交互控制的设计, 通过编写 HTML、CSS 和 JavaScript 代码, 实现了个性化的键盘操作功能, 增强了用户与界面的交互体验。此外, 我们还深入探讨了高质量代码的开发与维护, 包括响应式 UI 设计、动态设置字体和元素尺寸以及处理鼠标和触屏事件等方面的技巧和方法。这些实践不仅提高了代码的可读性和可维护性, 也为项目的稳定性和扩展性提供了保障。最后, 我们利用 gitBash 工具对项目的代码仓库进行了有效的管理, 并通过建立与 github 远端仓库的连接, 实现了代码的版本控制和协作开发。

关键词: Web 技术; HTML; CSS; JavaScript; 个性化; 不同用户端

Personalized UI design for Web technology

Abstract: The aim of this thesis is to design and develop a series of interactive operation functions with responsive UI and high-quality code through web platform and client-side technologies. Through an incremental and iterative development model, we gradually implemented the core functions of the project and conducted detailed tests to ensure its stability and usability. In terms of content design, we focused on adapting to narrow and wide screen terminals, and adopted responsive design techniques so that the UI could automatically adjust the layout and element size according to the screen size of different devices. At the same time, we also designed and developed universal interaction operations for mouse and touch screen devices to enhance user experience. In terms of personalised UI design, we paid special attention to the design of keyboard interaction control. By writing HTML, CSS and JavaScript codes, we implemented personalised keyboard operation functions to enhance the user's interaction experience with the interface. In addition, we also delved into the development and maintenance of high-quality code, including tips and tricks for responsive UI design, dynamically setting fonts and element sizes, as well as handling mouse and touch screen events. These practices not only improve code readability and maintainability, but also provide stability and scalability for the project. Finally, we used the gitBash tool to effectively manage the project's code repository, and achieved version control and collaborative development of the code by establishing a connection to the github remote repository.

Key words: Web technology; HTML; CSS; JavaScript; Personalization; Different clients

1. 引言

1.1 项目概述

在当今信息化高速发展的时代，软件应用已成为人们日常生活和工作中不可或缺的一部分。本项目旨在开发一个基于 Web 客户端技术的个性化 UI 设计与实现系统，该系统将充分利用现代 Web 技术，如 HTML5、CSS3、JavaScript 等，结合响应式设计原则，为用户提供一个自适应多种设备和交互方式的用户界面。

本项目的核心目标是构建一个功能强大、易于使用、用户体验优良的软件系统。我们计划实现的软件将不仅支持传统的鼠标和键盘输入，还将兼容触屏操作，确保用户能够在各种设备上获得流畅、无缝的操作体验。此外，该系统还将提供个性化推荐功能，根据用户的喜好和行为习惯，智能推荐相关的内容和服务，进一步提升用户体验。

为实现这一目标，我们将采取先进的软件工程开发方法，包括需求分析、系统设计、编码实现、测试验收等阶段。在每个阶段中，我们都将严格按照软件工程的标准和规范进行工作，确保项目的质量和进度。

1.2 研学计划

一、前期准备与需求明确

- 深入搜集和阅读相关文献，确保团队掌握最新技术和设计理念。
- 分析成功案例，结合市场需求和用户行为，明确项目功能需求和用户体验目标。

二、系统设计与原型开发

- 选择合适的前端框架、CSS 预处理器和版本控制工具，构建项目基础。
- 设计清晰的系统架构，确保模块间耦合度低、内聚度高，便于后期维护。
- 快速开发系统原型，并通过用户反馈进行持续优化，确保满足用户期望。

三、核心功能实现与项目展示

- 实现项目的核心功能模块，确保功能完整性和准确性。
- 进行模块集成测试，确保各模块协同工作无误，功能稳定可靠。
- 设计和实现响应式 UI，优化系统性能，提升用户体验。
- 总结项目经验，展示项目成果，为未来的项目提供参考和借鉴。

1.3 研究方法

1. 导师指导：项目将由具有丰富经验和专业知识的导师进行指导。导师将根据学生的实际情况和项目需求，制定详细的学习计划和任务安排，并定期进行进度检查和指导。同时，导师还将为学生提供专业的指导和建议，帮助他们解决在项目开发过程中遇到的问题。
2. 小组讨论：我们将定期组织小组讨论会，让学生分享自己的学习心得和实践经验。在小组讨论中，学生可以互相学习、互相借鉴，促进知识的交流和共享。同时，导师和助教也将参与讨论，为学生提供必要的指导和支持。
3. 实践操作：实践操作是本项目研学的重要组成部分。学生将直接参与项目的实际开发过程，从需求分析开始逐步完成系统的设计和实现。在实践操作中，学生将掌握软件开发的具体技能和工具使用，提升实际操作能力。
4. 反思总结：在项目开发过程中，我们将鼓励学生不断反思和总结自己的学习和实践过程。通过反思总结，学生可以发现自己的不足和问题，并制定相应的改进措施。同时，导师和助教也将对学生的反思总结进行点评和指导，帮助他们更好地提升自己的能力。

2. 技术总结及文献综述

在当今数字化时代，Web 客户端技术的发展日新月异，为用户提供了前所未有的交互体验。特别是在用户界面（UI）设计领域，随着技术的不断革新和用户需求的日益多样化，个性化 UI 设计成为了提升用户体验的重要途径。本项目的核心目标是利用现代 Web 技术，构建一个能够自适应多种设备和用户交互方式的个性化 UI 系统。

2.1 Web 平台和客户端技术概述

Web 技术的发展为用户界面的创新和丰富性提供了广阔的空间。HTML5、CSS3 和 JavaScript 作为 Web 技术的三大基石，为构建复杂且交互性强的用户界面提供了强大的支持。HTML5 不仅增强了页面的语义化，还引入了新的元素和 API，使得 Web 应用能够具备更加丰富的功能。CSS3 则极大地提升了页面的表现力和动画效果，为设计师提供了更多的创作空间。JavaScript 作为 Web 页面的脚本语言，能够实现复杂的交互逻辑和动态效果，使得用户界面更加生动和智能^[1]。

