成 绩：

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 andProgramming

院 （系）：元宇宙产业学院

专 业：计算机科学与技术

学生姓名：夏豪

学 号：20213664

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 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 夏豪

摘要：近十年来，HTML5 为核心的 Web 标准的软件开发技术以其跨平台、开源的优势广泛运用于各个领域的应用软件开发中。为了研究和实践这种技术，本项目选择 HTML5 的 Web 客户端技术作为技术路线，设计并开发了一个个性化的用户界面（UI）应用程序。通过广泛查阅技术书籍、开发者论坛和文献，项目团队在 开发中综合应用了 HTML 进行内容建模、CSS 进行 UI 外观设计、JavaScript 编程实现 UI 交互功能。本项目采用了响应式设计，智能适应各种屏幕尺寸，展示了面向对象编程思想的运用，例如构建通用的指针模型，兼容鼠标和触屏控制，实现高质量代码。在工程管理方面，本项目采用了增量式开发模式，以逐步求精的方式展开了六次代码重构，包括分析（Analysis）、设计（Design）、实现（Implementation）和测试（Testing）四个经典开发阶段，顺利完成了项目的设计、开发和测试。为了代码的开源和分享，项目使用 Git 工具进行版本管理，在开发过程中进行了六次代码重构提交和两次测试修改提交，最后通过 GitBash 工具将项目代码仓库上传至 GitHub。借助 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，用户可以通过地址和二维码便捷地跨平台访问该程序。

关键字：HTML；增量式开发模式；GitBash；GitHub；

1. 前言

1.1 毕设任务分析

毕设任务要求学生综合运用本科阶段学习的计算机科学技术知识，尤其是程序设计和软件工程领域学习的方法、训练的代码能力，架构自己感兴趣的技术路线，结合自己探求的问题形成软件需求，然后有条理地系统落实分析问题、建立模型、软件设计、系统实施、测试调试的等传统软件工程的全部的流程。践行毕业设计，总结开发文档撰写论文，践行二者的有机结合，从而在实践和理论二个维度训练学生专业的计算思维和工程思维。

1.1.1 目的和意义

（1）目的

在即将完成学业之际，设计开发一个本专业的作品，有助于回顾总结本专业学习的知识系统，梳理课程体系最核心的东西，体现我们的真实能力。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。

（2）意义

因此，毕业设计的意义就是大学理论的学习在实践层面做一次综合演练和总结，期间也需要要配合学习当前最新的一些流行技术，在以形成自己对计算机软硬件体系的系统而专业的理解后，再总结撰写毕业论文，这既是毕业论文的意义。

1.2 研学计划

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。

当仿造导师的案例的技术基本实现后，则可以视为实践和理论基本打通，此时就可进入第二个阶段，开始真正做自己的毕设软件。

第二阶段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。

1.3 研究方法

1.3.1 文献法

文献法的理解可以用一句话表达，即：“利用前人的文字深入学习总结和研究本领域的知识和技术的方法”，对于我们 Web 应用方向的学子而言，在自己

代码能力成长的任何阶段，我们无法离开文献法，然而承载知识的媒体形式非常丰富，包括：书籍、在线文档、社区论坛、期刊、会议报告等等。

1.3.2 模型研究法

对于写代码的本科生，必须擅长使用的另一种研究方法就是“模型研究法”。这个研究方法非常具体，也很有意思，其给人的快乐甚至可以与打游戏对比。比如承载我们 Web 应用的台式机、笔记本、手机、平板，传递在线信息要用到的互联网、服务器，沟通硬件和我们的代码之间的操作系统、浏览器、代码编辑器、编译器，这些软硬件对象，对我们而言，都值得从写代码的角度去研究，我们笼统地称它们为对象，这些对象最终会在我们大脑中就会被理解为抽象的模型，我们再通过分析把这些模型程序化、数据化，最后写出代码来，这种行为本质上就是先在思维上“建模”，再用 OOP 语言表述出来。

在 OOP 分析和开发过程中，我在毕设中试图解决的问题，也被定义成为了各级各种模型。模型研究只是更为抽象，与具体的计算机语言无关，在毕设中我也尝试使用国际标准 UML（Unified Modeling Language）语言来建立抽象模型。我感觉采用 UML 模型研究法和面向对象的程序设计方法的目标是一致的，只是在不同层面分析表述问题而已。因此采用模型研究法，我的毕设再用例设计采用了类似的 UML 对问题建模，景观 UML 比较抽象，设计准确有一定难度，而使用 OOP 程序对画好的模型开展程序设计则更为具体直观，通过熟悉的 OOP 语言和代码运行环境运行和调试模型，我们甚至可以倒推出模型设计的问题和缺陷。我个人的观点是对于本科生而言，可能直接写代码建立模型研究模型，代码跑通后，再利用 UML 语言绘制模型，作为代码的文档资料则更合理。毕竟本科生在思维上还很稚嫩，无法比拟高级程序员乃至系统架构师那样的抽象能力和丰富的经验。

