

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

（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：周嘉旺

学 号：20213643

指导教师：李建宏

2024 年 6 月 16 日

目录

基于 Web 客户端技术的个性化 UI 的设计和编程.....	3
(Customized UI design and Programming based on Web client technology)	3
1. 前言	3
1.1 毕设任务分析	错误! 未定义书签。
1.2 研学计划	4
1.3 研究方法	5
2. 技术总结和文献综述	5
2.1 Web 平台和客户端技术概述	6
2.2 项目的增量式迭代开发模式	7
3. 内容设计概要	9
3.1 分析和设计	9
3.2 项目的实现和编程	9
3.3 项目的运行和测试	10
3.4 项目的代码提交和版本管理	11
4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计	12
4.1 响应式设计——适应显示硬件	12
4.2 实现代码	13
4.3 项目的运行和测试	15
4.4 代码提交	16
5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI	17
5.1 移动互联网用户终端	17
5.2 项目的运行和测试	20
5.3 代码提交	21
6. 个性化 UI 设计中对鼠标交互的设计开发	22
6.1 个性化 UI 设计	22
6.2 项目的运行和测试	25
6.3 代码提交	26
7. 对触屏和鼠标的通用交互操作的设计开发	27
7.2 项目的运行和测试	29
7.3 代码提交	30
8. UI 的个性化键盘交互控制的设计开发	31
8.1 设计和开发 UI 个性化键盘	31
8.2 项目的运行和测试	33
8.3 代码提交	34
9. 谈谈本项目中的高质量代码	35
10. 用 gitBash 工具管理项目的代码仓库和 http 服务器	36
10.1 经典 Bash 工具介绍	36
10.2 通过 gitHub 平台实现本项目的全球域名	36
10.3 创建一个空的远程代码仓库	36
10.4 设置本地仓库和远程代码仓库的链接	37
参考文献:	41
写作指导:	错误! 未定义书签。

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 周嘉旺

摘要：随着互联网技术的飞速发展，WebApp 已成为人们日常生活和工作中不可或缺的一部分。WebApp UI 设计开发作为用户与应用交互的关键环节，直接影响着用户体验和应用的成功与否。本论文深入探讨了 WebApp UI 设计开发的相关理论和实践。首先，对 WebApp UI 设计的基本原则和要素进行了详细阐述，包括布局、色彩、字体、图标等方面，强调了以用户为中心的设计理念。通过对用户需求和行为的深入研究，提出了针对性的设计策略，以提高用户满意度和忠诚度。其次，分析了当前主流的 WebApp UI 开发技术和框架，如 HTML5、CSS3、JavaScript 等，并结合实际案例探讨了如何高效地实现设计理念。同时，研究了响应式设计在不同设备上的适配问题，以确保用户在各种终端上都能获得良好的体验。此外，还探讨了 WebApp UI 设计开发中的交互设计和可用性测试，通过建立用户模型和任务流程，优化了用户操作流程，提高了应用的易用性和效率。通过实际的可用性测试，不断改进和完善设计方案。最后，对未来 WebApp UI 设计开发的趋势进行了展望，预测了人工智能、虚拟现实等新技术将为其带来的新机遇和挑战。本研究旨在为 WebApp UI 设计开发人员提供理论支持和实践指导，推动 WebApp UI 设计开发领域的不断创新和发展。

关键词：WebApp、UI 设计、用户中心、开发技术、响应式设计、交互设计、可用性测试、创新实践、发展趋势。

1. 引言

在当今数字化时代，Web 应用程序（WebApp）已经成为人们日常生活和工作中不可或缺的一部分。WebApp 的用户界面（UI）设计开发在决定用户体验、应用程序的可用性和成功与否方面起着至关重要的作用。随着技术的不断进步和用户需求的日益多样化，WebApp UI 设计面临着诸多挑战和机遇。随着互联网的普及和移动设备的广泛使用，用户对于 WebApp 的期望越来越高。他们不仅要求功能强大、性能优越，更期望拥有简洁、美观、直观且易于操作的界面。一个优秀的 WebApp UI 设计能够有效地引导用户完成任务，提高用户满意度和忠诚度，从而为企业和开发者带来竞争优势。然而，WebApp UI 设计开发并非易事。它需要综合考虑多种因素，如用户需求分析、视觉设计原则、交互设计规范、响应式布局、前端技术实现等。同时，还需要紧跟行业的最新趋势和技术创新，以适应不断变化的用户期望和市场需求。近年来，诸如人工智能、大数据分析和虚拟现实等新兴技术的出现，为 WebApp UI 设计开发带来了新的思路和方法。例如，利用人工智能算法可以实现个性化的用户界面推荐，大数据分析能够深入了解用户行为和偏好，从而为设计优化提供依据，而虚拟现实技术则为用户带来更加沉浸式的体验。综上所述，WebApp UI 设计开发是一个充满挑战 and 创新的领域，对

于提升 WebApp 的质量和用户体验具有重要意义。本论文旨在深入探讨 WebApp UI 设计开发的相关理论和实践，分析当前存在的问题，并提出相应的解决方案和创新思路，以期为该领域的发展做出有益的贡献。