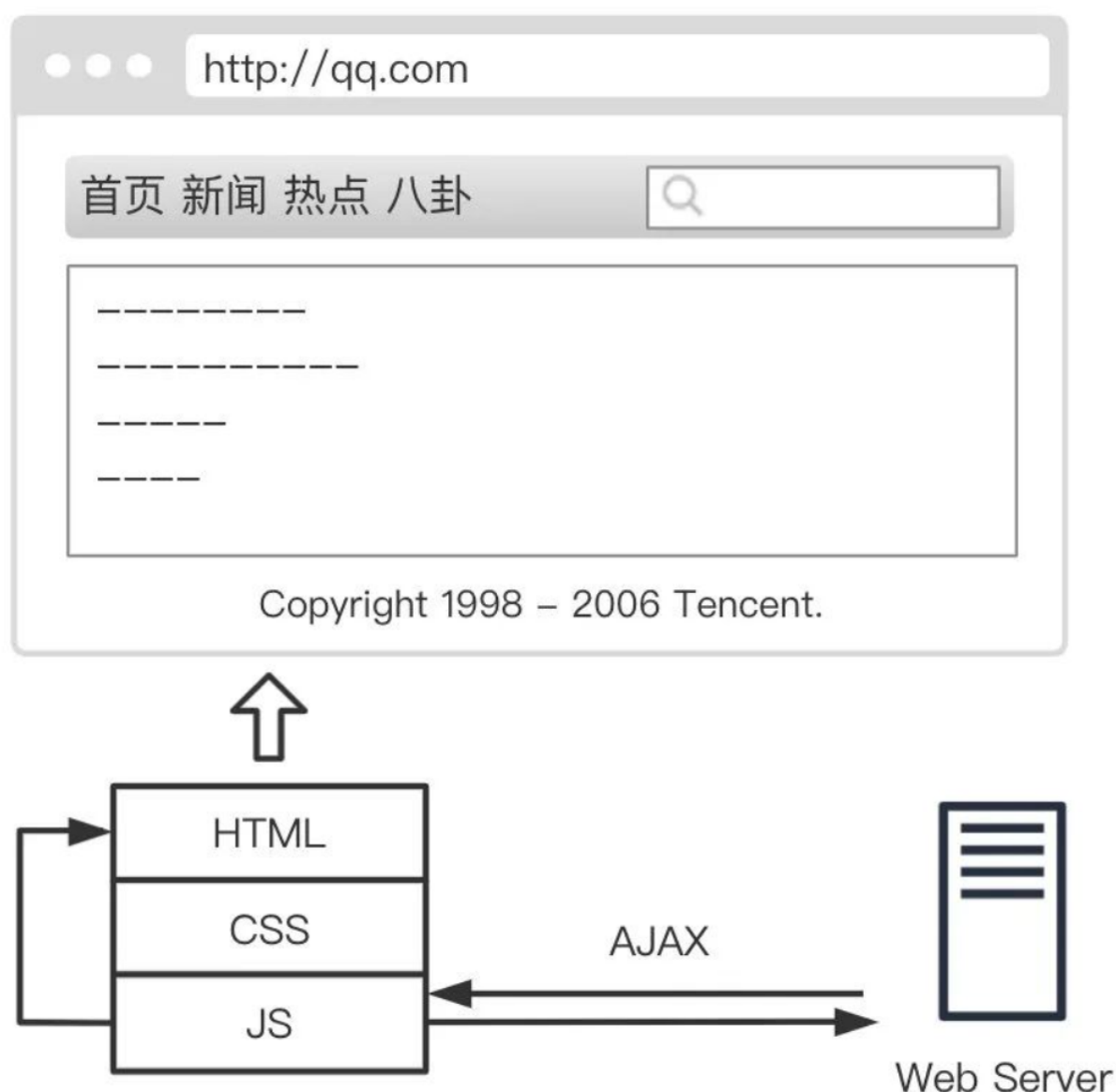


图 0.1 网页服务器工作图

在响应式设计方面，我们采用了媒体查询（Media Queries）和流式布局（Fluid Layouts）等技术，使得 UI 系统能够根据不同的设备和屏幕尺寸进行自适应调整。这种设计方式不仅保证了用户在不同设备上都能获得良好的体验，还使得开发者能够更加灵活地管理和维护代码^[2]。

此外，我们还充分利用了前端框架和库，这些框架和库提供了丰富的组件和工具，使得我们能够更加高效地构建用户界面。同时，这些框架和库还具有良好的扩展性和可维护性，为项目的长期发展奠定了坚实的基础。

2.2 增量式迭代开发模式的应用

在本项目的开发过程中，我们采用了增量式迭代开发模式。这种开发模式允许我们在每个迭代周期中交付一部分功能，并通过用户反馈和测试来不断优化和改进产品。这种开发方式不仅提高了项目的灵活性和可维护性，还使得我们能够更好地控制项目的进度和质量^[6]。

在六次迭代过程中，每一次都是对项目深入理解和精细打磨的机会。每次迭代中，我们不仅专注于实现特定功能，更着重于评估这些功能在实际使用中的表现，以及用户对这些功能的反馈。这些反馈和评估结果直接影响了后续迭代的方向和重点。通过这种方式，我们得以不断修正和优化产品，确保每个迭代都向着最终的目标稳步前进，从而构建出一个既符合用户需求又具备高度灵活性和可维护性的个性化 UI 系统。如下图 2-1 所示：

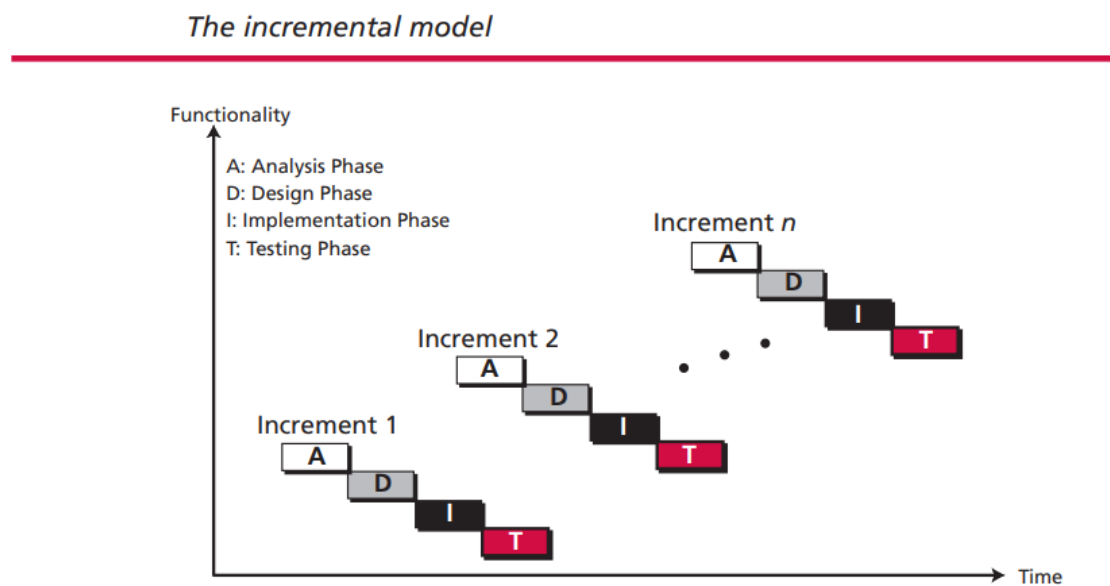


图 0.2 项目的开发迭代图

2.3 高质量代码的开发与维护

在代码开发方面，我们注重代码的质量和可维护性。我们采用了严格的代码规范和命名约定，确保代码的可读性和可理解性。同时，我们还利用版本控制系统（如 Git）来管理项目的代码仓库，从开发者实际工作的工作区开始，通过 `git add` 命令将修改或新增的文件从工作区移动到暂存区，这个过程允许开发者有选择性地提交特定的文件或文件更

改。接着，通过 `git commit` 命令，开发者可以将暂存区的文件更改正式提交到 Git 仓库中，并附上一个描述性的提交信息，以记录这次更改的内容和目的。这一整个流程不仅确保了项目的版本历史被准确、有序地记录，还提供了强大的版本回溯和协作开发能力，使开发者能够轻松地追踪和管理项目的各种变更，从而有效地提高开发效率和质量，最后我们将会将本地库的更新 `push` 到 `github` 的远端库完成一次更新迭代。