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，Sir Tim Berners-Lee 发明了万维网（参见原始提案）。他创造了“万维网”这一术语，并在 1990 年 10 月编写了第一个万维网服务器“httpd”以及第一个客户端程序（一个浏览器兼编辑器）“WorldWideWeb”。

他编写了“超文本标记语言”（HTML）的第一个版本，这是一种具有超文本链接能力的文档格式化语言，成为了网页发布的主要格式。随着网络技术的普及，他对 URI、HTTP 和 HTML 的最初规范进行了细化，并在更广泛的圈子中进行了讨论。

2.2.2 万维网联盟的成立

1994 年，在众多不断增加对网络投资的公司的强烈要求下，决定成立万维网联盟。Sir Tim Berners-Lee 开始领导万维网联盟团队的关键工作，旨在培育一个一致的架构，以适应网页构建标准、浏览器及设备体验网络所提供的一切功能的快速发展步伐。 在创立万维网联盟时，Sir Tim Berners-Lee 创建了一个同行社区。网络技术的发展速度已经非常快，因此集结一个单一组织来协调网络标准变得至关重要。Tim 接受了麻省理工学院（MIT）的提议，由其来托管 W3C，因为麻省理工学院在管理联盟方面有经验。从一开始，他就要求 W3C 必须具备全球影响力。

2.2.3 网络平台与网络编程

让我们从对网络（即万维网的简称）的简短描述开始。大多数人说“网络”而不是“万维网”，我们也将遵循这一习惯。网络是一个文档集合，这些文档被称为网页，它们在很大程度上被全世界的计算机用户共享。不同类型的网页执行不同的功能，但至少它们都能在计算机屏幕上显示内容。这里所说的“内容”包括文本、图片以及文本框和按钮等用户输入机制[2]。 网络编程是一个广阔的领域，不同类型的网络编程由不同的工具实现。所有这些工具都与核心语言 HTML 协同工作，因此几乎所有的网络编程书籍都会在一定程度上介绍 HTML。本教科书深入涵盖了 HTML5、CSS 和 JavaScript 这三个方面。这三项技术被认为是客户端网络编程的三大支柱。在客户端网络编程中，所有网页计算都在最终用户的计算机（客户端计算机）上完成[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：

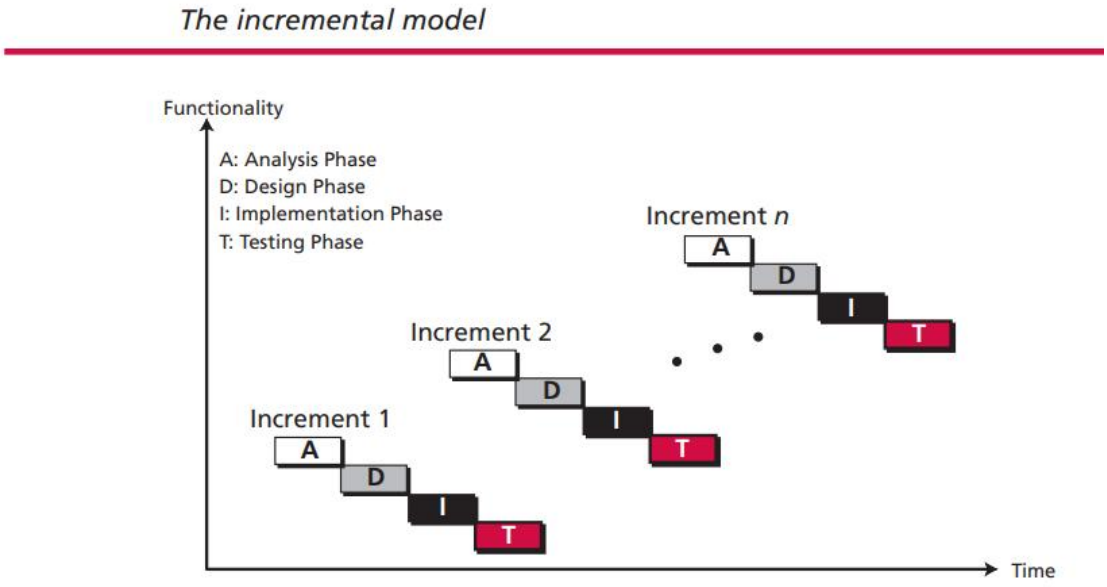

图 2-1 增量式迭代开发模型

2.2.1 增量式迭代开发模型

增量模型中，软件是通过一系列步骤开发的。开发者首先完成整个系统的一个简化版本。这个版本代表了整个系统，但不包含细节部分。图 2-1 展示的是增量模型的概念。

在第二个版本中，会添加更多的细节，同时也会留有一些未完成的部分，并再次对系统进行测试。如果出现任何问题，开发者就能知道问题出在新添加的功能上。在现有系统正常工作之前，他们不会继续添加新的功能。这一过程会持续进行，直到所有需要的功能都被添加完成[5]。