1.1 研学计划

第一阶段：WebApp UI 设计基础

1. WebApp 概述

介绍 WebApp 的特点、应用场景和发展趋势。分析优秀 WebApp 案例，了解其设计理念和用户体验。

2. UI 设计原则与规范

学习色彩搭配、字体排版、图标设计等基本原则。掌握响应式设计、栅格系统等布局规范。

3. 用户研究与需求分析

了解用户研究的方法和流程，如用户访谈、问卷调查等。学会从用户需求出发，进行产品定位和功能规划。

第二阶段：UI 设计工具与技巧

1. 常用设计工具实战

详细讲解 Adobe XD、Sketch、Figma 等工具的使用方法。通过实际案例练习，熟练掌握图形绘制、页面布局、交互设计等操作。

2. 视觉设计技巧

学习如何运用光影、质感、透视等效果增强界面的视觉冲击力。掌握图标设计的技巧和规范，能够设计出简洁、清晰的图标。

3. 原型设计与测试

利用设计工具制作 WebApp 的原型，进行交互流程的设计和优化。进行原型测试，收集用户反馈，改进设计方案。

第三阶段：前端开发基础

1. HTML 与 CSS 基础

学习 HTML 标签的使用，构建网页的结构。掌握 CSS 样式的编写，实现页面的布局 and 美化。

2. JavaScript 基础

了解 JavaScript 的基本语法和数据类型。学习 DOM 操作和事件处理，实现简单的交互效果。

3. 前端框架入门

介绍常用的前端框架，如 Vue.js、React 等。通过简单项目实践，了解框架的基本使用方法。

第四阶段：项目实践与成果展示（第 13 - 15 天）

1. 分组项目实践

学生分组，根据给定的主题或自选项目，进行 WebApp UI 设计与开发。团队协作，完成需求分析、设计方案、前端开发和测试等环节。

2. 成果展示与评估

各小组展示自己的项目成果，包括设计文档、原型和代码。进行项目评估，分享经验和教训，评选优秀作品。

1.2 研究方法

1. 理论学习

阅读相关的设计书籍，如《用户体验要素》《简约至上》等，了解 UI 设计的原则、流程和方法。学习色彩理论、排版原则、图标设计等基础知识。研究最新的 UI 设计趋势和案例，关注行业知名的设计网站和博客。

2. 实践操作

选择一款常用的设计工具，通过教程和练习熟悉工具的使用。从简单的项目开始，如设计一个登录页面或个人博客页面，逐步提升设计难度。参与实际的 WebApp 项目，或者模拟真实的项目需求进行设计实践。

3. 分析案例

收集优秀的 WebApp UI 案例，分析其布局、色彩搭配、交互方式等方面的优点。

对同类型的 WebApp 进行对比分析，找出差异和创新点。尝试还原案例的设计过程，思考背后的设计决策。

4. 用户研究

了解用户需求和行为习惯，通过用户调研、访谈、观察等方法获取用户反馈。基于用户数据进行分析，为设计提供依据。

5. 团队协作

与团队成员密切合作，了解项目的整体需求和技术限制。参与团队的讨论和评审，分享设计思路，接受他人的建议和批评。

6. 持续学习和改进

关注设计领域的新技术、新工具和新趋势，不断更新自己的知识和技能。对自己的设计作品进行反思和总结，不断改进设计方法和流程。

2. 技术总结和文献综述

2.1 Web 平台和客户端技术概述

Web 之父 Tim Berners-Lee 在发明 Web 的基本技术架构以后，就成立了 W3C 组织，该组织在 2010 年后推出的 HTML5 国际标准，结合欧洲 ECMA 组织维护的 ECMAScript 国际标准，几乎完美缔造了全球开发者实现开发平台统一的理想，直到今天，科学家与 Web 行业也还一直在致力于完善这个伟大而光荣的理想^[1]。学习 Web 标准和 Web 技术，学习编写 Web 程序和应用有关工具，最终架构一套高质量代码的跨平台运行的应用，是我的毕设项目应用的技术路线。

2.1.1 历史

1989 年，蒂姆·伯纳斯-李爵士发明了万维网(见最初的提案)。他创造了“万维网”这个词，并在 1990 年 10 月编写了第一个万维网服务器“httpd”和第一个客户端程序(一个浏览器和编辑器)“WorldWideWeb”。他编写了“超文本标记语言”(HTML)的第一个版本，这种文档格式语言具有超文本链接的功能，成为 Web 的主要发布格式。随着 Web 技术的传播，他对 uri、HTTP 和 HTML 的最初规范进行了改进和讨论。

2.1.2 万维网联盟

1994 年，在许多公司将越来越多的资源投入网络的敦促下，万维网联盟决定成立。蒂姆·伯纳斯-李爵士开始领导网络联盟团队的基本工作，以促进一个一致的架构，以适应快速发展的网络标准，用于构建网站、浏览器、设备，以体验网络所提供的一切。在创建万维网联盟时，蒂姆·伯纳斯-李爵士创建了一个同行社区。Web 技术已经发展得如此之快，以至于组建一个组织来协调 Web 标准变得至关重要。Tim 接受了麻省理工学院的邀请，他有与联盟打交道的经验，负责托管 W3C。他从一开始就要求 W3C 具有全球性的足迹。

2.1.3 Web 平台与 Web 编程

让我们先简单介绍一下 Web，也就是万维网的缩写。大多数人说“Web”而不是“World Wide Web”，我们将遵循这一惯例。网络是文档的集合，称为网页，

由世界各地的计算机用户共享(大部分)。不同类型的网页做不同的事情,但至少,它们都在电脑屏幕上显示内容。所谓“内容”,我们指的是文本、图片和用户输入机制,如文本框和按钮。[2]Web 编程是一个很大的领域,不同类型的 Web 编程由不同的工具实现。所有的工具都使用核心语言 HTML,所以几乎所有的 web 编程书籍都在某种程度上描述了 HTML。这本教科书涵盖了 HTML5, CSS 和 JavaScript,所有的深度。这三种技术被认为是客户端 web 编程的支柱。使用客户端 web 编程,所有网页计算都在最终用户的计算机(客户端计算机)上执行。[3]

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

2.2 项目的增量式迭代开发模式

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

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

瀑布模型

需要专业团队完美的配合，从分析、设计到实施，最后到测试，任何阶段的开始必须基于上一阶段的完美结束。而这对于我们大多数普通开发者是不太现实的，作为小微开发者由于身兼数职，其实无法 1 次就能完美完成任何阶段的工作，比如在实施过程中，开发者会发现前面的设计存在问题，则必须在下一次迭代项目时改良设计。在当今开源的软件开发环境中，开发者在软件的开发中总是在不断地优化设计、重构代码，持续改进程序的功能和代码质量。因此在本项目的开发中，也采用了增量模型的开发模式^[5]。本项目中我一共做了六次项目的开发迭代，

