江 西 科 技 师 范 大 学

课程设计（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：廖蓓林

学　　号：20213652

指导教师：李健宏

2024 年 6 月 18 日

目录

基于Web客户端技术的个性化UI设计和实现

摘要：HTML（HyperText Markup Language，超文本标记语言）是用来描述网页的一种语言，它不是一种编程语言，而是一种标记语言。它将Web带入一个成熟的应用平台，在这个平台上,视频、音频、图像、动画以及与设备的交互都进行了规范。在开发中利用HTML、CSS和JavaScript等技术实现设计稿中的视觉效果和交互功能，并注重响应式设计以适配不同设备。个性化功能的实现是关键，通过用户定制选项、智能推荐和动态内容更新等方式，满足用户的个性化需求。通过分析本次毕设任务，我们可以展开对程序设计和软件开发的研究和实践，广泛查阅相关技术书籍、开发者论坛和文献，设计开发了一个个性化的用户界面（UI）的应用程序。

本项目采用了git工具进行版本管理，利用gitbash工具把本项目的代码仓库上传到github上，再利用github提供的http服务器，本项目实现了UI应用在全球互联网的部署，我们可以通过地址和二维码便捷地跨平台高效访问这个程序。

关键字：html5；响应式设计；github；css

Abstract: HTML (HyperText Markup Language) is a language used to describe web pages, it is not a programming language, but a markup language. It brings the Web to a full-fledged application platform where video, audio, images, animation, and interaction with devices are regulated. In the development, HTML, CSS, JavaScript and other technologies are used to achieve visual effects and interactive functions in the design manuscript, and pay attention to responsive design to adapt to different devices. The realization of personalized function is the key, through user customization options, intelligent recommendation and dynamic content update, to meet the personalized needs of users. Through the analysis of this completion task, we can carry out the research and practice of programming and software development, extensively consult relevant technical books, developer forums and literature, and design and develop a personalized user interface (UI) application.

git tool was adopted for version management in this project, and gitbash tool was used to upload the code warehouse of this project to github. With the http server provided by github, this project realized the deployment of UI application on the global Internet, and we could conveniently and efficiently access this program across platforms through the address and two-dimensional code.

Keywords: html, responsive design, incremental model

1. 前言

个人对毕业设计和毕业论文的理解：毕业设计和毕业论文是高等教育阶段的重要组成部分，它们不仅是对学生专业知识掌握程度的检验，更是对学生独立思考、创新能力和实践能力的全面锻炼。以下是我对毕业设计和毕业论文的深入理解：

首先，毕业设计是一个综合性的实践过程。它要求学生根据所学专业知识，结合实际应用场景，提出一个具有实际意义的课题，并通过设计、实验、分析等一系列环节，最终完成一个具有创新性和实用性的项目。

此外，毕业设计和毕业论文还强调学生的独立思考和创新能力。在选题和设计过程中，学生需要发挥自己的想象力和创造力，提出新颖的观点和解决方案。

最后，毕业设计和毕业论文也是学生展示自己才华和能力的舞台。通过完成高质量的毕业设计和论文，学生可以展示自己的专业素养和实践能力，为未来的职业发展打下坚实的基础。

1.1 毕设任务分析

在我的毕业设计中，涉及的有关核心课程的理论包括：面向对象的程序设计语言、数据结构和算法、操作系统、软件工程等。以前这些核心课程供理论指导感觉非常抽象，加之基本上以理论知识为主，因此学完后我们感觉一直有所缺憾，本人与导师沟通后也一致认为，若能在实践层面应用这些核心课程的关键知识，则必然会在理解和技术二个维度提升自己的专业性。

我的毕设分为二个阶段完成，首先选择一条自己感兴趣的技术实践路线，把核心的技术加以整合学习，以导师的案例项目为参考，主要是理解好各个技术之间的关系，在项目中的作用和分工，更重要的是在项目实施中提升自己的写高质量的代码能力。

1.2 研学计划

研究计划是一份详尽的蓝图，为研究者指明了方向，确保了研究过程的有序性和高效性。在研究计划中，我们需要确定研究方向和目标、拥有文献综述与理论基础、研究内容与步骤、预期成果与贡献、实证研究法、案例研究法、模拟与仿真法、比较与归纳法。

研究计划让从事理工科的人们为实现复杂的工程而制定的计划或规划，而技术二字强调的是专业性，一般用本专业或行业的术语表达，而路线二字则强调逻辑性，一般表达了事物的因果关系或层次关系，我们常用技术路线来证明工程项目或研究的可行性。

具体安排和进度如下：

时间	任务
2024 年 3 月	确定论文题目
2024 年 4 月-5 月	编写项目
2024 年 5 月 16 日-5 月 24 日	编写论文内容和框架

2024 年 5 月 26 日	编写摘要
2024 年 6 月	修改论文格式

1.3 研究方法

研究的计划与研究方法，两者在学术研究中相辅相成，共同构成了研究工作的核心框架。研究方法则是实现研究计划的关键手段；它涉及在研究中发现新现象、新事物，或提出新理论、新观点，揭示事物内在规律的工具和手段。

研究方法一般按软件工程的标准来规范开发：1、结合自己的问题做出定义和分析；2、设计一套合适的技术解决方案；3、按解决方案设计流程和编写相关代码，实现技术部署；4、调试代码、测试软件、性能调优。其中第 3、4 步可以发现前面步骤的问题，因此可能会在第 2，3，4 步多次循环，发现和解决第 2 步的设计失误或第 3 步的代码错误。当然大部分工作是用在第 3 步的构建代码体系和落实软件架构的具体实施和细节。

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。

2.2.1 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。

2.2.2 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。

2.2.3 web 平台的 web 项目编程

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，如下图 2-1 所示：

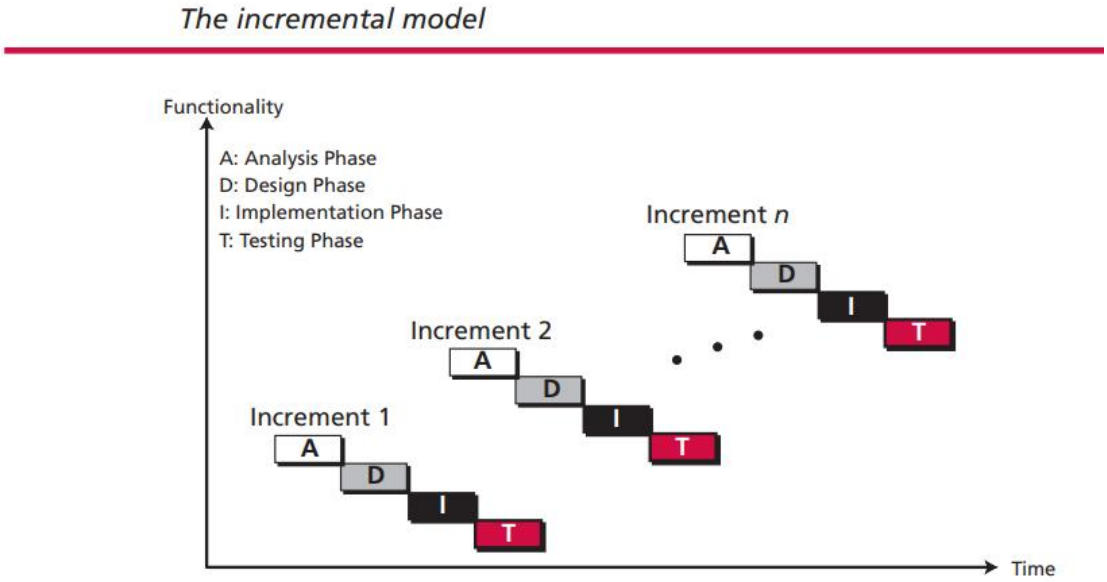


图 2-1 增量模型

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［5］。

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。对象模型如图 3-1 用例图、3-2 dom 树状图所示。