2.2.2 瀑布模型

瀑布模型是一种软件开发的经典模型，它呈现出严格的线性顺序流程，从需求分析开始，依次经过设计、编码、测试等阶段，最后进入维护阶段，各阶段划分清晰且具有明确的任务指向，其突出特点是阶段间的依赖性强且注重文档管理，虽然这种模型能确保流程的清晰和有序，但面对需求变更时灵活性不足，在一些需求稳定、对流程规范性要求较高的项目中能发挥较好作用。瀑布模型如图 2-2 所示。

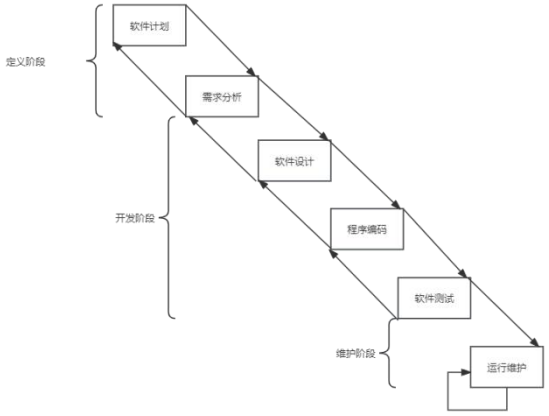

图 2-2 瀑布模型

2.2.3 增量式开发模型

增量模型是一种软件开发模型。它将整个软件项目分解为多个小的部分或增量，逐个进行开发和交付。在增量模型中，每个增量都包含了一组功能或特性，这些增量可以按顺序逐步添加到系统中。它具有以下特点：首先，它允许在项目进行过程中逐步展示成果，让用户可以较早地看到部分功能并提供反馈，有利于及时调整和优化；其次，它能够降低项目风险，因为每次只开发和交付一部分，便于控制和管理；再者，它在一定程度上提高了开发的灵活性，可以根据实际情况灵活安排增量的开发顺序和内容。比如，在开发一个大型电商平台时，可以先开发基本的商品展示和购买功能作为第一个增量，后续再逐步增加用户评价、促销活动等其他增量，通过这种方式逐步完善整个系统，既满足了用户的阶段性需求，又保证了开发的有序推进。模型图如图 2-3 所示。

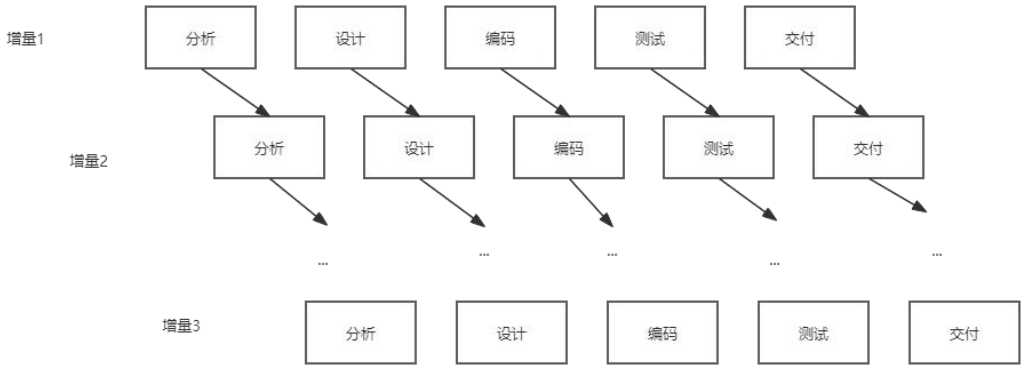

图 2-3 增量式开发模型

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。作为项目的初始版本，只搭建了一个大致框架，便于后续开发。项目用例图如图 3-1 所示：