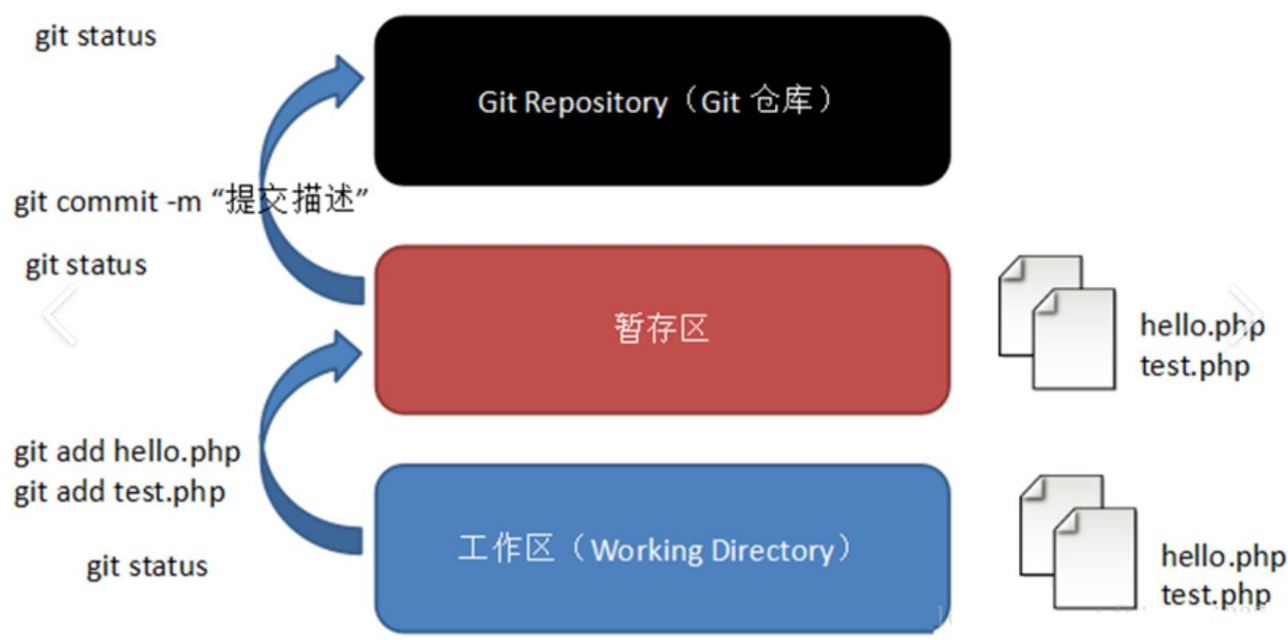


图 0.3 git 管理项目

在代码测试方面，我们采用了自动化测试和手动测试相结合的方式。通过编写测试用例和运行测试脚本，我们能够确保代码的质量和稳定性。同时，我们还邀请了部分用户参与测试，收集他们的反馈和建议，以便更好地改进产品。

2.4 文献综述

随着 Web 技术的快速发展和用户需求的多样化，个性化 UI 设计逐渐成为 Web 应用开发的重要方向之一。在本项目中，我们对个性化 UI 设计的理论基础和实践经验进行了深入的文献综述^[7]。

我们梳理了个性化 UI 设计的发展历程和现状。从早期的静态页面设计到如今的动态交互设计，个性化 UI 设计经历了从简单到复杂、从单一到多元的演变

过程。特别是在移动互联网时代，随着智能手机和平板电脑等移动设备的普及，个性化 UI 设计得到了更加广泛的应用和发展。

我们分析了个性化 UI 设计的关键技术点。这些技术点包括响应式设计、交互设计、动画效果、用户反馈等。响应式设计是实现个性化 UI 设计的重要基础，它能够根据用户的设备和屏幕尺寸进行自适应调整。交互设计则是实现用户与界面之间有效沟通的关键环节，它涉及到用户的行为习惯、心理需求等多个方面。动画效果则能够增强界面的表现力和吸引力，提升用户的视觉体验。用户反馈则是优化和改进产品的重要依据，它能够帮助我们及时发现和解决问题。

我们总结了个性化 UI 设计的实践经验和案例分析。通过对一些成功的 Web 应用进行分析和比较，我们发现这些应用都具备一些共同的特点，如界面简洁明了、交互方式自然流畅、用户体验良好等。这些特点为我们提供了宝贵的启示和借鉴。

综上所述，个性化 UI 设计是提升 Web 应用用户体验的重要途径之一。通过深入研究和开发，我们能够构建出更加符合用户需求和期望的个性化 UI 系统。同时，我们也需要不断学习和探索新的技术和方法，以应对不断变化的市场和用户需求。

3. 内容设计概要

3.1 分析和设计

在项目开发的初始阶段，本项目采用了一种直观且用户友好的“三段式”内容设计策略。首先，我们通过一个醒目的标题性元素，如独特的 logo 或引人注目的文字标题，迅速吸引用户的注意力并传达出主题的核心信息。这一步骤旨在确保用户第一时间就能明确项目的主题和重点。

我们进入了内容展示的主要区域。这一区域被设计为项目的核心焦点，强调“内容为王”的原则。我们精心策划和组织了内容，确保用户能够轻松获取

到关键信息，并体验到高质量的内容呈现。内容区的设计不仅注重信息的丰富性和深度，还考虑到用户的阅读习惯和浏览体验，力求为用户提供流畅、直观的信息获取过程。

在页面的底部，我们设置了附加信息区域。这一区域用于展示一些用户可能关心的细节变化或额外信息。通过精心编排的附加内容，我们为用户提供了更多的背景知识和相关信息，帮助他们更全面地了解项目。同时，这一区域也作为页面布局的补充，使整个界面更加完整和平衡。

通过这种“三段式”的内容设计策略，本项目不仅成功吸引了用户的注意力，还为他们提供了丰富、有深度的内容体验。同时，附加信息区域的设置也进一步满足了用户的多样化需求，提升了整体的用户体验。

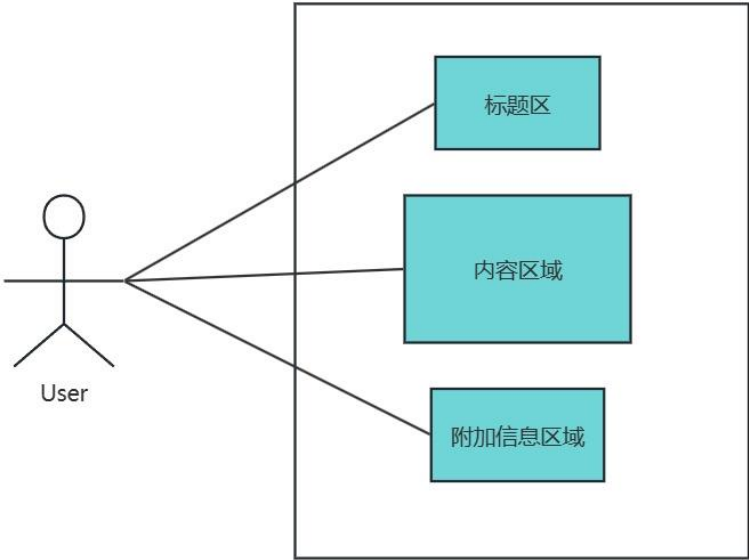


图 0.1 用例图

3.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
    《 我的毕设题目 》
</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>
```

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 Chrome 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



图 3.3 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。

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
$ 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：

```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git add index.html

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
[master 181e0fb] 项目第一版：“三段论”式的内容设计概要开发
2 files changed, 52 insertions(+), 56 deletions(-)
delete mode 100644 1-1.html
create mode 100644 index.html
```

图 0.4 gitbash 代码提交反馈

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：

```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git log
commit 181e0fbb48dbf9752af488acad8cb293fae5052f (HEAD -> master)
Author: LKH <2339909259@qq.com>
Date: Tue Jun 18 16:36:53 2024 +0800