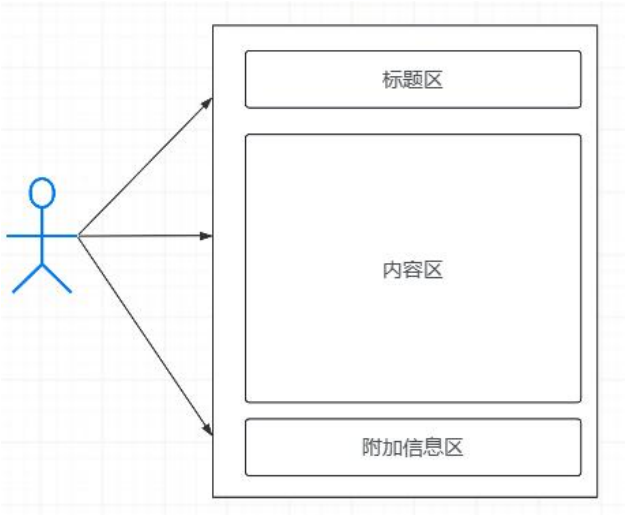


图 3-1 用例图

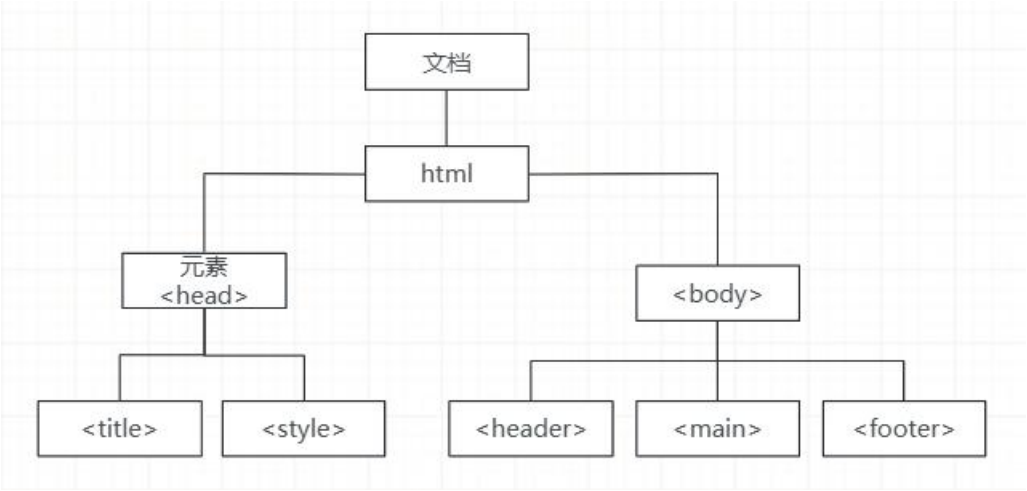


图 3-2 DOM 树状图

3.2 项目的实现和编程

3.2.1HTML 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>
```

3.2.2 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 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。

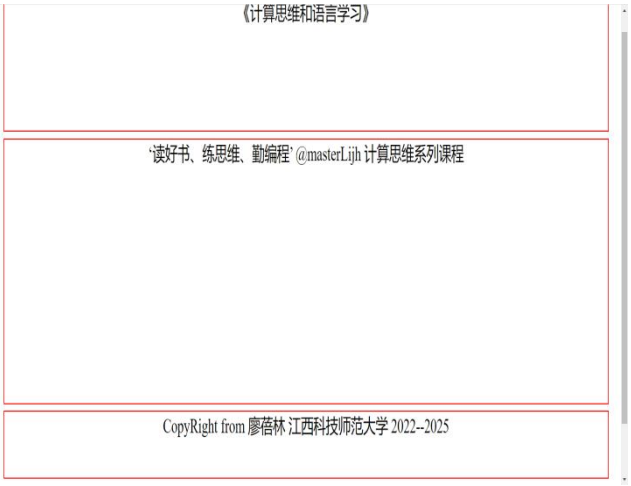


图 3-3 PC 端运行效果图

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：如开发者的名字和电子邮件。

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /d
$ mkdir webUI
$ cd webUI
$ git init
$ git config user.name liaobeilin603
$ git config user.email 2469766762@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：

```
86198@DESKTOP-02FV6IA MINGW64 /d/webUI (master)
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
[master (root-commit) 79b45f0] 项目第一版：“三段论”式的内容设计概要开发
 1 file changed, 44 insertions(+)
 create mode 100644 index.html
```

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：

```
86198@DESKTOP-02FV6IA MINGW64 /d/webUI (master)
$ git log
commit 79b45f0a60a1c555bdad41d7424130f1ac87b7b2 (HEAD -> master)
Author: liaobeilin603 <2469766762@qq.com>
Date:   Wed Jun 12 14:36:17 2024 +0800

    项目第一版："三段论"式的内容设计概要开发
```

4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计

4.1 分析和设计

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]。对象模型如下图 4-1 用例图、图 4-2 dom 树状图所示。

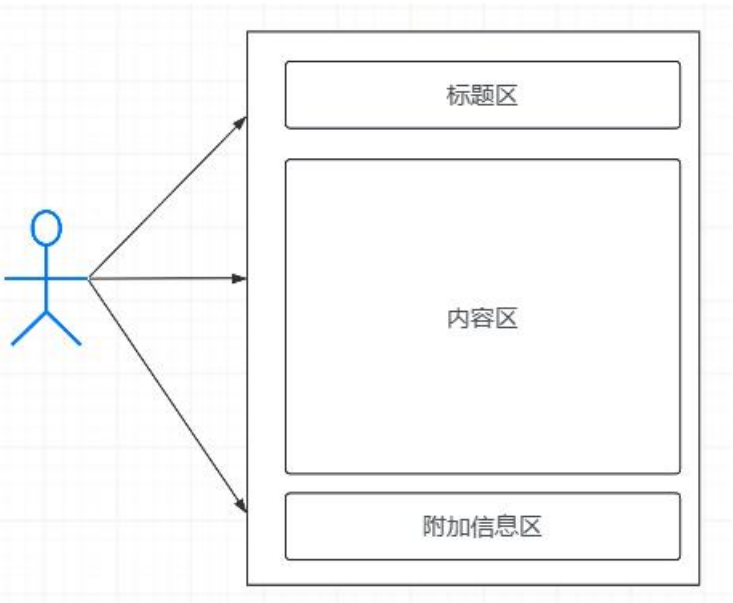


图 4-1 窄屏终端的响应式设计用例图

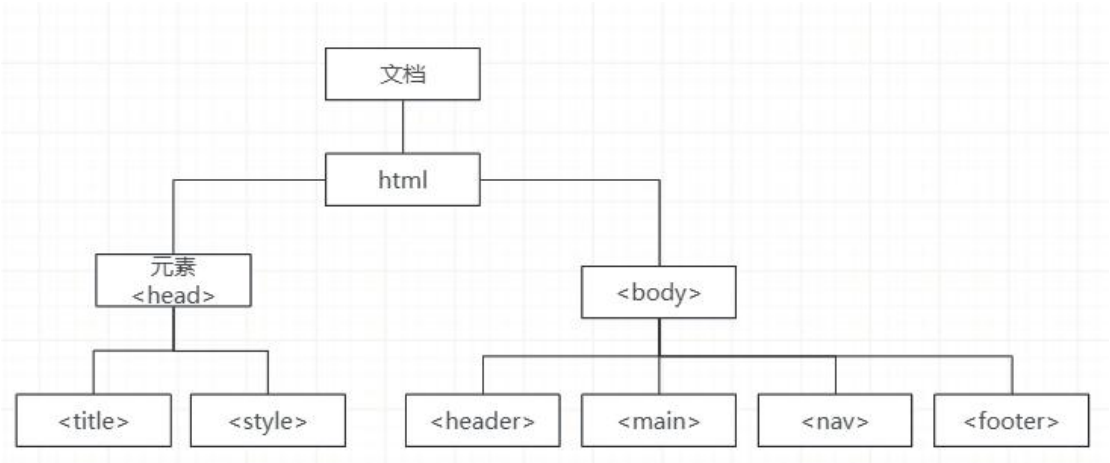


图 4-2 窄屏终端的响应式设计的 DOM 树状图

4.2 项目的实现和编程

4.2.1 CSS 代码编写如下

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 red solid;
       height:13%;
       background-color: blue;
       color: white;
  }
  nav{
       border:2px solid red ;
       height:7%;
  }
  main{

       height: 70%;
       background-image: url("../lesson/bitCoin.jpg");
       background-repeat: no-repeat;
```

```
        background-size: cover;
        background-position: center;
    }
    footer{
        border:2px red solid;
        height:10%;
        background-color: black;
        color: white;
    }
</style>
```

代码块 4-1

4.2.2 JavaScript 代码编写如下

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

项目的运行和测试通过二类终端，本文此处仅给出 PC 端用 Chrome 浏览器打开项目的结果，如下图 4-3 和 4-4 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 4-5 的二维码，运行测试本项目的第一次开发的阶段性效果。