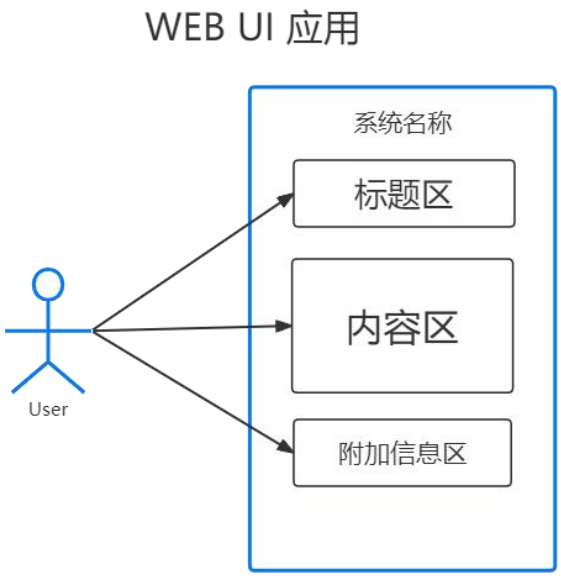

图 3-1 用例图

DOM 图如图 3-2 所示：

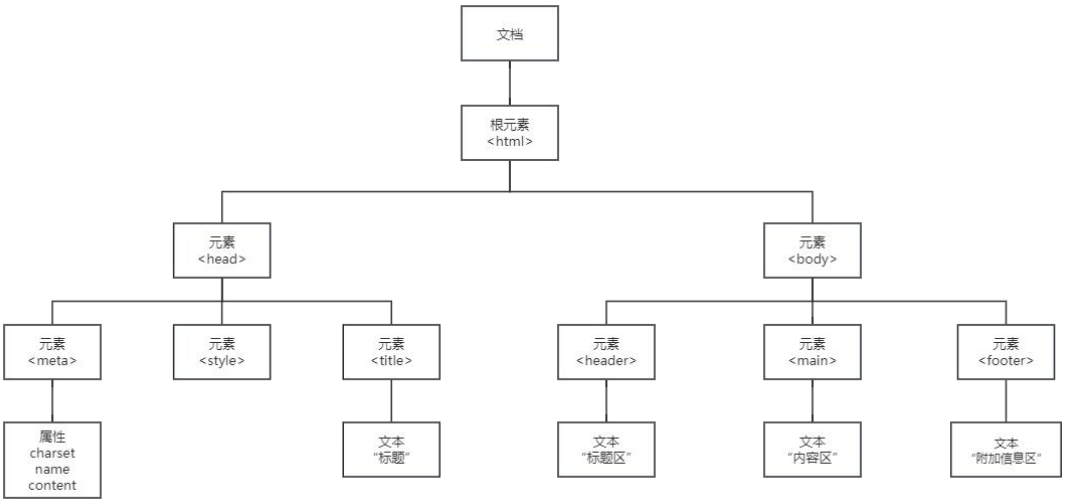

图 3-2DOM 图

3.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
   《计算思维和语言学习》
</header>
 <main>
   ‘读好书、练思维、勤编程’ @夏豪 计算思维系列课程
 </main>
<footer>
CopyRight from 夏豪 江西科技师范大学 2022--2025
</footer>
```

二、CSS 代码编写如下：

```
  body{
    font-size: 30px;
  }
  header{
    border: 2px solid blue;
    height: 200px;
  }

  main{
    border: 2px solid blue;
    height: 400px;
  }
```

```
footer{
  border: 2px solid blue;
  height: 100px;
}
```

3.3 项目的运行和测试

该项目在 PC Chrome 浏览器上运行效果如图 3-3 所示，该页面的二维码如图 3-4 所示。

图 3-3 PC 端运行效果图

图 3-4 网页二维码

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 江科师大夏豪
$ git config user.email 19880041639@163.com
$ touch 1.1.html
```

编写好 1.1.html 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add 1.1.html
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发，初步建立了项目主体框架
```

成功提交代码后，gitbash 的反馈如下图 3-5 所示：

```
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
[master (root-commit) 32de024] 项目第一版：“三段论”式的内容设计概
要开发
 2 files changed, 46 insertions(+)
 create mode 100644 index.html
 create mode 100644 myCss.css
```

图 3-5 代码提交反馈图

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图 3-6 所示：

```
Us@LAPTOP-NVCO6L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (master)
$ git log
commit 2bb3bd897182fe45d9ef6b3f77ad9e06039a304e (HEAD -> master)
Author: 江科师大夏豪 <19880041639@163.com>
Date:   Thu Jun 13 14:46:11 2024 +0800