增量模型

在增量模型中，软件是按一系列步骤开发的。开发人员首先完成整个系统的简化版本。这个版本代表整个系统，但不包括细节。图中显示了增量模型概念。在第二个版本中，添加了更多的细节，而一些未完成，并再次测试系统。如果有问题，开发人员就会知道问题出在新功能上。在现有系统正常工作之前，它们不会添加更多的功能。这个过程一直持续到添加了所有需要的功能。如下图 2-1 所示：

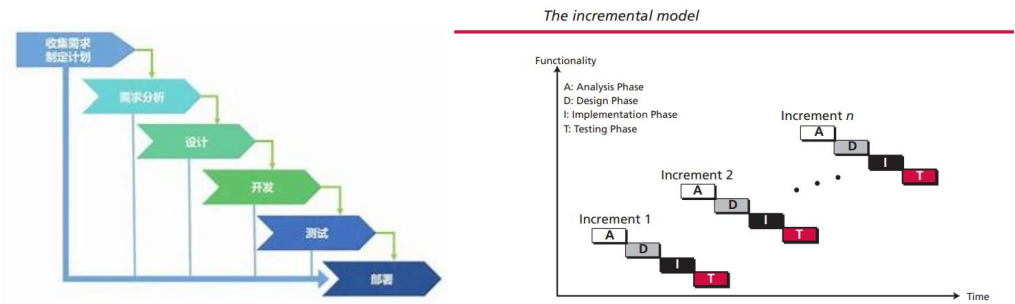


图 2—1

3. 内容设计概要

3.1 分析和设计

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

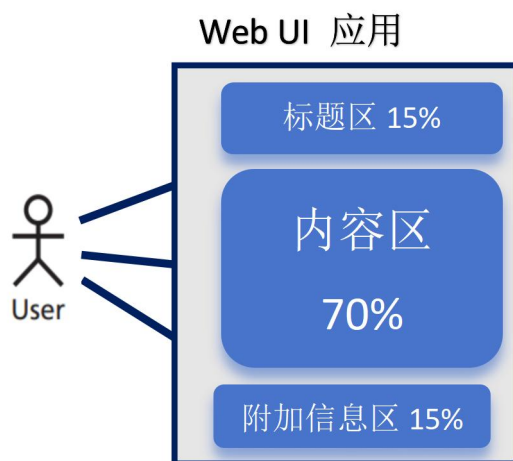


图 4-1 用例图

3.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
    《 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>
    Copyright XXX 江西科技师范大学 2024-2025
</footer>
```

二、CSS 代码编写如下：

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

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

3.3 项目的运行和测试

项目的运行和测试至少要通过二类终端，本文此处仅给出 PC 端用 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图 4-3 的二维码，运行测试本项目的第一次开发的阶段性效果。

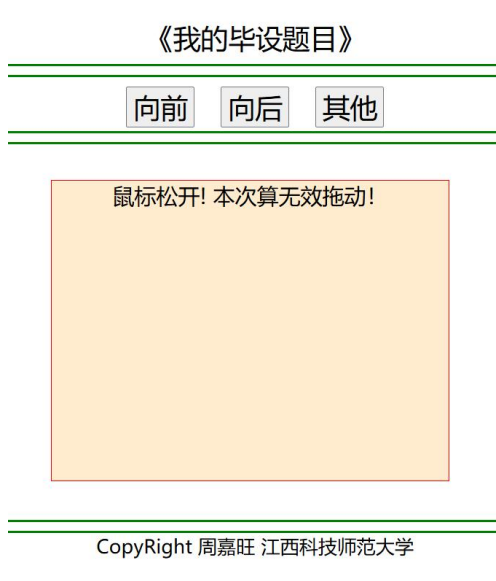


图 4-2 PC 端运行效果图

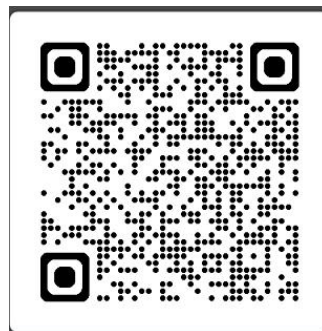


图 4-3 移动端二维码

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。

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ mkdir webUI
$ cd webUI
$ git init
$ git config user.name 江科师大周嘉旺
$ git config user.email 3220573793@qq.com
$ touch index.html myCss.css
```

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

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

成功提交代码后，gitbash 的反馈如下所示：

```
L@LAPTOP-MUCKLR31 MINGW64 ~/master11jh.github.io (ma
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html
        modified:   myCSS.css
```

项目代码仓库自此也开启了严肃的历史记录，我们可以输入日志命令查看，

```
$ git log
```

4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计

响应式设计——适应显示硬件

计算机所使用的显示硬件千差万别，显示器的大小和分辨率取决于成本。设计师们选择让网页给出一般的布局准则，并允许浏览器选择如何在给定的计算机上显示页面，而不是为每种类型的显示提供每个网页的版本。因此，一个网页不能提供很多细节。例如，网页的作者可以指定一组句子组成一个段落，但作者不能指定诸如一行的确切长度或是否缩进段落开头等细节。[1]

允许浏览器选择显示细节会产生一个有趣的结果：当通过两个浏览器或在硬件不同的两台计算机上浏览时，网页可能会显示不同的内容。如果一个屏幕比另一个屏幕宽，则可以显示的文本行的长度或图像的大小不同。重点是：网页给出了关于期望呈现的一般指导方针；浏览器在显示页面时选择详细信息。因此，同一网页在两台不同的计算机或不同的浏览器上显示时可能会略有不同。[1]