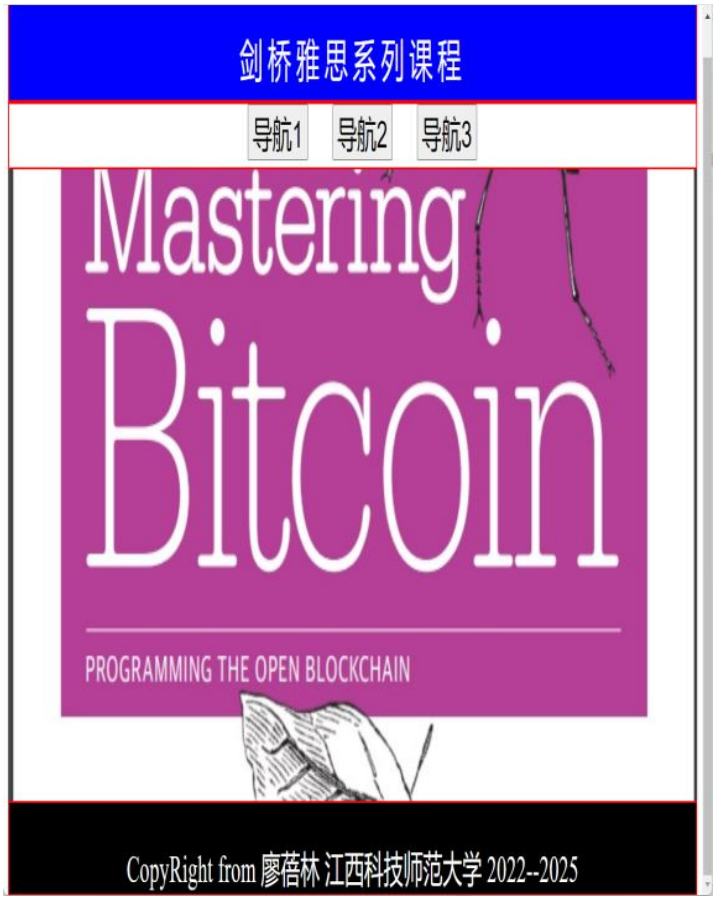


图 4-3 PC 端运行效果图

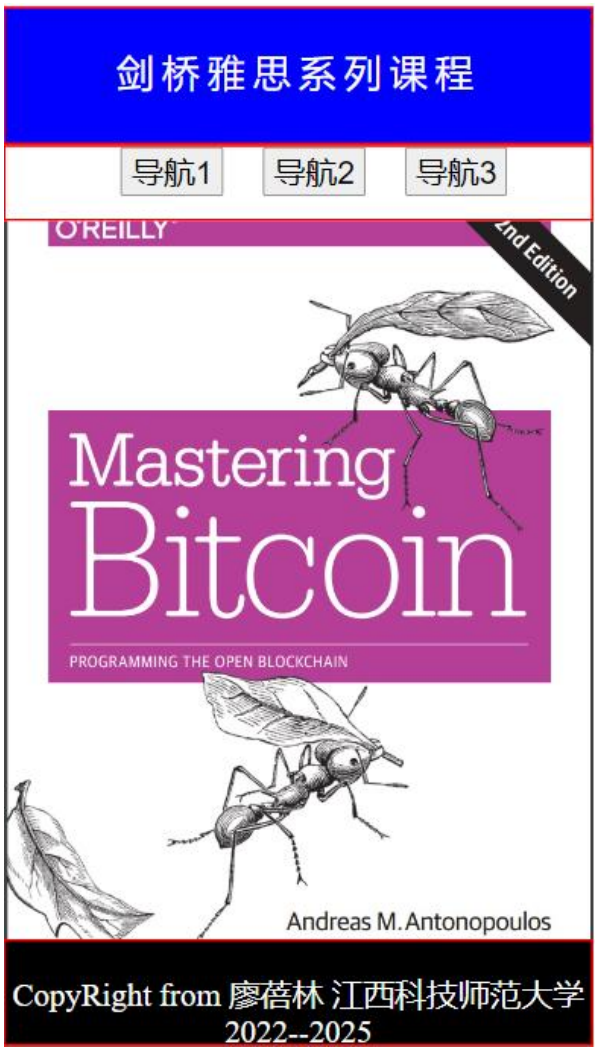


图 4-4 iPhone14 Pro max 端运行效果图

图 4-5 移动端二维码

4.4 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，作为项目的第二次迭代，在代码提交和版本管理环节，我们实现了 pc 端和手机端的响应式设计，向 github 提交了代码。进入 gitBash 命令行后，再进入文件夹，按次序输入下图 4-6 的命令：

```
86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git add 1.2.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git commit -m 项目第二版“窄屏终端的响应式设计”
[master e11f74e] 项目第二版“窄屏终端的响应式设计”
 1 file changed, 14 insertions(+), 11 deletions(-)

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git log
commit e11f74e0c8d6825f0bb02ed5fb2bdc38e193475f (HEAD -> master)
Author: githubssh_key <2469766762@qq.com>
Date:   Thu Jun 13 11:04:29 2024 +0800

    项目第二版“窄屏终端的响应式设计”
```

图 4-6

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

应用响应式设计技术开发可适配窄屏和宽屏的 UI 是一个综合性的过程，它涉及到对用户体验的深入理解、对设计元素的灵活处理以及对技术的熟练运用。UI 设计应具备足够的灵活性，能够根据不同设备的屏幕尺寸和分辨率进行自适应调整，使用 HTML、CSS 和 JavaScript 等前端技术实现响应式 UI。对象模型如图 5-1 用例图、5-2 dom 树状图所示。