    项目第一版：“三段论”式的内容设计概要开发，初步建立项目主体框架
```

图 3-5 代码提交日志

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种类繁多，其尺寸和分辨率取决于成本。设计师没有为每种类型的显示屏创建网页的不同版本，而是选择让网页提供总体布局指南，并允许浏览器决定如何在给定的计算机上显示页面。因此，网页不会提供许多详细信息。例如，网页作者可以指定一组句子构成一个段落，但无法指定如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带来了一个有趣的结果：通过两个不同的浏览器或在硬件不相同的两台电脑上查看时，同一个网页可能呈现出不同的样子。如果一个屏幕比另一个宽，那么文本行的长度或可显示图像的大小就会不同。重点在于：网页提供了关于期望展示效果的总体指导原则；而浏览器在显示页面时选择具体细节。结果就是，同一个网页在两台不同的电脑上或通过不同的浏览器显示时，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。

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。从而实现在不同的尺寸的显示设备下，都能较好的展示界面。基于这样的要求下，本项目在第一版宽屏的基础上，开发了一套适用于窄屏的框架。本项目移动端展示使用 iPhone SE。项目用例图如图 4-1 所示。

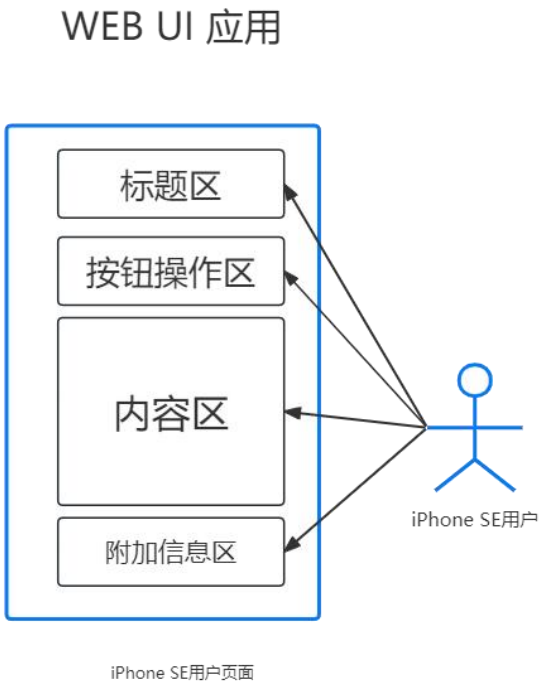

图 4-1 用例图

DOM 图如图 4-2 所示：

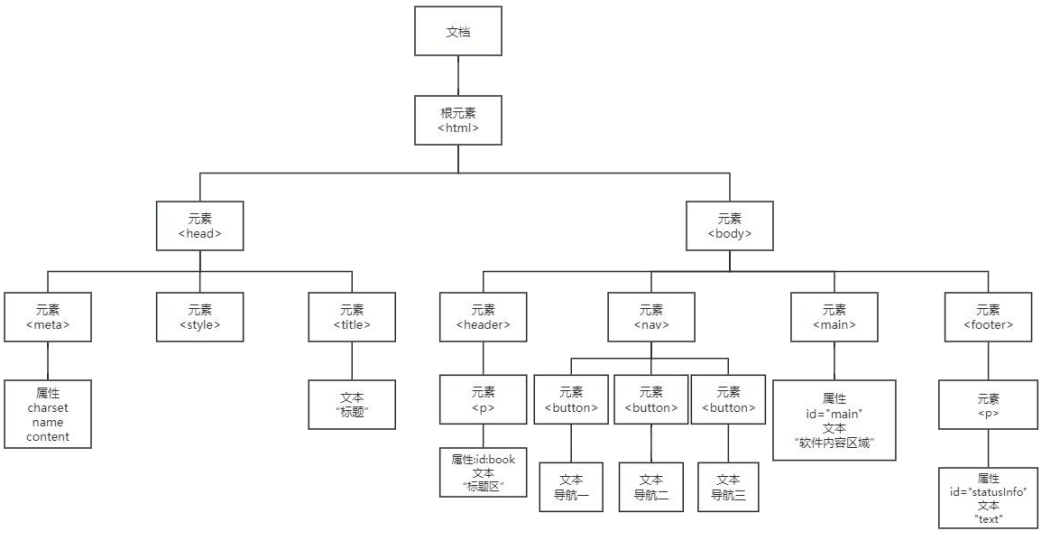

图 4-2 DOM 图

4.2 项目的实现和编程

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: 65%;
     font-size: 1.2em;

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

该项目在 iPhone SE 运行效果如图 4-3 所示。

图 4-3 iPhone SE 运行效果图

图 4-4 网页二维码

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。

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ cd webUI
$ touch 1.2.html
```

编写好 1.2.html 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：初步设计并实现了移动互联网时代 UI 开发——窄屏终端的响应式设计。
```

成功提交代码后，gitbash 的反馈如下图 4-5 所示：

```
Us@LAPTOP-NVCO6L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (master)
$ git commit -m 项目第二版：初步设计并实现了移动互联时代UI开发--窄屏终端的响应式设计
[master 23ca9cc] 项目第二版：初步设计并实现了移动互联时代UI开发--窄屏终端的响应式设计
 1 file changed, 78 insertions(+)
 create mode 100644 exp/1.2.html
```

图 4-5 代码提交反馈图

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图 4-6 所示：

```
Us@LAPTOP-NVCO6L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (master)
$ git log
commit 23ca9cc86e088c319e26f9700a9a3bad80b9b40d (HEAD -> master)
Author: 江科师大夏豪 <19880041639@163.com>
Date:   Thu Jun 13 17:14:19 2024 +0800