    项目第一版：“三段论”式的内容设计概要开发

commit cdec3b29058da9373183da852dd07a535bda1afe (origin/master)
Author: LKH <2339909259@qq.com>
Date: Tue Jun 18 11:39:56 2024 +0800

    first commit
```

图 0.5 gitbash 的仓库日志反馈

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

随着科技的飞速进步，移动设备的普及和网络技术的日新月异，用户对于通过多元化终端访问 Web 应用的需求愈发强烈。在这个移动互联的时代，窄屏终端，特别是智能手机和平板电脑，因其便携性和功能的丰富性，已经成为了人们日常生活中不可或缺的一部分。

为了满足用户在不同屏幕尺寸和分辨率下的访问需求，响应式设计技术应运而生，成为了现代 Web 开发中的一项重要手段。响应式设计不仅仅是简单地调整网页在不同屏幕上的布局和尺寸，更是通过深入考虑用户体验、交互设计和内容呈现，确保 Web 应用在各种设备上都能呈现出最佳的视觉效果和交互体验。

响应式设计的关键在于其灵活性和适应性。通过运用媒体查询、流式布局和弹性图片等技术，响应式设计能够自动识别用户的设备类型和屏幕尺寸，并据此调整页面的布局、字体大小、图片尺寸等元素，以确保页面在不同设备上的显示效果都能达到最佳状态。

在响应式设计中，UI 设计也扮演着至关重要的角色。通过精心设计的 UI 元素和布局，可以引导用户更轻松地浏览和操作 Web 应用，提升用户体验。同时，响应式设计还注重内容的可读性和可访问性，确保用户在不同设备上都能轻松获取所需信息。

除了适应不同屏幕尺寸外，响应式设计还需要考虑不同设备间的交互差异。例如，触摸屏设备需要支持触摸操作，而键盘和鼠标则更适合于传统的 PC 设备。因此，响应式设计还需要针对不同设备的交互特性进行相应的优化和调整，以确保用户在使用 Web 应用时能够获得更加流畅和自然的体验。

4.2 项目的实现和编程

一、HTML 代码编写如下：

```
<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">
    XXX @ 江西科技师范大学 2025
  </p>
</footer>
```

二、CSS 代码编写如下：

```
*{
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%;  
}
```

三、javascript 代码编写如下：

```
<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>
```

4.3 项目的运行和测试

确保项目在不同设备和浏览器上的兼容性和性能是响应式设计的核心要素。为了实现这一目标，我们采取了一系列的措施来全面测试和优化 Web 应用的显示效果和性能。

我们在不同的主流浏览器（如 Chrome、Firefox、Safari、Edge 等）上进行详尽的 UI 测试。由于各个浏览器对 HTML、CSS 和 JavaScript 的解析和渲染方式可能有所不同，因此我们需要确保 Web 应用在每种浏览器上都能呈现出一致的显示效果，避免因浏览器差异导致的用户体验下降。

其次，对于移动设备，我们需要在真实的设备上进行测试，以模拟用户在实际使用场景中的体验。不同设备拥有不同的屏幕尺寸、分辨率和操作系统，因此我们需要确保 Web 应用在各种屏幕尺寸和分辨率上都能保持最佳的显示效果和交互体验。通过在实际设备上的测试，我们可以发现并解决一些在模拟器或仿真器中无法发现的问题。

此外，利用开发者工具（如 Chrome DevTools）中的设备模拟功能，我们快

速验证 Web 应用在不同设备和浏览器上的显示效果。这些工具允许我们模拟各种设备和浏览器的屏幕尺寸、分辨率和操作系统，从而大大加快了测试的速度和效率。

最后，通过用户测试，我们可以收集实际用户的反馈，了解他们在不同设备上的使用体验。这些反馈可以帮助我们发现一些在测试阶段可能忽略的问题，并根据用户的反馈进行调整和改进。通过不断优化和改进，我们可以提升用户满意度，让 Web 应用在不同设备和浏览器上都能提供出色的用户体验。

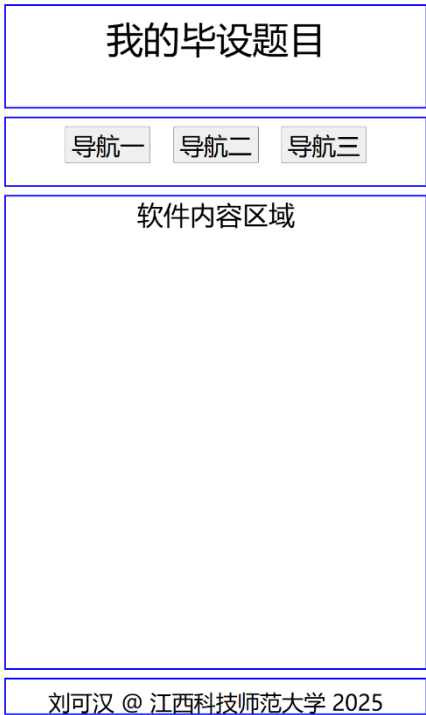


图 4.1 移动端项目的 PC 端访问



图 4.2 移动端访问

4.4 项目的代码提交和版本管理

编写好的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add 1.2.html
$ git commit -m 项目第二版：移动互联时代的 UI 开发初步——窄屏终端的响应式设计
```

成功提交代码后，gitbash 的反馈如下所示：

```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git add 1.2.html

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git commit -m 项目第二版：移动互联时代的UI开发初步—窄屏终端的响应式设计
[master 9047f84] 项目第二版：移动互联时代的UI开发初步—窄屏终端的响应式设计
1 file changed, 77 insertions(+)
create mode 100644 1.2.html
```

图 4.3 gitbash 提交反馈

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：

```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git log
commit 9047f84d1fedc3818548f32ebd0b35f1c4e539d6 (HEAD -> master)
Author: LKH <2339909259@qq.com>
Date: Tue Jun 18 16:59:29 2024 +0800
```

项目第二版：移动互联时代的UI开发初步—窄屏终端的响应式设计

图 4.4 gitbash 的仓库日志反馈

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.1 分析和设计

在当今数字化时代，随着移动设备、平板电脑、笔记本电脑和桌面显示器等设备的多样化，确保用户界面（UI）能够灵活适应不同屏幕尺寸和分辨率变得至关重要。这正是响应式设计技术所追求的目标——构建能够根据不同屏幕尺寸和设备类型自动调整布局和元素的 UI。应用响应式设计技术开发的 UI，就像一块能够变形的布料，无论是窄屏还是宽屏，都能展现出恰到好处的外观和布局。在窄屏设备上，如智能手机，UI 会智能地收缩内容区域，优化元素间距，甚至隐藏一些非关键信息，以确保用户能够轻松浏览和操作。而在宽屏设备上，如桌面显示器，UI 则会扩展内容区域，展示更多信息，为用户提供更加丰富和详细的界面体验。这种自适应的能力不仅提升了用户在不同设备上的使用便利性，还增强了应用的可用性和可访问性。无论是快速浏览信息、进行交互操作还是享受视觉体验，用户都能在应用中找到最适合自己的方式。响应式设计技术的应用，不仅要求设计师具备精湛的技艺和敏锐的洞察力，还需要开发团队具备跨平台、跨设备的开发能力。通过精细的布局规划、灵活的媒体查询和先进的 CSS 技术，我们能够打造出既美观又实用的 UI，为用户带来卓越的体验。