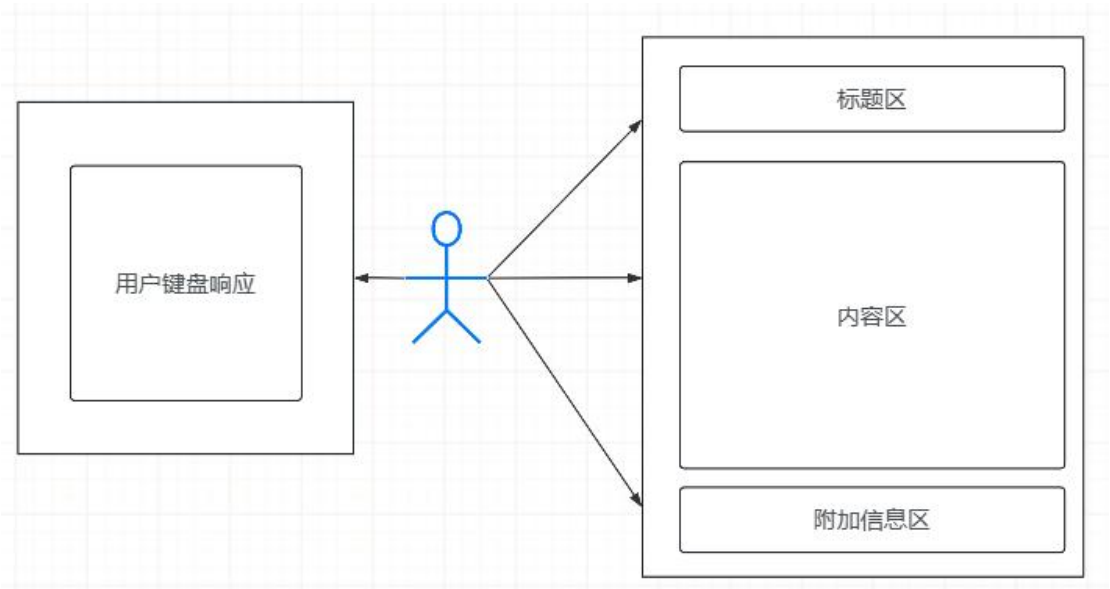


图 5-1 应用响应式设计技术开发可适配窄屏和宽屏的 UI 用例图

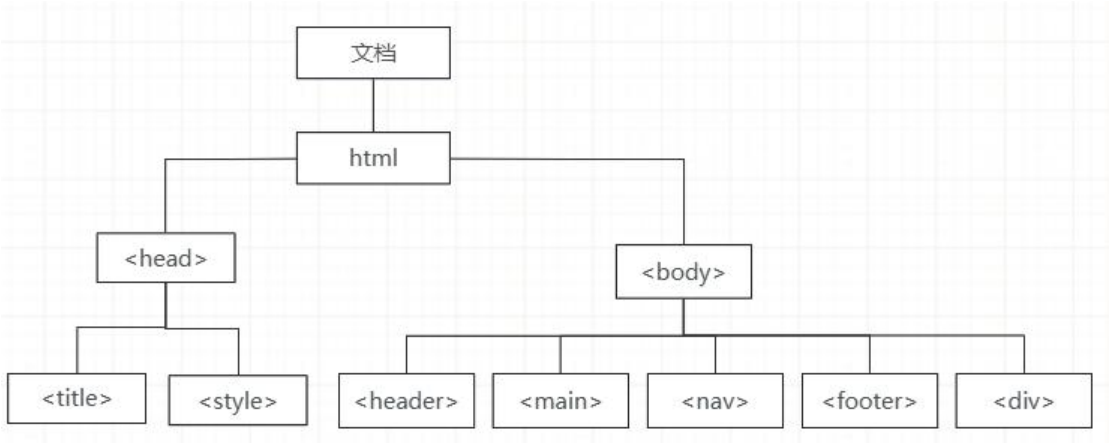


图 5-2 应用响应式设计技术开发可适配窄屏和宽屏的 UI 的 DOM 树状图

5.2 项目的实现和内容

5. 2. 1 css 代码编写如下

与上一阶段比较，本阶段增加 user-scalable=no，可以让移动设备的屏幕宽度和高度，与 window 对象的 innerWidth 和 innerHeight 精确对应。实现代码，如代码块 5-1 所示：

```
<meta name="viewport" content="width=device-width,initial-scale=1, user-scalable=no">
    <style>
      *{
        text-align: center;
        box-sizing: border-box ;
      }
        header,main,div#bookface,nav,footer{
          margin:1em;
        }
        header{
          border:2px red solid;
          height:13%;
          background-color: blue;
          color: white;
        }
        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;
    }
  </style>
```

代码块 5-1

5.2.2 JavaScript 代码编写如下

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 JavaScript 改变 body 对象的字体大小，这个属性可以影响其后代；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；尝试对鼠标设计 UI 控制和键盘按键的响应设计；使用 function $(ele){....}创建自己的自定义的$函数参数的数据类型。如代码块 5-2 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 + "px";
    document.body.style.height = UI.appHeight - 62 + "px";
  if(window.innerWidth<1000){

      $("aid").style.display='none';
  }
    $("aid").style.width=window.innerWidth-UI.appWidth-30+'px';
    $("aid").style.height= UI.appHeight-62+'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;
    let s="按键是："+k+" 编码是：";
    $("keyboard").textContent=s+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 $


    </script>
```

代码块 5-2

5.3 项目的运行和测试

项目的运行和测试通过二类终端，本文此处仅给出 PC 端用 Chrome 浏览器打开项目的结果，如下图 5-3 和 5-4 所示。由于本项目的阶段性文件已经上传 github 网站，移动端用户可以通过扫描图 5-5 的二维码，运行测试本项目的第一次开发的阶段性效果。

图 5-3 PC 端运行效果图

图 5-4 iPhone14 Pro max 端运行效果图

图 5-5 移动端二维码

5.4 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，作为项目的第三次迭代，在代码提交和版本管理环节，我们更新了代码，让项目更好的适配手机端和 pc 端，向 github 提交了代码。进入 gitBash 命令行后，再进入文件夹，按次序输入下图 5-6 的命令：

```
86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git add 1.3.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git commit -m 项目第三版“应用响应式设计技术开发可适配窄屏和宽屏的UI”
[master d103c01] 项目第三版“应用响应式设计技术开发可适配窄屏和宽屏的UI”
 1 file changed, 174 insertions(+)
 create mode 100644 exp/1.3.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git log
commit d103c01b3e3e4808f95359e27aaed1d5334283a8 (HEAD -> master)
Author: githubssh_key <2469766762@qq.com>
Date:   Thu Jun 13 11:06:57 2024 +0800

    项目第三版“应用响应式设计技术开发可适配窄屏和宽屏的UI”
```

图 5-6

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

在个性化 UI 设计中，鼠标交互的设计开发是一个至关重要的环节。鼠标作为用户与界面进行交互的主要工具之一，其交互设计的优劣直接影响到用户的使用体验和满意度。鼠标交互设计应该直观易懂，用户能够轻松理解并快速上手。避免设计过于复杂或晦涩难懂的交互方式，以免给用户带来困扰。而且，鼠标交互应该高效便捷，能够减少用户的操作步骤和时间成本。对象模型如图 6-1 用例图、6-2 dom 树状图所示。

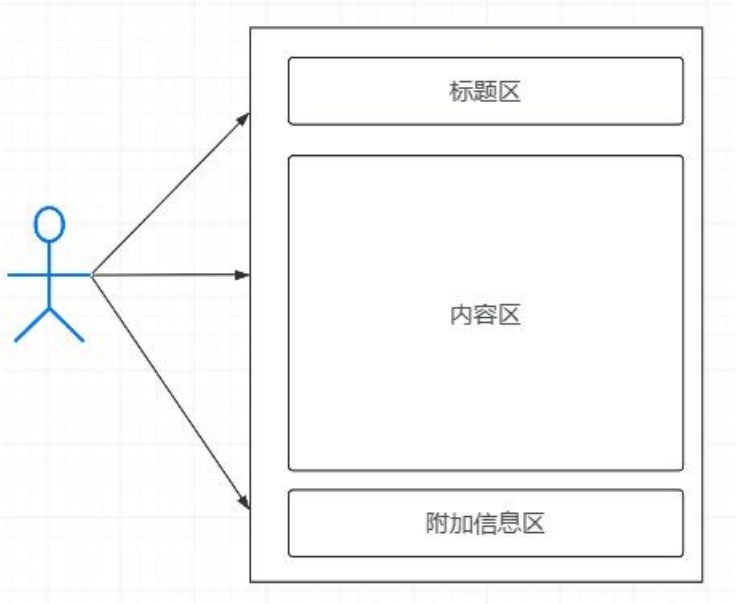


图 6-1 个性化 UI 设计中对鼠标交互的设计开发用例图

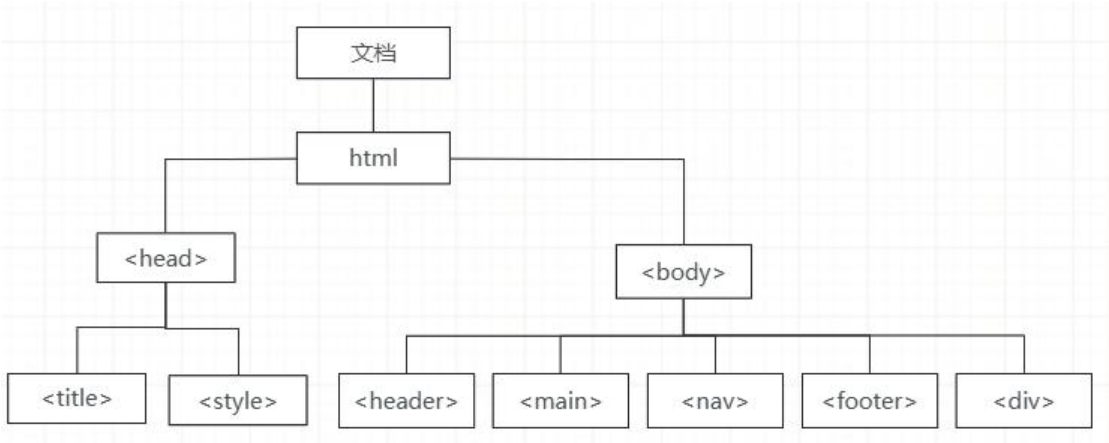


图 6-2 个性化 UI 设计中对鼠标交互的设计开发的 DOM 树状图

6.2 项目的实现和内容

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 JavaScript 尝试对鼠标设计 UI 控制，$("bookface").addEventListener("mousedown",function(ev){...}显示鼠标按下拖动时坐标；$("bookface").addEventListener("mouseup",function(ev){...}显示鼠标松开的坐标，在拖动不足 100 时，显示“这是无效拖动”，回到坐标 0 的位置；$("bookface").addEventListener("mouseout",function(ev){...}按下松开时的坐标，在拖动不足 100 时，显示“这是无效拖动”；$("bookface").addEventListener("mousemove",function(ev){...}显示鼠标拖动的距离。如代码块 6-1 所示：