    项目第二版：初步设计并实现了移动互联时代UI开发--窄屏终端的响应式设计
```

图 4-5 代码提交日志

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

在现代 Web 开发中，响应式设计技术是确保网页在各种设备上提供一致且优质用户体验的关键。通过响应式设计，我们可以创建一个页面，它能够自动适应不同的屏幕尺寸，无论是在移动设备上的窄屏还是在 PC 上的宽屏。在前两次版本迭代中，我们分别设计了能够适用于宽屏和窄屏的 WebUI。然而，现实情

况下，移动端和 PC 端上显示的页面通常是同一个页面。为了实现这一目标，我们需要综合考虑不同设备的特点，并采用响应式设计技术来开发可适配窄屏和宽屏的 UI。通过识别设备的高度和宽度，分别设置不同的页面。当页面是宽屏时，为其添加一个键盘响应区，若设备为窄屏，则隐藏键盘响应区，以适配手机窄屏页面。在宽屏上，书本封面区域会监听鼠标位置，并能响应鼠标点击事件，显示鼠标操作时间以及坐标。并且右侧相较于窄屏，会额外有一个键盘响应区。会响应用户键盘事件，显示用户按下键的名称以及字符编码。项目用例图如图 5-1 所示。

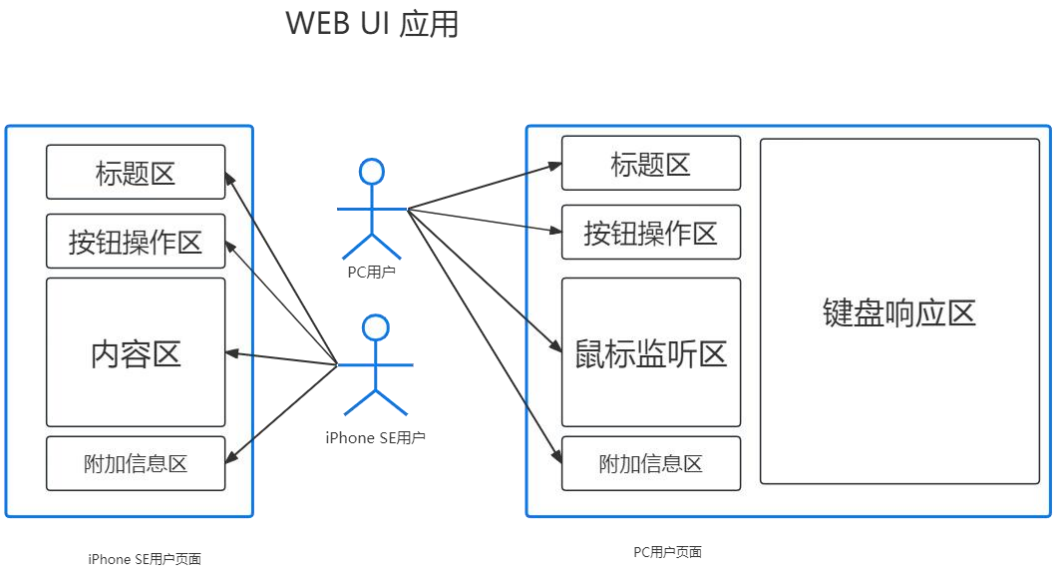

图 5-1 用例图

以及 DOM 结构如图 5-2 所示

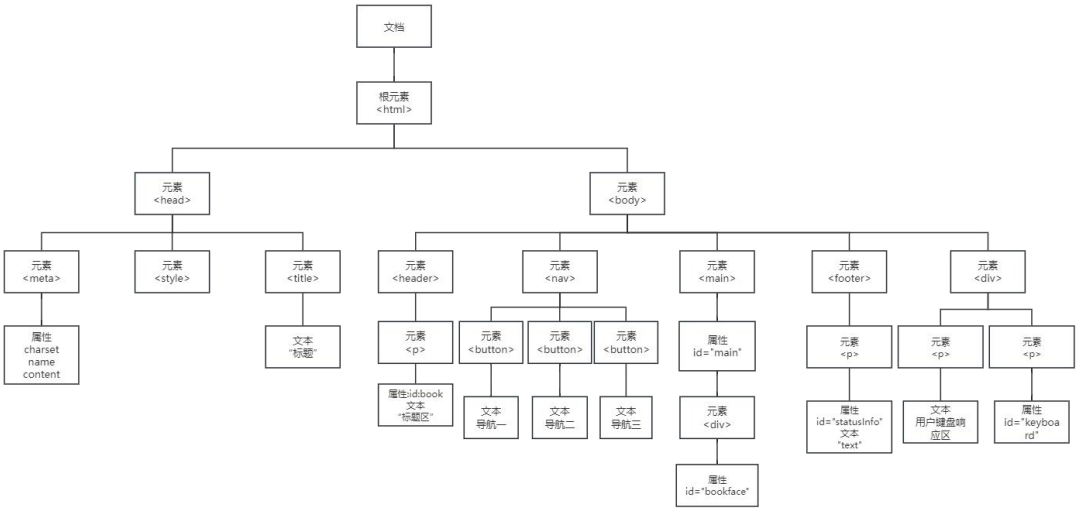

图 5-2 DOM 图

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>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>
  </div>
 </main>

<footer>

CopyRight 夏豪 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：

```
<style>
*{
   margin: 10px;
   text-align: center;
}
header{
   border: 3px solid green;
   height: 10%;
   font-size: 1em;
```

```
}
nav{
  border: 3px solid green;
  height: 10%;
}
main{
  border: 3px solid green;
  height: 70%;
  font-size: 0.8em;
  position: relative;
}

#box{
  position: absolute;
  right: 0;
  width: 100px;
}

footer{
  border: 3px solid green;
  height:10%;
  font-size: 0.7em;
}
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% ;
}
```

```
  </style>
```

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 - 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 $

  </script>
```