分析移动互联时代的多样化屏幕的需求。

4.1 分析与设计

本项目在上一次的“三段论”方式开展内容设计的基础上，用一个标题性信息展示 logo 或文字标题，吸引用户的注意力，接着在标题下方添加了一个按键区域，按键区设计了三个导航按钮，可以为用户提供导航的便利，然后展现的主要区域是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的主题，最后则是足部的附加信息，用来显示一些用户可能关心的细节变化。如下图所示：

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

4.2 实现代码

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

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

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

main{
```

```

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

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

```

代码块 4-1

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

```

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

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

```

代码块 4-2

4.3 项目的运行和测试

本文通过 PC 平台,用 chrome 浏览器打开本阶段上传至 http 服务器的代码,网址 (<https://20216342.github.io/1.2.html>),效果图如图 4-3 所示。还可以通过手机平台,用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址 (<https://20216342.github.io/1.2.html>),效果图如图 4-4 所示。表明目前项目运行正常,达到了项目初步的目标。移动端可以使用图 4-5 的二维码,查看代码运行效果。

代码的运行效果图:



图 4-3 PC 平台运行效果图



图 4-4 手机平台效果运行图



图 4-5 移动端二维码入口

4.4 代码提交

本阶段完成的代码存放在 1index.html 和 1myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 4-6 所示：

```
$ git add 1myCSS.css
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   1index.html
    new file:   1myCSS.css
```

图 4-6 本次代码提交的文件列表

命令行如下：

```
git add 1index.html 1myCSS.css
git commit -m 移动互联网时代的响应式设计和窄屏代码实现
```

本次代码提交成功，结果如图 4-7 所示：

```
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git config user.email 2814044280@qq.com

L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git commit -m移动互联网时代的响应式设计和窄屏代码实现
[main de7c402] 移动互联网时代的响应式设计和窄屏代码实现
3 files changed, 91 insertions(+)
 create mode 100644 1index.html
 create mode 100644 1myCSS.css
 create mode 100644 index.1
```

图 4-7 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示:

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.1 移动互联网用户终端

移动互联网时代的特点之一就是用户终端的多样性。用户可以使用各种设备访问互联网,包括手机、电脑等智能设备。这些设备的屏幕尺寸、分辨率、输入方式等有所不同,因此网页和应用的设计需要能够适应这些多样化的设备。响应式设计 (Responsive Design) 是一种网页设计和开发的方法,旨在使网页能够响应用户的行为和环境,包括屏幕尺寸、平台和方向。这种设计方式确保了网页在不同设备上都能提供良好的用户体验,无论用户使用的是智能手机、平板电脑、笔记本电脑还是台式电脑。

CSS 语言和 JavaScript 语言实现响应式设计,加入了#aid 和#bookface,首次使用了 position 用法。如码块 5-1 所示:

```
*{
  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;
}

```

码块 5-1

用 JavaScript 实现了软件界面的全屏设置。如代码块 5-2 所示：

```

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 $

```

码块 5-2

5.2 项目的运行和测试

本文通过 PC 平台，用 chrome 浏览器打开本阶段上传至 http 服务器的代码，网址 (<https://20216342.github.io/1.3.html>)，效果图如图 5-3 所示。还可以通过手机平台，用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址 (<https://20216342.github.io/1.3.html>)，效果图如图 5-4 所示。表明目前项目运行正常，达到了项目初步的目标。移动端可以使用图 5-5 的二维码，查看代码运行效果。

代码的运行效果图：



图 5-3 PC 端运行效果图



图 5-4 手机端运行效果图

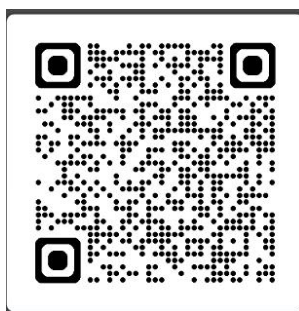


图 5-5 移动端二维码入口

5.3 代码提交

本阶段完成的代码存放在 2index.html 和 2myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 5-6 所示：

```
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git add 2index.html 2myCSS.css

L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file:   2index.html
new file:   2myCSS.css
```

图 5-6 本次代码提交的文件列表

命令行如下：

```
git add 2index.html 2myCSS.css
git commit -m 适用移动互联网时代的响应式设计
```

本次代码提交成功，结果如图 5-7 所示：

图 5-7 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：

6. 个性化 UI 设计中对鼠标交互的设计开发

6.1 个性化 UI 设计

在个性化 UI 设计中，为了确保用户体验的一致性，需要为触屏和鼠标交互设计一套共同的代码逻辑。这意味着无论是通过触摸屏还是鼠标，用户都能以相似的方式与 UI 进行交互。以下是如何使用一套代码逻辑同时为触屏和鼠标建立对象模型的方法，利用 CSS 语言和 JavaScript 语言完成设计，代码块如图 6-1 所示：

```
*{
  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% ;
}

```

CSS 块码 6-1

用 JavaScript 实现了软件界面的全屏设置。如代码块 6-2 所示：

```

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 $

```

码块 6-2

6.2 项目的运行和测试

本文通过 PC 平台，用 chrome 浏览器打开本阶段上传至 http 服务器的代码，网址 (<https://20216342.github.io/1.4.html>)，效果图如图 6-3 所示。还可以通过手机平台，用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址 (<https://20216342.github.io/1.4.html>)，效果图如图 6-4 所示。表明目前项目运行正常，达到了项目初步的目标。移动端可以使用图 6-5 的二维码，查看代码运行效果。

代码的运行效果图：



图 6-3 PC 端运行效果图



图 6-4 手机端运行效果图



图 6-5 移动端二维码入口

6.3 代码提交

本阶段完成的代码存放在 3index.html 和 3myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 6-6 所示：

```
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git add 3index.html 3myCSS.css

L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   3index.html
    new file:   3myCSS.css
```

图 6-6 本次代码提交的文件列表

命令行如下：

```
git add 3index.html 3myCSS.css
git commit -m 个性化 UI 设计中鼠标模型
```

本次代码提交成功，结果如图 6-7 所示：