5.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
  <p id="book">
    《我的毕设题目》
  </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">
    书的封面图
  </div>
</main>
<footer>
  Copyright 刘可汉 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>
```

二、CSS 代码编写如下:

```
*{
  text-align: center;
  box-sizing: border-box ;
}
header,main,div#bookface,nav,footer{
  margin:1em;
}
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%;
    }
    body{
    position:relative ;
    }
    #aid{
        position: absolute;
        border: 3px solid blue;
        top: 0.5em;
        left: 600px;
    }
    #bookface{
        width: 80%;
        height: 80%;
        border: 1px solid red;
        background-color: blanchedalmond;
        margin:auto;
    }
}

```

三、 javascript 代码编写如下：

```

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 - 8*baseFont + "px";

if(window.innerWidth < 900){
    $("aid").style.display='none';
}
$("aid").style.width=window.innerWidth - UI.appWidth - 2*baseFont + 'px';
$("aid").style.height= document.body.clientHeight + 'px';

//尝试对鼠标设计 UI 控制
var mouse={};
mouse.isDown= false;
mouse.x= 0;

```

```

mouse.deltaX=0;
$("#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){
    //console.log(ev);
    $("#bookface").textContent="鼠标已经离开";

});
$("#body").addEventListener("keypress",function(ev){
    let k = ev.key;
    let c = ev.keyCode;
    $("#keyboard").textContent = "您的按键 : " + k + " , " + "字符编码 : " + c;
});

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 $

```


5.3 项目的运行和测试

确保响应式 Web 应用在不同设备和浏览器上的兼容性和性能至关重要。我们采取了一系列措施，包括主流浏览器测试、真实移动设备测试、开发者工具模拟以及用户测试，来全面验证和优化应用的显示效果和性能。这些测试旨在确保 Web 应用在各种设备和浏览器上都能提供一致且优质的用户体验，从而提升用户满意度。



图 5.1 鼠标模型 PC 端



图 5.2 鼠标模型移动端

5.4 项目的代码提交和版本管理

编写好的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add 1.3.html
$ git commit -m 项目第三版：响应式设计（宽屏和窄屏）
```

成功提交代码后，gitbash 的反馈如下所示：

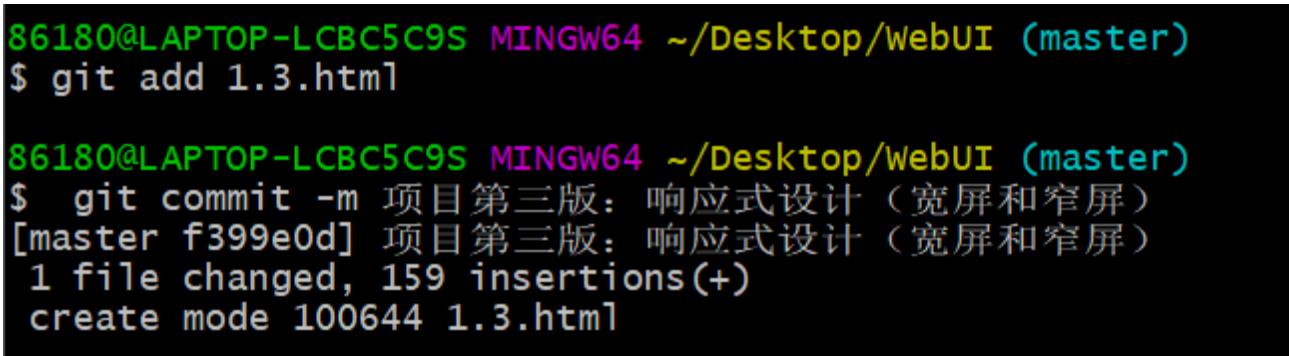


图 5.3 gitbash 提交反馈

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：

```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git log
commit f399e0dedf269b360bc916b1c82e6b51974f5cde (HEAD -> master)
Author: LKH <2339909259@qq.com>
Date: Tue Jun 18 17:19:04 2024 +0800
```

项目第三版：响应式设计（宽屏和窄屏）

图 5.4 gitbash 的仓库日志反馈

6. 个性化 UI 设计中对鼠标交互的设计开发

6.1 分析和设计

在构建现代 Web 应用时，鼠标作为 PC 端用户的主要交互工具，其个性化设计的重要性不言而喻。为了提升用户体验、增强应用的易用性和吸引力，我们需要精心设计和实现基于鼠标的个性化 UI。首先，通过深入分析用户的使用习惯和需求，我们可以确定哪些鼠标交互行为能够为用户带来更大的便利和愉悦。在设计阶段，我们将这些分析结果融入 UI 设计中，创造出独特且符合用户期望的鼠标交互效果。接下来，利用 HTML、CSS 和 JavaScript 等前端技术，我们将这些设计转化为实际的功能实现，确保鼠标的各种动作（如悬停、点击、拖动等）都能触发相应的界面反馈和动画效果。完成实现后，我们会对应用进行严格的运行和测试，确保鼠标交互的准确性和流畅性，并在不同浏览器和设备上进行兼容性测试。最后，我们将代码提交到版本管理系统，并进行规范的版本管理，以便跟踪代码变更、协同开发以及后续的问题修复和功能迭代。这样，通过全方位的设计、实现、测试和版本管理，我们能够为用户提供更优质、更个性化的鼠标交互体验。

6.2 项目的实现和编程

一、HTML 代码编写如下：

```
<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 刘可汉 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>
```

二、CSS 代码编写如下:

```
*{
  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;
    }
    body{
        position: relative;
    }
    button{
        font-size: 1em;
    }
    #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% ;
    }
}

```

三、javascript 代码编写如下：

```

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';

//尝试对鼠标设计 UI 控制
var mouse={};
mouse.isDown= false;
mouse.x= 0;
mouse.y= 0;
mouse.deltaX=0;
$("bookface").addEventListener("mousedown",function(ev){
    mouse.isDown=true;
    mouse.x= ev.pageX;
    mouse.y= ev.pageY;
    console.log("mouseDown at x: "+(" "+mouse.x +"," +mouse.y +"))");
    $("bookface").textContent= "鼠标按下，坐标： "+(" "+mouse.x+","+mouse.y+");
});
$("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' ;
    }
});
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 $

```

6.3 项目的运行和测试

为了确保基于鼠标的个性化 UI 功能稳定运行并为用户带来卓越体验，我们实施了一套全面的测试策略。首先，我们编写详尽的单元测试，确保每个鼠标事件处理函数和交互逻辑都能按照预期正常工作，没有遗漏或错误。这些单元测试覆盖了从简单的点击事件到复杂的拖动和悬停交互的各种场景。

接着，我们进行集成测试，以验证不同模块之间的交互是否顺畅，以及整体功能是否满足设计要求。集成测试确保了各个组件在整合到系统中后，仍然能够保持其独立测试时的稳定性和正确性。

为了进一步提升测试的全面性和准确性，我们在多种主流浏览器（如 Chrome、Firefox、Safari、Edge）上测试鼠标交互功能，确保在不同浏览器环境下，鼠标操作都能呈现一致且正确的行为。这一步骤有助于我们发现并修复

潜在的浏览器兼容性问题。

最后，我们进行跨设备测试，确保在不同屏幕尺寸、分辨率和操作系统的设备上，鼠标操作的响应速度和显示效果都能保持正常。这种跨设备测试覆盖了从桌面电脑到平板电脑再到智能手机的广泛范围，确保我们的 Web 应用能够在各种设备上提供优质的用户体验。