5.3 项目的运行和测试

该项目在 PC 上的运行效果图如图 5-3 所示。

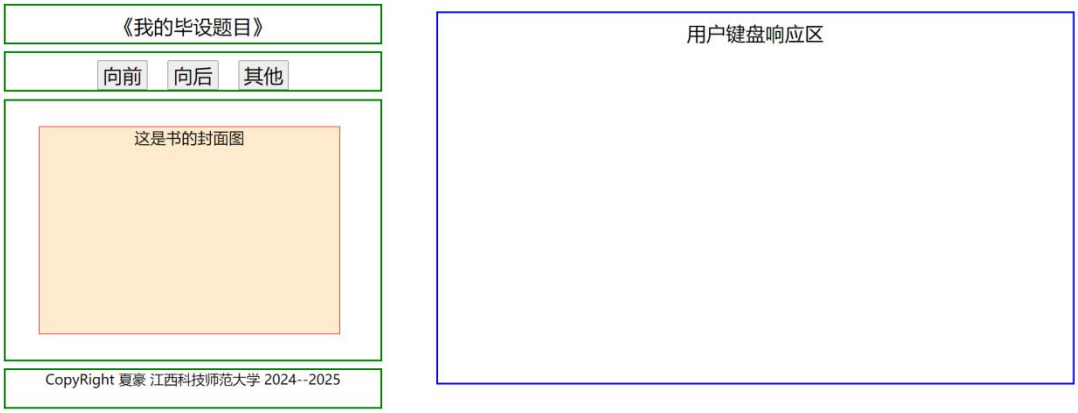

图 5-3 PC 运行效果图

该项目在窄屏终端 iPhone SE 上的运行效果图如图 5-4 所示。

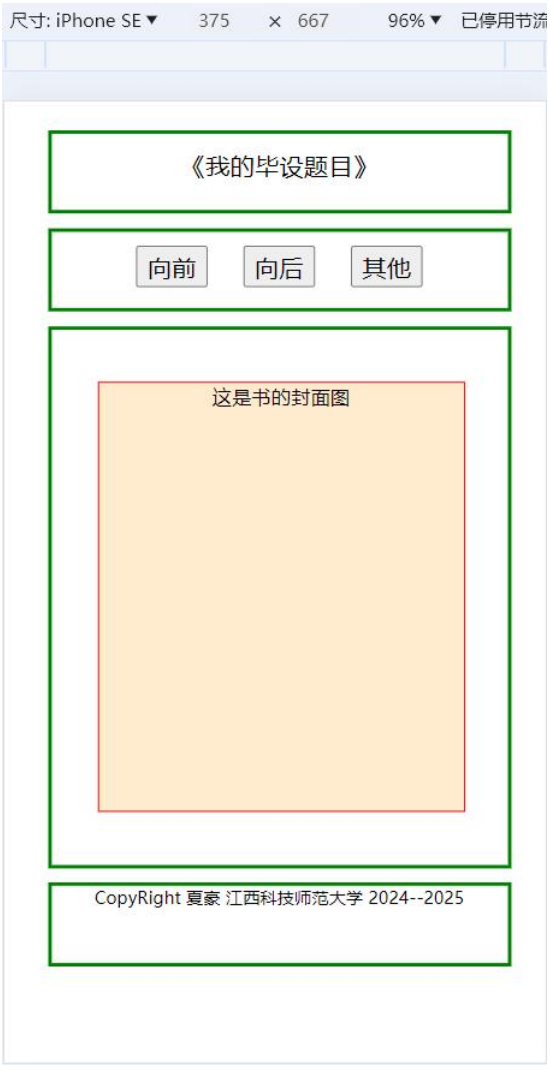

图 5-4 iPhone SE 运行效果图

该项目的网页二维码如图 5-5 所示。

图 5-5 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。

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ cd webUI
$ touch 1.3.html
```

编写好 1.3.html 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：应用响应式设计技术开发可适配窄屏和宽屏的 UI，实现了宽窄屏的响应，对于宽屏，实现了监听鼠标事件以及对键盘进行响应
```

成功提交代码后，gitbash 的反馈如下图 5-6 所示：

```
Us@LAPTOP-NVCO6L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (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，实现了宽窄屏的响应，对于宽屏，实现了监听鼠标事件以及对键盘进行响应
[master ce301cd] 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，实现了宽窄屏的响应，对于宽屏，实现了监听鼠标事件以及对键盘进行响应
 1 file changed, 169 insertions(+)
 create mode 100644 exp/1.3.html
```

图 5-6 代码提交反馈图

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图 5-7 所示：

```
Us@LAPTOP-NVCO6L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (master)
$ git log
commit ce301cd3311d6f3c55396edbfba75393e680a763 (HEAD -> master)
Author: 江科师大夏豪 <19880041639@163.com>
Date:   Thu Jun 13 19:31:42 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，实现了宽窄屏的响应
，对于宽屏，实现了监听鼠标事件以及对键盘进行响应
```

图 5-7 代码提交日志

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

在本次迭代开发中，将对鼠标建立对象模型，对鼠标设计 UI 控制，为鼠标监听区域添加鼠标拖动效果。并对鼠标事件进行了更加细致的划分，对鼠标按下和鼠标松开都有监听以及文字标识。为了更好的响应以及监听鼠标多样化事件。当鼠标在监听区域按下并拖动时，网页会对拖动的距离和方向进行响应。当拖动距离大于 100 时，则为有效拖动，反之则为无效拖动。

在本次迭代中，还会初步对网页的样式进行调整，为鼠标监听区域添加书本封面，并且能随着鼠标拖动而移动。也初步为按钮添加功能，能够切换书本封面。项目用例图如图 6-1 所示。