```
@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git commit -m个性化UI设计中鼠标模型
[main f220dba] 个性化UI设计中鼠标模型
2 files changed, 94 insertions(+)
create mode 100644 3index.html
create mode 100644 3myCSS.css
```

图 6-7 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：

7. 对触屏和鼠标的通用交互操作的设计开发

设计一套代码逻辑，使其能够同时适用于触屏和鼠标的交互操作，需要考虑两种输入设备的共性和差异。在软件工程中，这通常通过抽象层来实现，使得上层应用逻辑可以忽略底层的具体输入设备类型。以下是实现这一目标的基本步骤：首先，需要定义一个通用的交互接口（例如，`IInputDevice`），它包含了所有输入设备共有的基本操作，如点击、移动、拖拽等。这个接口应该足够抽象，以便能够适应不同类型的输入设备。接着，为每种输入设备（触屏和鼠标）实现这个通用接口。每种设备的具体实现会根据其特性有所不同。在实际应用中，需要能够识别用户当前使用的输入设备，并选择合适的模型来处理交互。这可以通过检测用户代理（`User Agent`）或其他方式来实现。在处理用户交互时，需要将不同设备的原始事件（如触屏的 `touchstart`、`touchmove` 和鼠标的 `mousedown`、`mousemove`）适配为通用接口中定义的方法。最后，需要对上述代码进行优化和测试，确保在不同设备和浏览器上都能正常工作。这可能包括对不同设备和操作系统的兼容性测试、性能优化等。。通过这种方式，我们可以用一套代码逻辑同时为触屏和鼠标建立对象模型，从而实现跨设备的交互操作。这样的设计不仅提高了代码的复用性，还使得应用更加易于维护和扩展。

以下是如何使用一套代码逻辑同时为触屏和鼠标建立对象模型的方法，利用 CSS 语言和 JavaScript 语言完成设计，代码块如图 7-1 所示：

```
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' ;
    }
});

```

JavaScript 块码 7-1

7.2 项目的运行和测试

本文通过 PC 平台,用 chrome 浏览器打开本阶段上传至 http 服务器的代码,网址 (<https://20216342.github.io/1.5.html>),效果图如图 7-2 所示。还可以通过手机平台,用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址 (<https://20216342.github.io/1.5.html>),效果图如图 7-3 所示。表明目前项目运行正常,达到了项目初步的目标。移动端可以使用图 7-4 的二维码,查看代码运行效果。

代码的运行效果图:

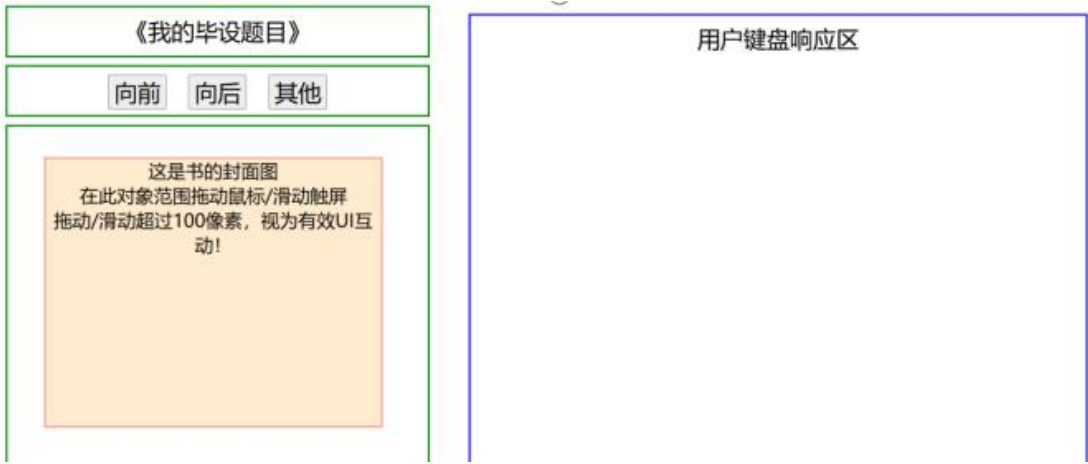


图 7-2 PC 端运行效果图



图 7-3 手机端运行效果图

图 7-4 移动端二维码入口

7.3 代码提交

本阶段完成的代码存放在 4index.html 和 4myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 7-5 所示：

```
$ git add 4index.html 4myCSS.css
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   4index.html
    new file:   4myCSS.css
```

图 7-5 本次代码提交的文件列表

命令行如下：

```
git add 4index.html 4myCSS.css
git commit -m UI 的个性化键盘交互控制的设计开发
```

本次代码提交成功，结果如图 7-6 所示：

```
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git commit -m对触屏和鼠标的通用交互操作的设计开发
[main 8efe5d3] 对触屏和鼠标的通用交互操作的设计开发
2 files changed, 106 insertions(+)
create mode 100644 5index.html
create mode 100644 5myCSS.css
```

图 7-6 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：

8.UI 的个性化键盘交互控制的设计开发

8.1 设计和开发 UI 个性化键盘

在设计和开发 UI 的个性化键盘交互控制时，可以利用 `keydown` 和 `keyup` 事件来捕获和处理键盘交互。这两个事件是键盘交互的底层事件，它们分别在用户按下和释放键盘按键时触发。通过精细地控制这些事件，可以为未来的 UI 提供强大且灵活的键盘功能。分为 `keydown` 和 `keyup` 事件的基础，`keydown` 事件：当用户按下键盘上的任意键时触发，无论该键是否产生字符值。它通常用于处理按键的按下动作，如游戏中的连续移动。`keyup` 事件：当用户释放键盘上的按键时触发。它通常用于处理那些需要在按键释放时执行的动作，如输入验证或命令执行。为了捕获这些事件，可以在 DOM 元素上添加事件监听器。通常，我们会在文档级别或特定的输入元素上添加这些监听器。在某些情况下，你可能需要阻止键盘事件的默认行为。例如，你不希望按箭头键时滚动页面，可以在事件处理函数中调用在设计键盘交互时，需要考虑到不同浏览器和操作系统的兼容性。虽然 `keydown` 和 `keyup` 事件在现代浏览器中得到了很好的支持，但在旧版本浏览器中可能会有不同的行为。因此，进行全面的跨浏览器测试是必要的。在设计键盘交互时，还需要考虑到无障碍性（Accessibility）。确保所有的功能和操作都可以通过键盘访问，这对于使用键盘导航的用户（如视力受损的用户）来说是非常重要的。通过探索和利用 `keydown` 和 `keyup` 事件的底层能力，可以为 UI 的键盘交互提供强大的基础，实现更加丰富和个性化的用户体验。在设计时，应该充分考虑用户的实际需求，创造出既直观又高效的键盘交互方式。

因为系统中只有一个键盘，所以我们在部署代码时，把键盘事件的监听设置

在 DOM 文档最大的可视对象——body 上,通过测试,不宜把键盘事件注册在 body 内部的子对象中。代码如下所示:

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

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