图 6.1 触屏和鼠标的通用操作 PC 端



图 6.2 移动端显示

6.4 项目的代码提交和版本管理

编写好的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add 1.4.html
```

```
$ git commit -m 项目第四版：UI 设计之鼠标模型的控制
```

成功提交代码后，gitbash 的反馈如下所示：

```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git add 1.4.html

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git commit -m 项目第四版：UI设计之鼠标模型的控制
[master f50140a] 项目第四版：UI设计之鼠标模型的控制
1 file changed, 190 insertions(+)
create mode 100644 1.4.html
```

图 6.3 gitbash 提交反馈

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：


```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/webUI (master)
$ git log
commit f50140acac0ff9cc73fc91e2e55818c98cf52040 (HEAD -> master)
Author: LKH <2339909259@qq.com>
Date: Tue Jun 18 17:31:49 2024 +0800
```

项目第四版：UI设计之鼠标模型的控制

图 6.4 gitbash 的仓库日志反馈

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

在构建现代 Web 应用时，用户可能通过多样化的设备访问，无论是带有触摸屏的移动设备还是使用鼠标的桌面电脑。为了确保无缝的用户体验，设计和开发一个能够灵活适应触屏与鼠标操作、统一而流畅的交互界面变得至关重要。接下来，我们将深入探讨如何细致分析、精心设计和有效实现这种跨设备的通用交互策略。

7.2 项目的实现和编程

二、HTML 代码编写如下：

```
<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 刘可汉 江西科技师范大学 2024--2025
</footer>
<div id="aid">
  <p>用户键盘响应区</p>
</div>
```

二、CSS 代码编写如下：

```
*{
    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;
}
body{
    position: relative;
}
button{
    font-size: 1em;
}
#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% ;}
```

三、javascript 代码编写如下：

```
var UI = {};
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 $

```

7.3 项目的运行和测试

在保障触屏与鼠标通用交互操作顺畅运行时，全面测试与优化是不可或缺的环节。首先，进行集成测试，验证不同交互操作和模块间的协同效果。同时，我

们需在主流桌面浏览器和移动浏览器中测试应用，确保行为一致且体验流畅。进一步地，我们应在各种实际设备上开展测试，包括不同品牌和型号的智能手机、平板电脑以及 PC，以验证应用的广泛兼容性。此外，邀请真实用户在实际环境中使用应用，并观察他们的操作行为和收集反馈。最后，我们需分析并优化事件处理的响应速度，确保在触屏和鼠标操作下都能提供流畅无阻的用户体验。

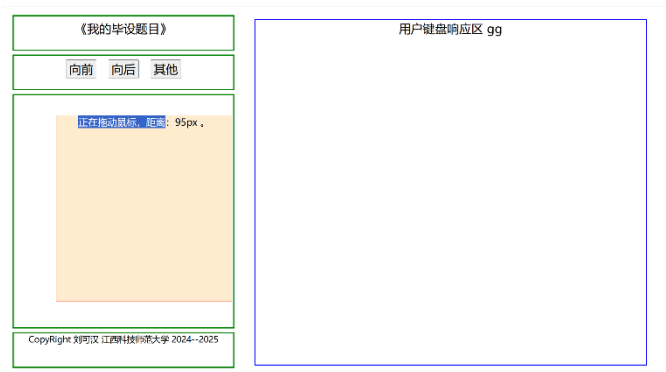


图 7.1 UI 个性化键盘交互控制 PC 端



图 7.2 移动端访问

7.4 项目的代码提交和版本管理

编写好的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add 1.5.html
$ git commit -m 项目第五版：通用的 UI 设计，用一套代码同时为触屏和鼠标建模
成功提交代码后，gitbash 的反馈如下所示：
```



图 7.3 gitbash 提交反馈



图 7.4 gitbash 日志反馈

8. UI 的个性化键盘交互控制的设计开发

8.1 分析和设计

在现代 Web 应用设计中，键盘交互对于提升用户体验的重要性不容忽视，特别是在桌面环境中。用户习惯于依赖键盘进行高效的操作，尤其是在需要快速输入或执行复杂命令的场景中。因此，个性化的键盘交互设计不仅是提升用户操作效率的关键，更是为用户带来便利和愉悦体验的重要手段。

设计个性化的键盘交互控制需要深入理解用户的使用习惯和特定应用场景的需求。首先，我们需要对用户群体进行细致的分析，了解他们的工作流程、常用操作以及可能遇到的挑战。基于这些洞察，我们可以开始构思如何优化键盘交互，使其更符合用户的自然操作习惯。

在本次的项目中我们通过实现了 web 与键盘的交互，通过对键盘事件的监听，我们探索了 web 平台下的键盘交互事件。

8.2 项目的实现和编程

三、HTML 代码编写如下：

```
<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">
    本利代码的运行需要超过 1 千像素宽度的宽屏<br>
    建议使用有键盘的 PC 运行和调试代码<br>
    当然拖动/滑动超过 100 像素的 UI 互动依然有效！
  </div>
</main>
```



```
<footer>
```

```
    Copyright 刘可汉 江西科技师范大学 2024--2025
```

```
</footer>
```

```
<div id="aid">
```

```
    用户键盘响应区
```

```
    <p id="typeText"></p>
```

```
    <p id="keyStatus"></p>
```

```
</div>
```

二、CSS 代码编写如下：

```
*{
    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;
}
footer{
    border: 3px solid green;
    height: 10%;
    font-size: 0.7em;
}
body{
    position: relative;
}
button{
    font-size: 1em;
}

#bookface{
    position: absolute;
    width: 80%;
    height: 80%;
```

```

        border: 1px solid red;
        background-color: blanchedalmond;
        left: 7% ;
        top: 7% ;
    }
    #aid{
        position: absolute;
        border: 3px solid blue;
        top: 0px;
        left: 600px;
    }
    #typeText{
        border: 1px solid blue;
        padding: 0.2em ;
        color: gray ;
    }
    #keyStatus{
        position: absolute;
        border: 1px solid blue;
        width: 90% ;
        right: 0;
        bottom: 0;
        font-size: 0.6em;
        padding: 0.5em ;
    }
}

```

三、javascript 代码编写如下：

//提出问题：研究利用"keydown"和"keyup"2 个底层事件，实现同时输出按键状态和文本内容

```

$("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)
});
} //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 $

```

8.3 项目的运行和测试

首先，我们对每一个按键按下松开的效果进行了实验，验证不同交互操作和模块间的协同效果，结果同期望。同时，我们需在主流桌面浏览器和移动浏览器中测试应用，确保行为一致且体验流畅。进一步地，我们应在各种实际设备上开

展测试，包括不同品牌和型号的 PC，以验证应用的广泛兼容性。此外，邀请真实用户在实际环境中使用应用，并观察他们的操作行为和收集反馈。



图 8.1 触屏和鼠标的通用操作 PC 端



图 8.2 移动端显示

8.4 项目的代码提交和版本管理

编写好的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add 1.6.html
$ git commit -m 项目第六版：UI 的个性化键盘控制——巧妙应用 keydown 和 keyup 键盘底层事件
```

成功提交代码后，gitbash 的反馈如下所示：

```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git add 1.6.html

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git commit -m 项目第六版：UI的个性化键盘控制——巧妙应用keydown和keyup键盘底层事件
[master c9d3b87] 项目第六版：UI的个性化键盘控制——巧妙应用keydown和keyup键盘底层事件
1 file changed, 255 insertions(+)
create mode 100644 1.6.html
```

图 8.3 gitbash 提交反馈