```
    //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' ;
      }

    });
```

代码块 6-1

6.3 项目的运行和测试

项目的运行和测试通过二类终端，本文此处仅给出 PC 端用 Chrome 浏览器打开项目的结果，如下图 6-3 和 6-4 所示。由于本项目的阶段性文件已经上传 github 网站，移动端用户可以通过扫描图 6-5 的二维码，运行测试本项目的第一次开发的阶段性效果。

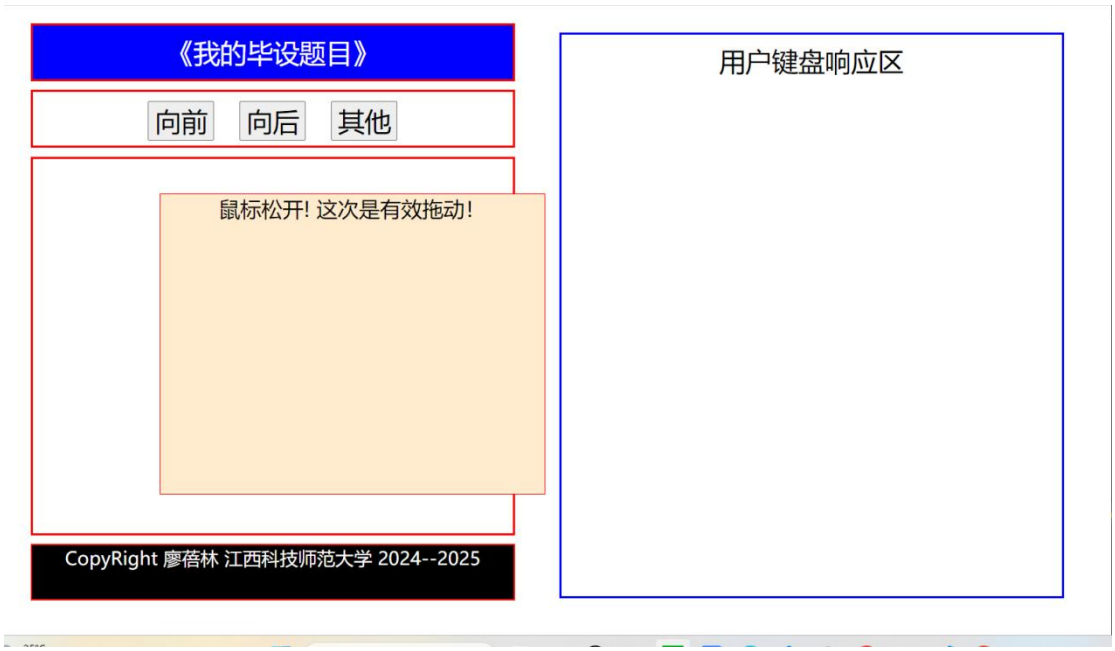


图 6-3 PC 端运行效果图

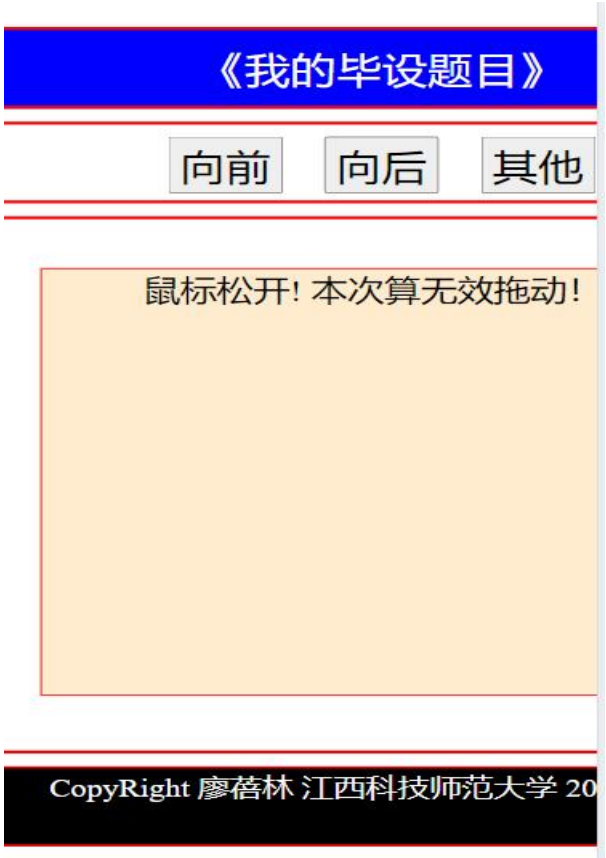


图 6-4 iPhone14 Pro max 端运行效果图

图 6-5 移动端二维码

6.4 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，作为项目的第四次迭代，在代码提交和版本管理环节，我们实现了个性化 UI 设计中对鼠标交互的设计开发，让项目更加人性化，向 github

提交了代码。进入 gitBash 命令行后，再进入文件夹，按次序输入下图 6-6 的命令：

```
86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git add 1.4.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git commit -m 项目第四版“个性化UI设计中对鼠标交互的设计开发”
[master f1bb639] 项目第四版“个性化UI设计中对鼠标交互的设计开发”
 1 file changed, 194 insertions(+)
 create mode 100644 exp/1.4.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git log
commit f1bb639e772f01776a81ef90d25dbd96c5c7f4a2 (HEAD -> master)
Author: githubssh_key <2469766762@qq.com>
Date:   Thu Jun 13 11:08:39 2024 +0800

    项目第四版“个性化UI设计中对鼠标交互的设计开发”
```

图 6-6

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

我们需要分析触屏和鼠标在操作特性上的差异。触屏交互更加直观和直接，用户可以通过手指触摸、滑动、缩放等手势来完成操作，这种交互方式在移动设备和触摸屏电脑上非常普遍。而鼠标交互则依赖于指针的移动和点击，具有更高的精度和稳定性，尤其在处理复杂任务和精确操作时表现更为出色。对于鼠标操作，可以通过优化鼠标指针的灵敏度、提供拖拽功能，提高操作的精度和效率。对象模型如图 7-1 用例图、7-2 dom 树状图所示。

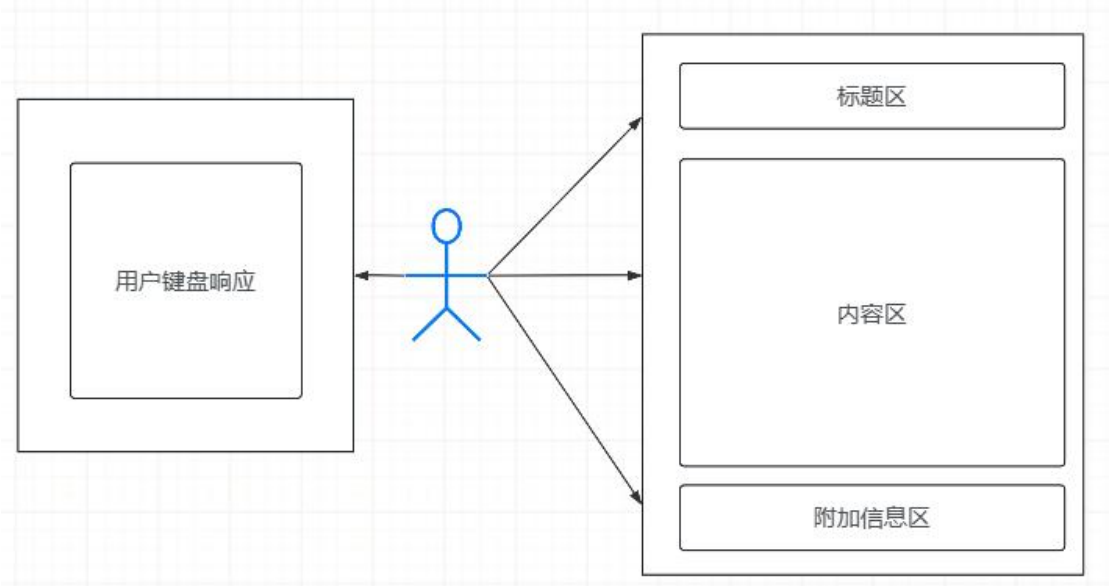


图 7-1 对触屏和鼠标的通用交互操作的设计开发用例图

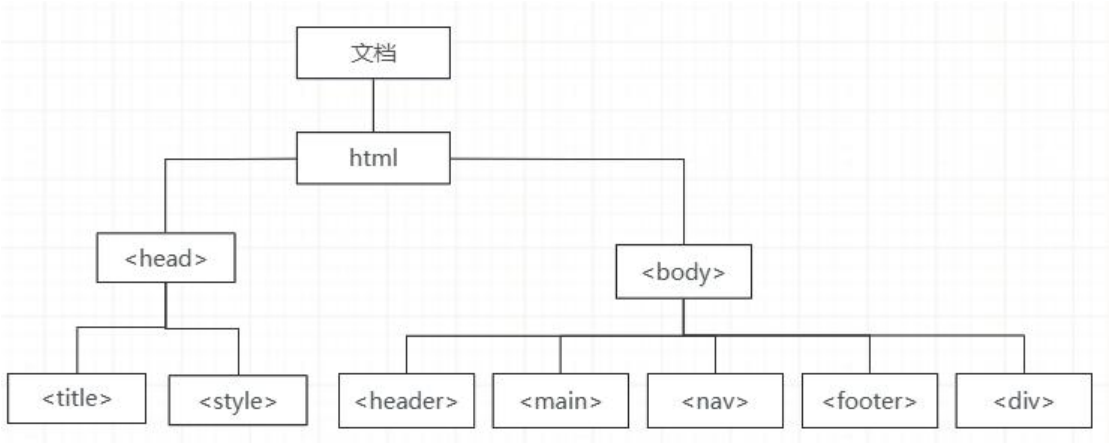


图 7-2 对触屏和鼠标的通用交互操作的设计开发的 DOM 树状图

7.2 项目的实现和内容

用汉语言来描述我们是如何实现的：与上一阶段比较，本阶段优化了代码，创建一个 Pointer 对象，设计一套代码同时对鼠标和触屏实现控制，let handleBegin = function(ev){...}，触屏事件开始，显示坐标；鼠标按下，显示坐标；let handleEnd = function(ev){...}，表示触屏事件结束，鼠标松开。let handleMoving = function(ev){...}，松开，鼠标正在滑动触屏，显示滑动距离；不松开，鼠标正在拖动鼠标，显示距离；$("body").addEventListener("keypress", function(ev){...}显示键盘按键。如代码块 7-1 所示：