```
function printLetter(k){
    if (k.length > 1){ //学生须研究这个逻辑的作用
        return false ;
    }
    let puncs =
['~','`','!','@','#','$','%','^','&','*','(',')','-','_','+','=',' ','.',',',';',':','<','>','?','/',' ','\','\"'];
    if ( (k >= 'a' && k <= 'z') || (k >= 'A' && k <= 'Z')
|| (k >= '0' && k <= '9')) {
        console.log("letters") ;
        return true ;
    }
    for (let p of puncs ){
        if (p === k) {
            console.log("puncs") ;
            return true ;
        }
    }
    return false ;
    //提出更高阶的问题,如何处理连续空格和制表键 tab?
} //function printLetter(k)
```



```
});
```

码块 8-1

8.2 项目的运行和测试

本文通过 PC 平台,用 chrome 浏览器打开本阶段上传至 http 服务器的代码,网址 (<https://20216342.github.io/1.6.html>),效果图如图 8-2 所示。还可以通过手机平台,用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址 (<https://20216342.github.io/1.6.html>),效果图如图 8-3 所示。表明目前项目运行正常,达到了项目初步的目标。移动端可以使用图 8-4 的二维码,查看代码运行效果。

代码的运行效果图:

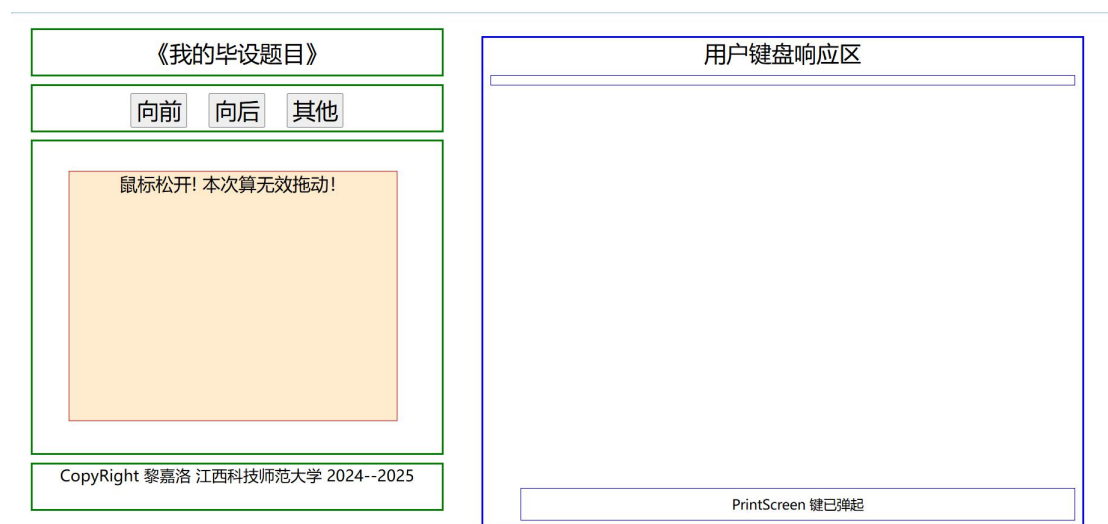




图 8-3 手机端效果运行图

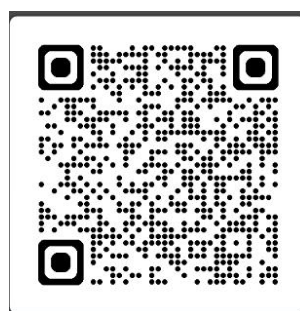


图 8-4 移动端二维码入口

8.3 代码提交

本阶段完成的代码存放在 5index.html 和 5myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 8-5 所示：

命令行如下：

```
git add 5index.html 5myCSS.css
git commit -m 对触屏和鼠标的通用交互操作的设计开发
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git add 5index.html 5myCSS.css
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   5index.html
    new file:   5myCSS.css
```

图 8-5 本次代码提交的文件列表

本次代码提交成功，结果如图 8-6 所示：