```
$ git log

gitbash 反馈代码的仓库日志如下所示：
```

```
86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git log
commit c9d3b87d08f68c9f0751be864a89d4f36f2d7064 (HEAD -> master)
Author: LKH <2339909259@qq.com>
Date: Tue Jun 18 18:00:14 2024 +0800

    项目第六版：UI的个性化键盘控制——巧妙应用keydown和keyup键盘底层事件
```

图 8.4 gitbash 的仓库日志反馈

9. 谈谈本项目中的高质量代码

9.1 响应式 UI 设计

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

图 9.3 移动端访问

这段代码根据窗口的宽度动态设置 `UI.appWidth`，确保应用宽度不会超过 600px，同时 `UI.appHeight` 设置为窗口的内高度。这是响应式设计的一个基本示例，确保应用在不同屏幕尺寸下都能良好地显示。

9.2 动态设置字体和元素尺寸

```
let baseFont = UI.appWidth / 20;  
document.body.style.fontSize = baseFont + "px";  
document.body.style.width = UI.appWidth - baseFont + "px";  
document.body.style.height = UI.appHeight - baseFont * 5 + "px";
```

图 9.3 移动端访问

这里，我们根据应用的宽度动态设置了一个基准字体大小，并用它来计算和设置 `body` 元素的宽度和高度。这种方法允许整个应用的 UI 元素根据屏幕尺寸进行缩放，保持一致的布局和可读性。

9.3 处理鼠标和触屏事件

```
var Pointer = {};  
// ... (初始化Pointer对象的代码)  
  
let handleBegin = function(ev){  
    // ... (处理触摸或鼠标按下事件的代码)  
};  
  
let handleEnd = function(ev){  
    // ... (处理触摸或鼠标释放事件的代码)  
};  
  
let handleMoving = function(ev){  
    // ... (处理触摸或鼠标移动事件的代码)  
    // 注意：这部分代码被截断了，但通常它会处理指针移动的逻辑  
};
```

图 9.3 移动端访问

这部分代码展示了如何设置和处理鼠标和触屏事件。通过 `handleBegin`、`handleEnd` 和 `handleMoving` 函数，我们可以检测到用户的触摸或鼠标动作，并据此更新 UI 或执行其他操作。特别是，代码中还包含了判断触摸滑动或鼠标拖动是否“有效”的逻辑，基于移动的距离（`Pointer.deltaX`）来判断。

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

Git Bash 是专为 Windows 环境下使用 Git 的用户设计的一个命令行工具。它提供了一个类似于 Unix/Linux 的终端环境，让用户能够在 Windows 系统中直接运行 Git 命令以及其他常见的 Unix/Linux 命令行工具。

Git Bash 的主要功能是提供一个模拟 Unix/Linux 的 shell 环境，让用户能够在这个环境中执行 Git 命令。这意味着，无论用户是在 Windows 系统上工作，还是在 Unix/Linux 系统上工作，他们都可以通过相同的 Git Bash 命令行界面来进行版本控制操作，从而保证了跨平台的一致性。此外，Git Bash 还内置了许多常用的 Unix/Linux 命令行工具，如 `ls`、`cd`、`grep`、`awk` 等，使得用户在进行文件操作、文本处理等方面也能够获得与 Unix/Linux 系统相似的体验。

Git Bash 的使用场景主要集中在需要进行版本控制的软件开发项目中。通过 Git Bash，用户可以轻松地克隆远程仓库、创建分支、提交代码、合并分支等操作，从而实现代码的版本控制和协作开发。同时，Git Bash 还支持执行 Shell 脚本，这对于需要自动化执行某些命令或操作的用户来说也是一个非常有用的功能。

10.2 创建 github 代码仓库

库进入 github 后点击如下图标



图 10.1 仓库创建按钮


选择合适且合法的仓库名

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

 dave021119 / WEB-UI

✓ WEB-UI is available.

点击如下图标创建

Create repository

10.3 建立连接

在这里我们使用 ssh 密钥进行连接

首先输入 `$ ssh-keygen -t rsa -C 2339909259@qq.com`

```
lenovo@LAPTOP-7Q8SK644 MINGW64 ~
$ ssh-keygen -t rsa -C "2339909259@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/lenovo/.ssh/id_rsa):
Created directory '/c/Users/lenovo/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/lenovo/.ssh/id_rsa
Your public key has been saved in /c/Users/lenovo/.ssh/id_rsa.pub
The key's fingerprint is:
SHA256:PQ6IzqGwK6Ea0LOa0YarHz2coC67rTfwROtSNALARC0 1525281916@qq.com
The key's randomart image is:
+---[RSA 3072]-----+
|*o.
|.E.
|.
|..+ . .
|.++oo . S o
|+.+@ o o .
|=B* B .
|OO=o .
|/%+.
+----[SHA256]-----+
```

图 10.2 git 创建 ssh 密钥反馈

此时电脑已经有“id_rsa”，“id_rsa.pub”文件

此电脑 > Windows-SSD (C:) > 用户 > lenovo > .ssh

名称	修改日期
id_rsa	2021/1/24 18:22
id_rsa.pub	2021/1/24 18:22

图 10.3 电脑反馈

在 github 上打开“Account settings”-“SSH Keys”页面，新建 ssh keys 并把 id_rsa.pub 的全部内容填入

SSH keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

https://blog.csdn.net/qq_29493173

图 10.4 github 创建密钥

验证是否成功，在 git bash 里输入下面的命令

```
$ ssh -T git@github.com
```

```
lenovo@LAPTOP-7Q8SK644 MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (192.168.1.1)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWG17E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,192.168.1.1' (RSA) to the list of known
hosts.
Hi xu-xiaoya! You've successfully authenticated, but GitHub does not provide shell
access.
```

图 10.5 创建验证

10.4 本地仓库与 github 远端仓库连接

根据创建好的 Git 仓库页面的提示（找自己仓库的提示代码）

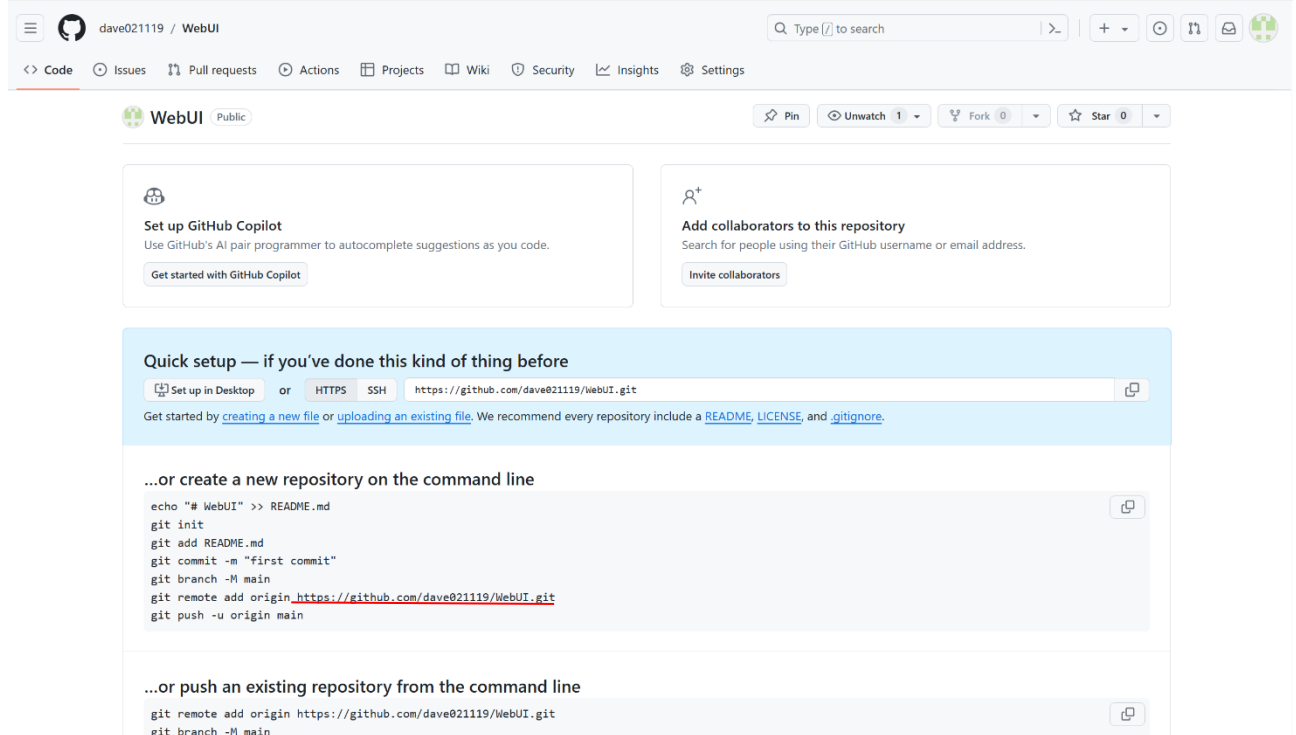


图 10.6 github 仓库创建

输入以下指令