```
    //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
```

代码块 7-1

7.3 项目的运行和测试

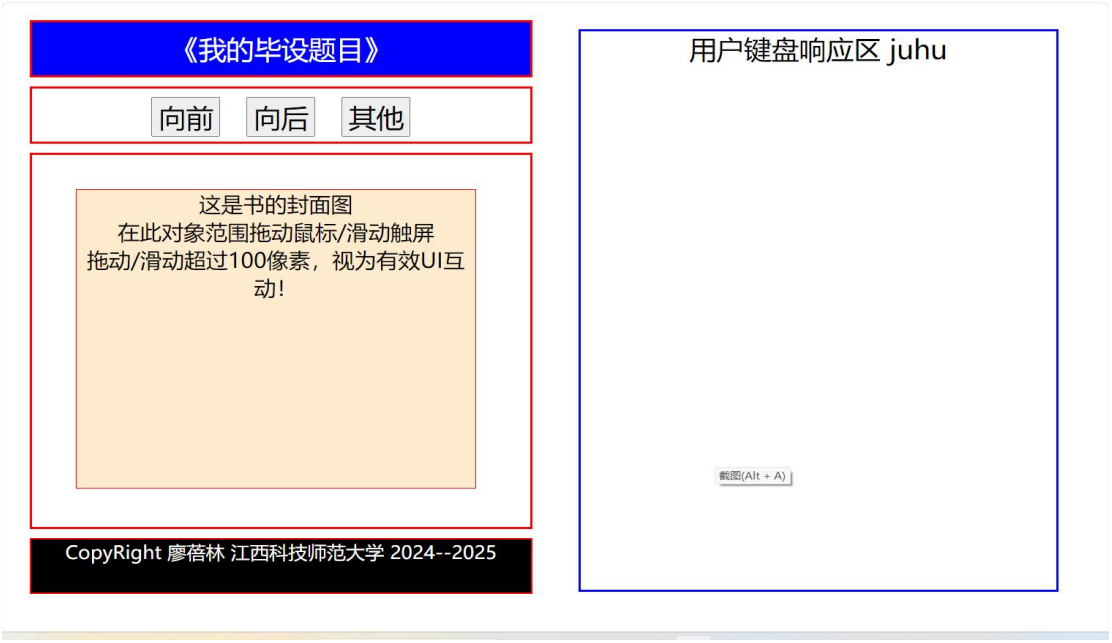


图 7-3 PC 端运行效果图

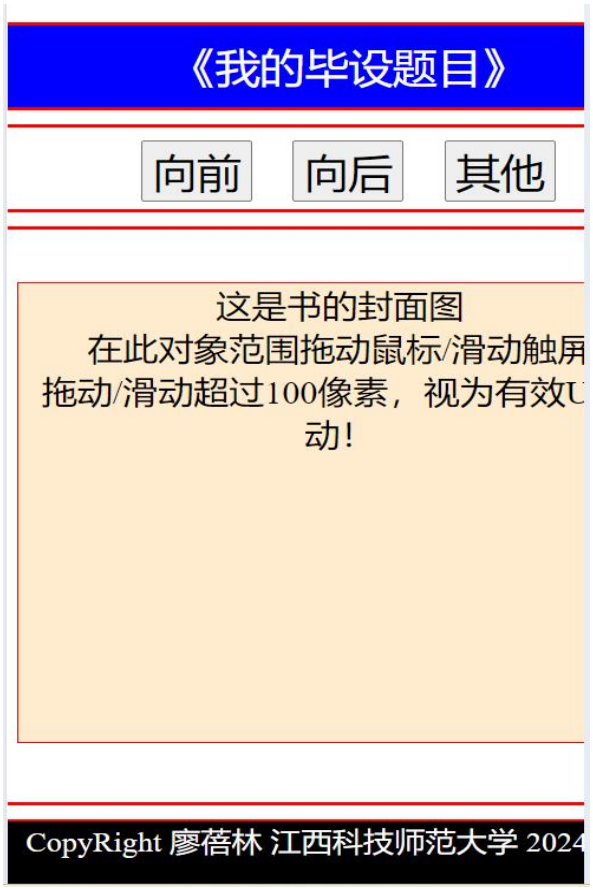


图 7-4 iPhone14 Pro max 端运行效果图

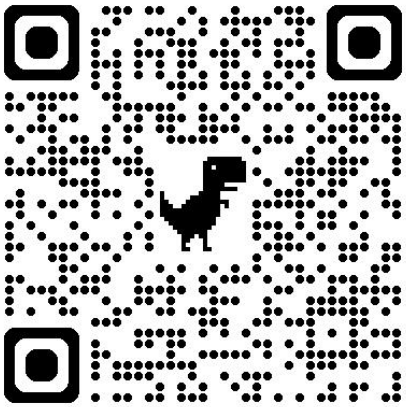


图 7-5 移动端二维码

7.4 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，作为项目的第五次迭代，在代码提交和版本管理环节，我们实现了对触屏和鼠标的通用交互操作的设计开发，让项目具有更高的精度和稳定性，尤其在处理复杂任务和精确操作时表现更为出色。向 github 提交了代码。进入 gitBash 命令行后，再进入文件夹，按次序输入下图 7-6 的命令：

```
86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git add 1.5.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git commit -m 项目第五版“对触屏和鼠标的通用交互操作的设计开发”
[master 743d118] 项目第五版“对触屏和鼠标的通用交互操作的设计开发”
 1 file changed, 227 insertions(+)
 create mode 100644 exp/1.5.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git log
commit 743d118697a9053895db80d7eac96022ade45447 (HEAD -> master)
Author: githubssh_key <2469766762@qq.com>
Date:   Thu Jun 13 11:10:05 2024 +0800