```
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (mai
$ git commit -mUI的个性化键盘交互控制的设计开发
[main 8d122f1] UI的个性化键盘交互控制的设计开发
2 files changed, 96 insertions(+)
create mode 100644 4index.html
create mode 100644 4myCSS.css
```

图 8-6 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：

9. 谈谈本项目中的高质量代码

这是一本关于计算机教学的书。今天，电脑就像螺丝刀一样常见，但它们要复杂得多，让它们做你想让它们做的事情并不总是那么容易。如果你给你的电脑的任务是一个常见的，很容易理解的任务，比如显示你的电子邮件或像计算器一样工作，你可以打开适当的应用程序并开始工作。但对于独特的或开放式的任务，可能没有应用程序。

这就是编程可能发挥作用的地方。编程是构建程序的行为——一组告诉计算机该做什么的精确指令。因为计算机是愚蠢的、迂腐的野兽，编程从根本上来说是乏味和令人沮丧的。幸运的是，如果你能克服这个事实，甚至享受用愚蠢的机器可以处理的方式思考的严谨性，编程是有回报的。它可以让你在几秒钟内完成手工永远做不完的事情。它是一种让你的计算机工具做它以前不能做的事情的方法。它还提供了一个很好的抽象思维练习。[6]

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。

（围绕着抽象定义函数、代码块、模型设计以及降低全局变量的使用来写）

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

当我们谈到命令行时，我们实际上指的是 shell。shell 是一个接受键盘命令并将其传递给操作系统执行的程序。几乎所有的 Linux 发行版都提供了一个来自 GNU 项目的 shell 程序，名为 bash。这个名字是 Bourne -again shell 的首字母缩略词，指的是 bash 是 sh 的增强替代品，sh 是 Steve Bourne 编写的原始 Unix shell 程序。[7]

像 Windows 一样，像 Linux 这样的类 unix 操作系统用所谓的分层目录结构来组织文件。这意味着它们被组织成树状的目录模式(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，子目录包含更多的文件和子目录，以此类推。[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

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 (*).

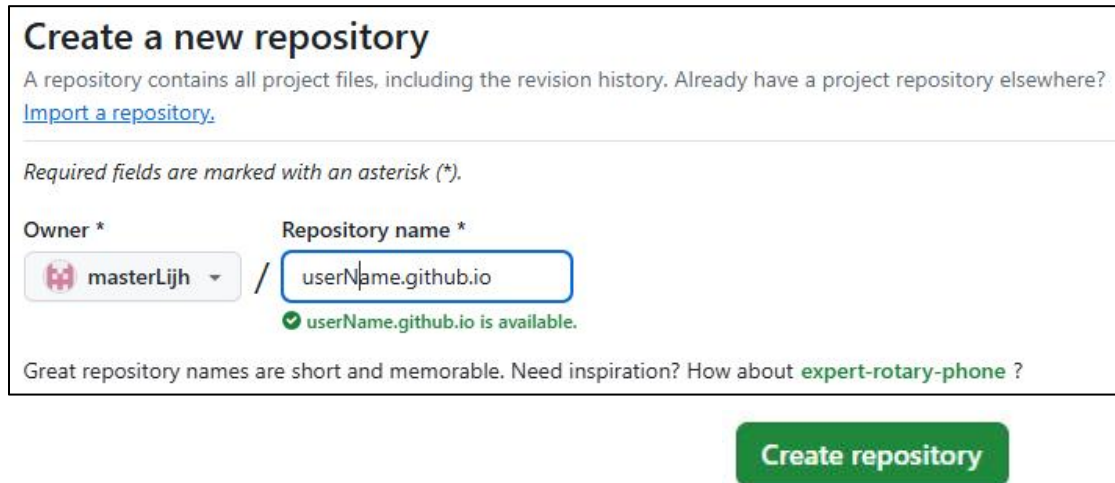
Owner *	Repository name *
 zjiaw	/ zjiaw
	✔ zjiaw is available.

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 *

 masterLijh ▾

/

Repository name *

userName.github.io

✓ userName.github.io is available.

Great repository names are short and memorable. Need inspiration? How about [expert-rotary-phone](#) ?

Create repository

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

10.4 设置本地仓库和远程代码仓库的链接

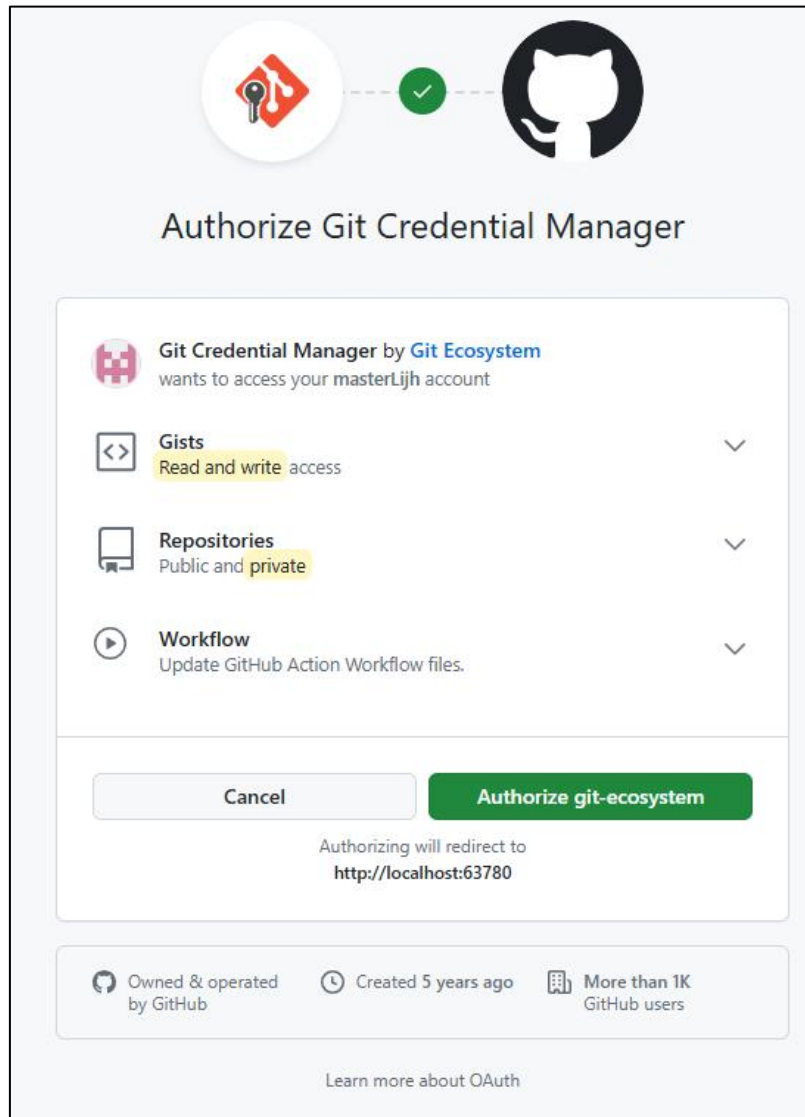
进入本地 webUI 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接