```
$ 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/dave021119/WebUI.git
$ git push -u origin main
```

```

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ echo "WebUI应用的远程http服务器设置" >> README.md

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git init
Reinitialized existing Git repository in C:/Users/86180/Desktop/WebUI/.git/

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git commit -m "这是我第一次把代码仓库上传至github平台"
[master d850cf7] 这是我第一次把代码仓库上传至github平台
1 file changed, 1 insertion(+)
create mode 100644 README.md

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ branch -M main
bash: branch: command not found

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (master)
$ git branch -M main

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (main)
$ git remote add origin https://github.com/dave021119/WebUI.git
error: remote origin already exists.

86180@LAPTOP-LCBC5C9S MINGW64 ~/Desktop/WebUI (main)
$ git push -u origin main
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 16 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (25/25), 9.18 KiB | 2.29 MiB/s, done.
Total 25 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), done.
remote:
remote: Create a pull request for 'main' on GitHub by visiting:
remote:   https://github.com/dave021119/WebUI/pull/new/main
remote:
To https://github.com/dave021119/WebUI.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

```

图 10.7 git 反馈

至此本地库与远程库连接成功，并且完成了第一次代码上传

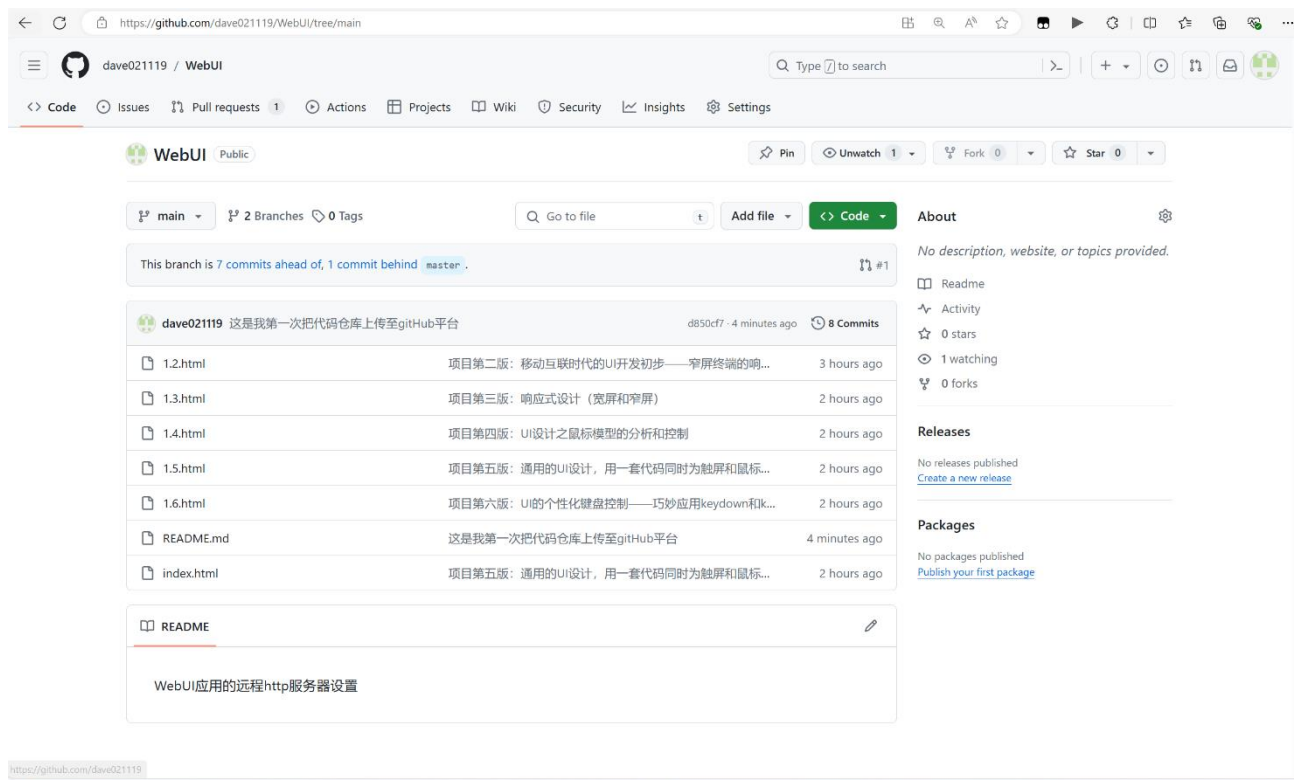


图 10.8 github 仓库反馈

参考文献

- [1] 刘华星, 杨庚. HTML5——下一代Web开发标准研究[J]. 计算机技术与发展, 2011,21(08):54-58.
- [2]HTML5在移动应用开发上的应用前景{Author}:黄永慧;陈程凯[J]. 计算机技术与发展, 2013,23{Issue}: 07:207-210.
- [3]基于响应式Web设计的网页模板的设计与实现{Author}: 张树明[J]. 计算机与现代化, 2013,{Issue}: 06:125-127.
- [4]UML建模技术及其应用{Author}: 陆晓燕,秦朝辉,尹治本[J]. 成都信息工程学院学报, 2004:414-417.
- [5]软件开发模型研究综述{Author}:张友生;李雄[J]. 计算机工程与应用, 2006:109-115.
- [6]瀑布模型、快速原型模型和增量模型的对比{Author}: 张越[J]. 电子技术与软件工程, 2019,{Issue}: 03:32.
- [7]Web服务核心支撑技术:研究综述{Author}: 岳昆,王晓玲,周傲英[J]. 软件学报, 2004:428-442.
- [8]罗超. 鼠标、键盘退场智能硬件掀起交互革命[J]. IT时代周刊, 2014(10):36-37.
- [9]郑桦. 嵌入式Linux文件系统的设计与实现[D]. 武汉理工大学, 2004.
- [10]Git——版本管理之利器{Author}: 宋冬生[J]. 程序员, 2007,{Issue}: 11:118-119.