    项目第五版“对触屏和鼠标的通用交互操作的设计开发”
```

图 7-6

8. UI 的个性化键盘交互控制的设计开发

8.1 分析和设计

UI 的个性化键盘交互控制设计开发是一个专注于提升用户输入体验和效率的关键环节。个性化键盘旨在根据用户的习惯、偏好和使用场景，提供定制化的输入方式和交互控制。因为系统中只有一个键盘，所以我们把键盘事件的监听设置在 DOM 文档最大的可视对象——body 上。对象模型如图 8-1 用例图、8-2 dom 树状图所示。

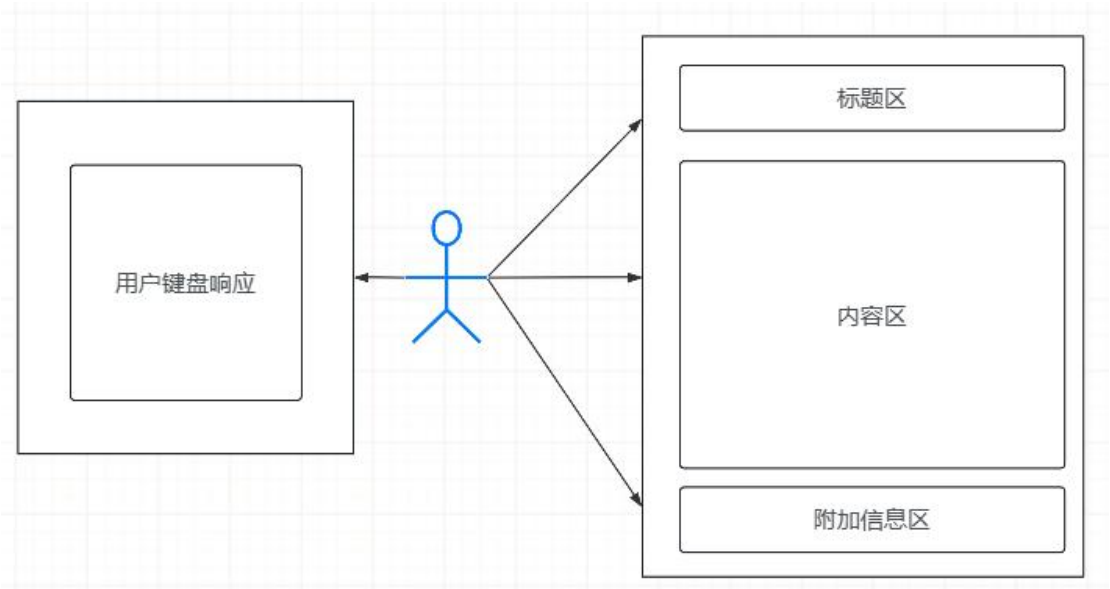


图 8-1 UI 的个性化键盘交互控制的设计开发用例图

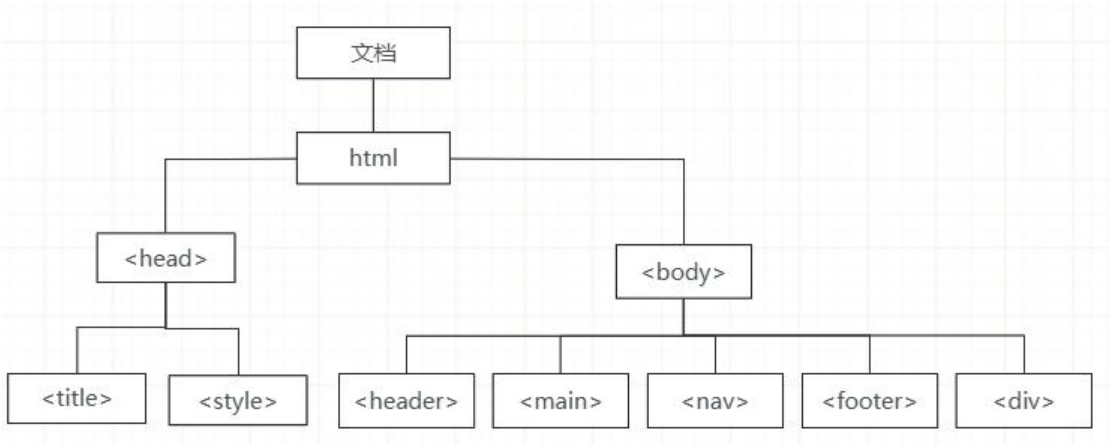


图 8-2 UI 的个性化键盘交互控制的设计开发的 DOM 树状图

8.2 项目的实现和内容

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)
});
```

8.3 项目的运行和测试

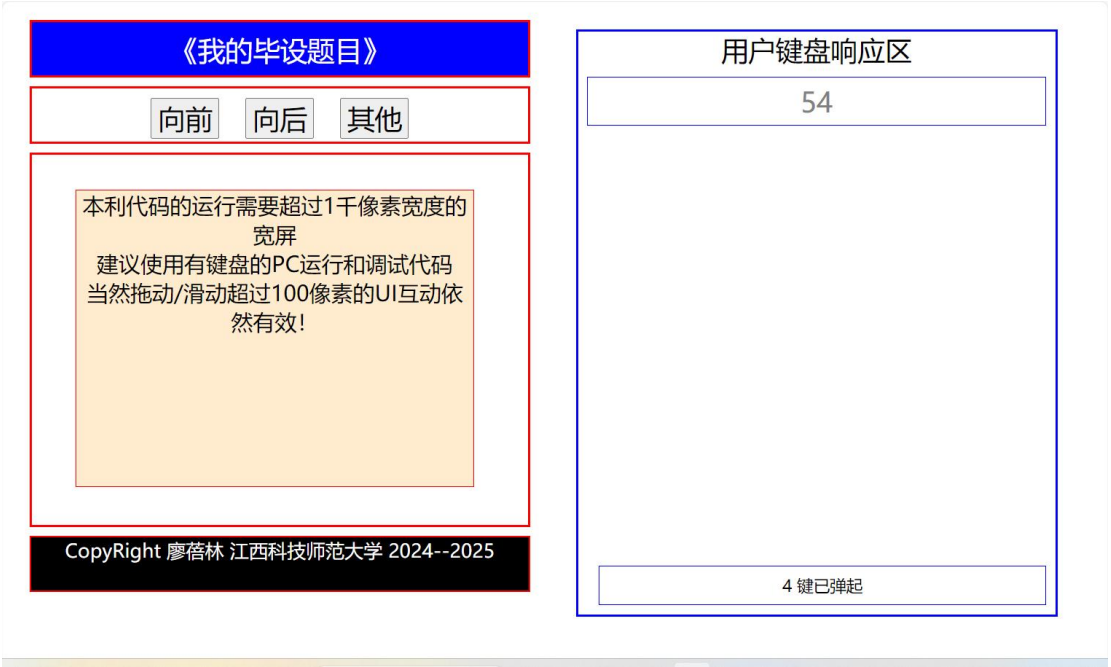


图 8-3 PC 端运行效果图

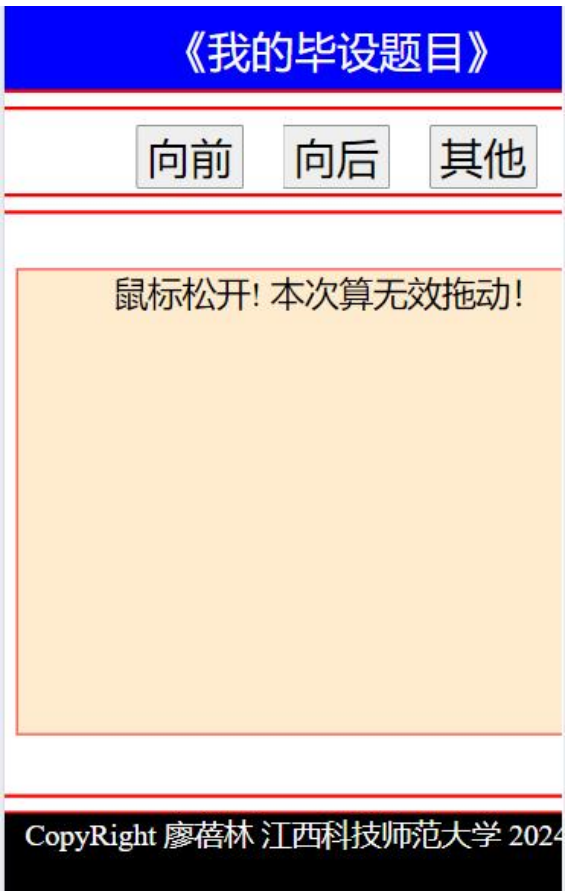


图 8-4 iPhone14 Pro max 端运行效果图

图 8-5 移动端二维码

8.4 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，作为项目的第五次迭代，在代码提交和版本管理环节，我们实现了 UI 的个性化键盘交互控制的设计开发，向 github 提交了代码。进入 gitBash 命令行后，再进入文件夹，按次序输入下图 8-6 的命令：

```
86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git add 1.6.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git commit -m 项目第六版“UI的个性化键盘交互控制的设计开发”
[master 248ae1e] 项目第六版“UI的个性化键盘交互控制的设计开发”
 1 file changed, 260 insertions(+)
 create mode 100644 exp/1.6.html

86198@DESKTOP-02FV6IA MINGW64 /d/abc/exp (master)
$ git log
commit 248ae1ef0209d5583c6704d48d937f8329e1e7f4 (HEAD -> master)
Author: githubssh_key <2469766762@qq.com>
Date:   Thu Jun 13 11:11:37 2024 +0800

    项目第六版“UI的个性化键盘交互控制的设计开发”
```