```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md
$ git init
$ git add README.md
$ git commit -m "这是我第一次把代码仓库上传至 gitHub 平台"
$ git branch -M main
$ git remote add origin
    https://github.com/20216342/lijialuo.github.io.git
$ git push -u origin main
```

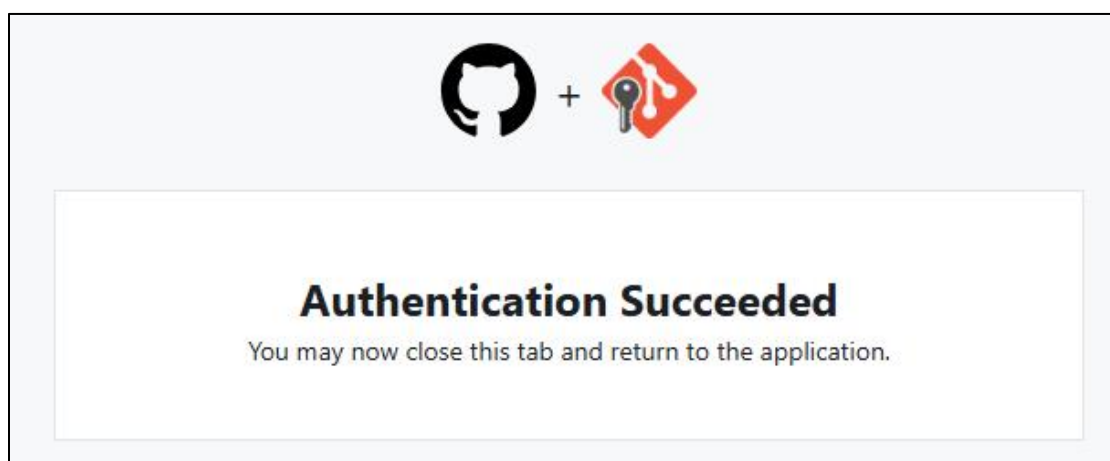
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



再次确认授权 **gitBash** 拥有访问改动远程代码的权限，如下图所示：



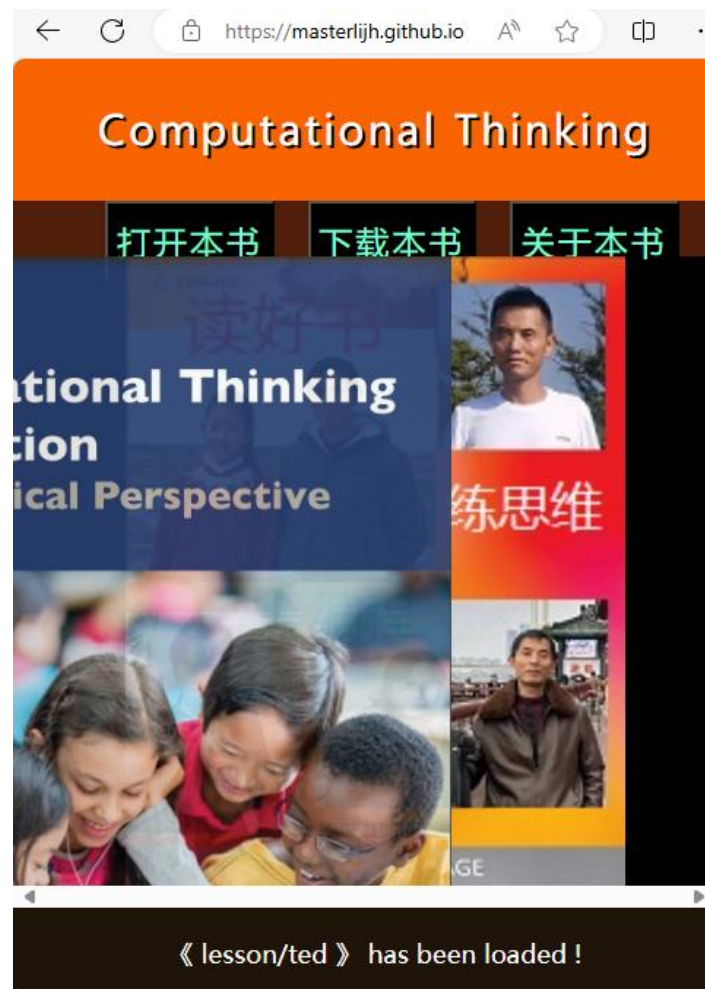
最后，GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。



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

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

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



全文完成，谢谢！

参考文献

- [1] W3C. W3C's history. W3C Community. [EB/OL]. <https://www.w3.org/about/>. <https://www.w3.org/about/history/>. 2023.12.20
- [2] Douglas E. Comer. The Internet Book [M] (Fifth Edition). CRC Press Taylor & Francis Group, 2019: 217-218
- [3] John Dean, PhD. Web programming with HTML5, CSS, and JavaScript[M]. Jones & Bartlett Learning, LLC. 2019: 2
- [4] John Dean, PhD. Web programming with HTML5, CSS, and JavaScript[M]. Jones & Bartlett Learning, LLC. 2019: xi
- [5] Behrouz Forouzan. Foundations of Computer Science[M](4th Edition). Cengage Learning EMEA, 2018: 274--275
- [6] Marijn Haverbeke. Eloquent JavaScript 3rd edition. No Starch Press, Inc, 2019.
- [7] William Shotts. The Linux Command Line, 2nd Edition [M]. No Starch Press, Inc, 245 8th Street, San Francisco, CA 94103, 2019: 3-7