图 8-6

9. 用 gitBash 工具管理项目的代码仓库和 http 服务器

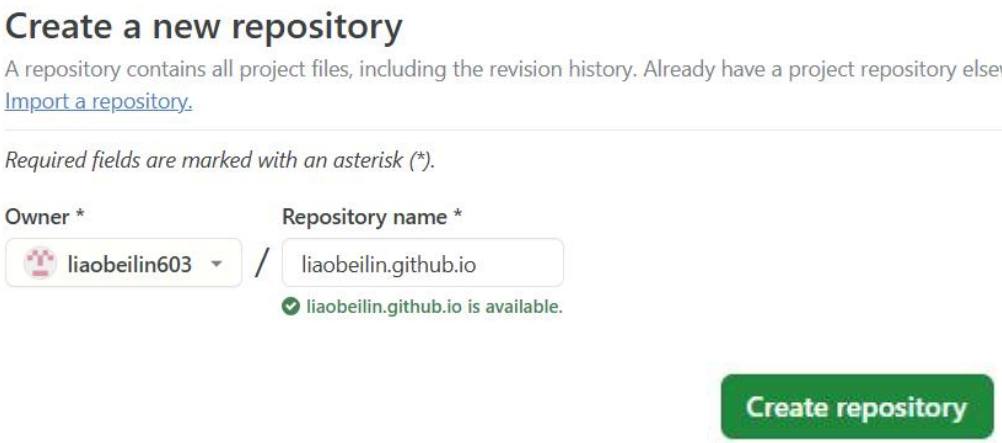
9.1 经典 Bash 工具介绍

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]。

9.2 通过 gitHub 平台实现本项目的全球域名

9.3 创建一个空的远程代码仓库



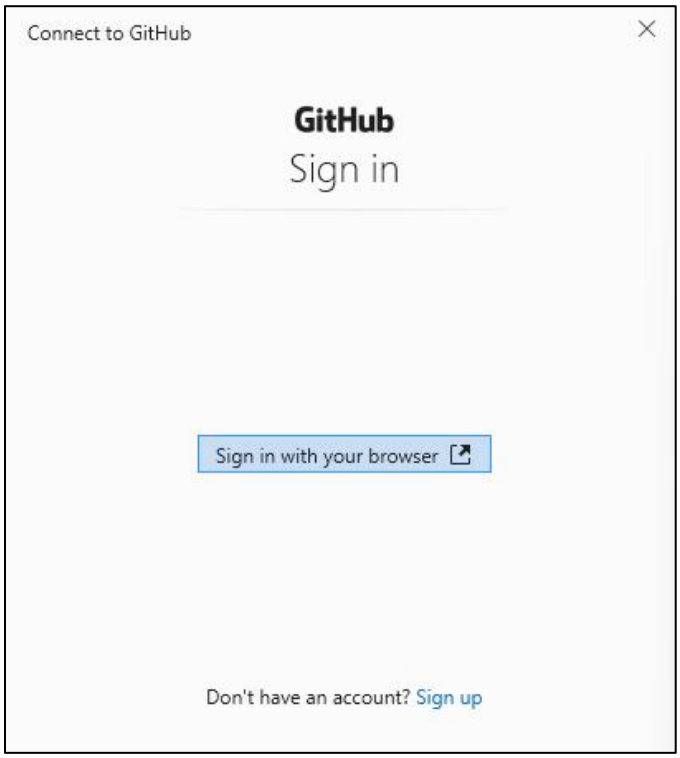
点击窗口右下角的绿色“Create repository”，则可创建一个空的远程代码仓库。

9.4 设置本地仓库和远程代码仓库的链接

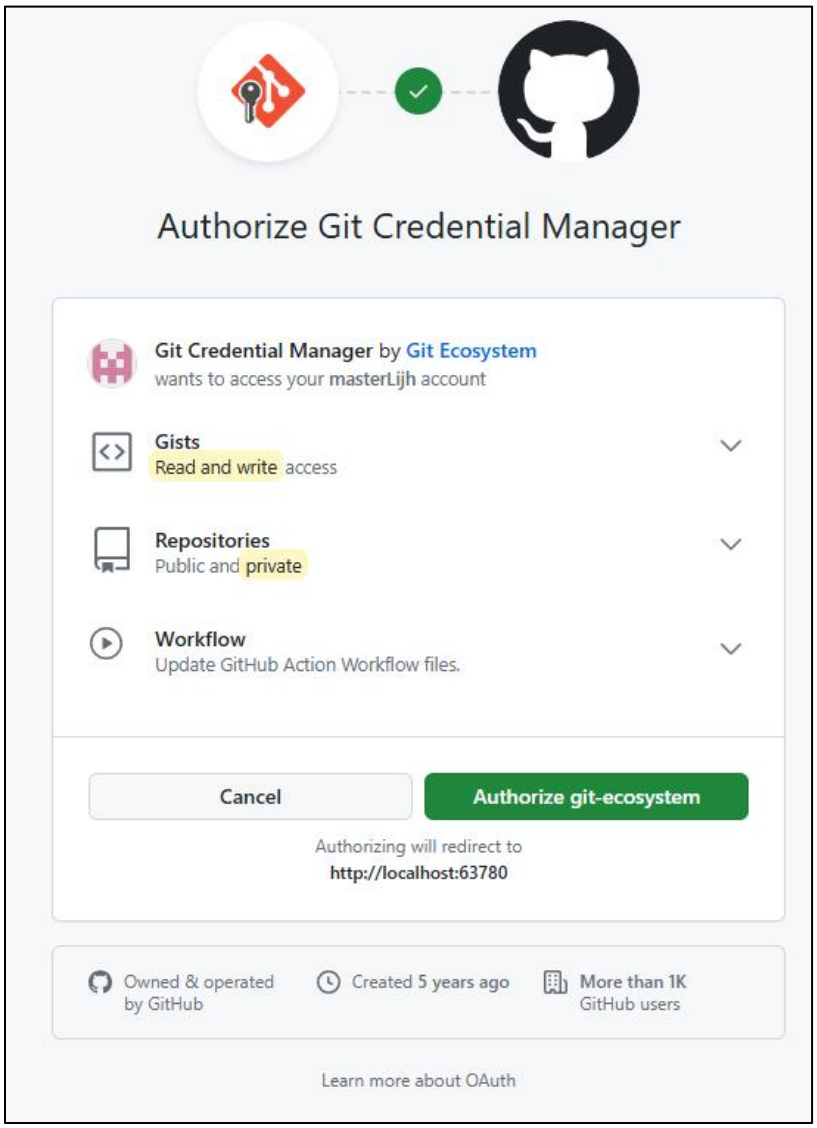
进入本地 abc 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接

```
$ 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/masterLijh/userName.github.io.git
$ git push -u origin main
```

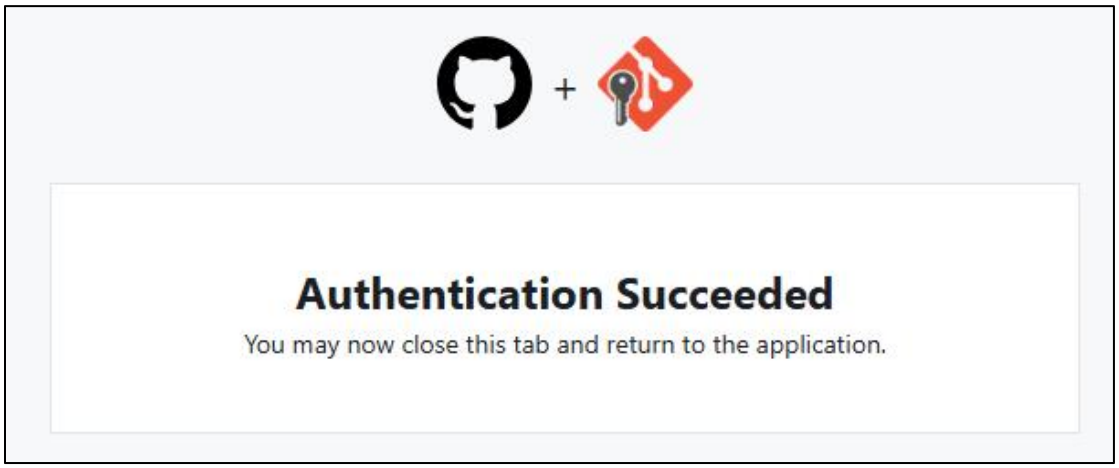
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



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