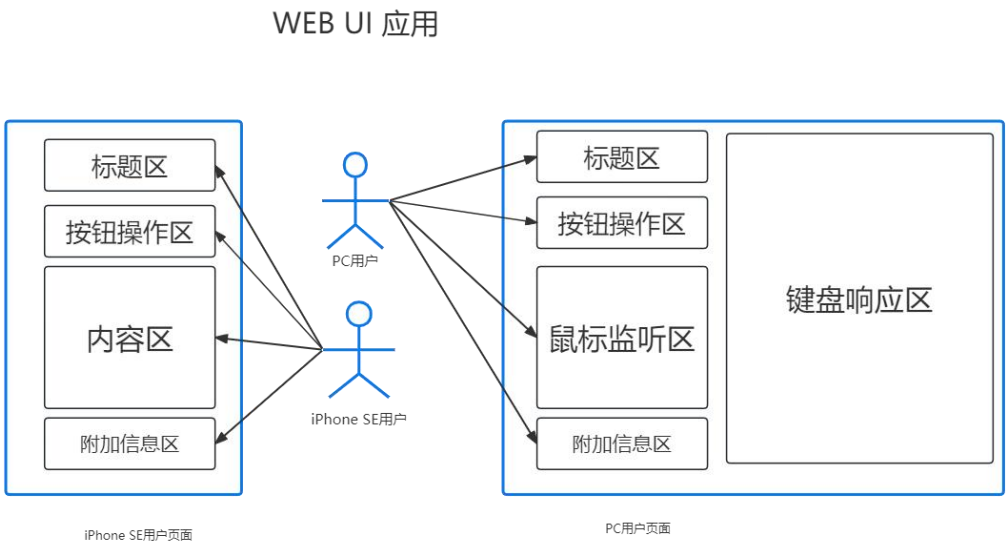

图 6-1 用例图

以及 DOM 结构如图 6-2 所示

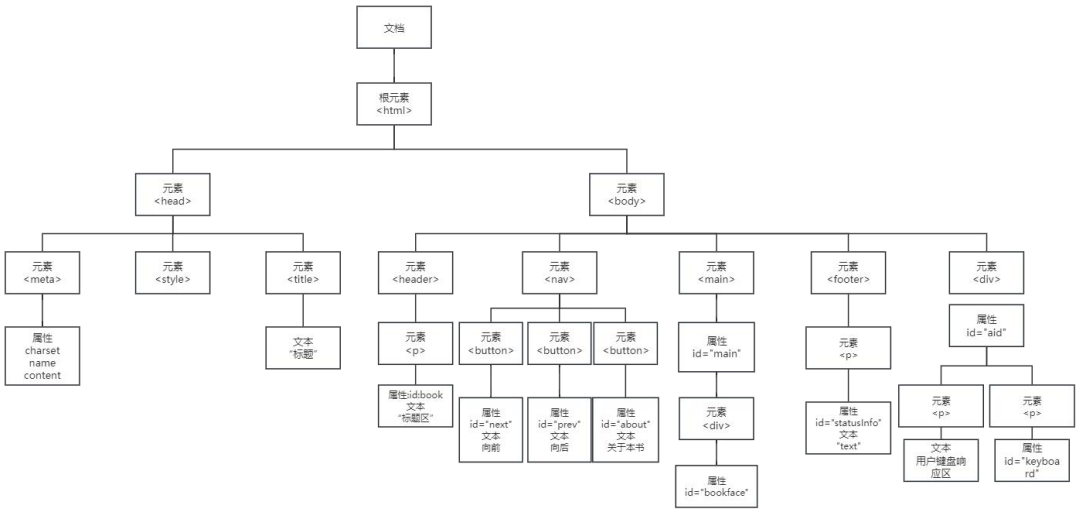

图 6-2 DOM 图

6.2 项目的实现和编程

一、HTML 代码

```
<body >
  <header>
    <p id="book">
      《WebAppUI 设计开发》
    </p>
   </header>
  <nav>
   <button id="next">向前</button>
   <button id="prev">向后</button>
   <button id="about">关于本书</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 夏豪 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

```
<style>
 *{
   margin: 10px;
   text-align: center;
 }
 header{
   border: 3px solid green;
   height: 10%;
   font-size: 1em;

 }
 nav{
   border: 3px solid green;
   height: 10%;
 }
 main{
   border: 3px solid green;
   height: 70%;
   font-size: 0.8em;
   position: relative;
 }

 #box{
   position: absolute;
   right: 0;
   width: 100px;
 }

 footer{
   border: 3px solid green;
   height:10%;
   font-size: 0.7em;
 }
 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% ;
   background-image: url("../lesson/CS.jpg");
   background-size: cover;
 }

</style>
```

三、JavaScript 代码

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

 UI.appHeight = window.innerHeight;

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

```
  $("aid").style.height= UI.appHeight - baseFont*3 +'px';

//尝试对鼠标设计 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' ;
  }

});
$("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 $

      var myDiv = document.getElementById('bookface');
   var images = [
     "../lesson/CS.jpg",
     "../lesson/CSS.jpg",
     "../lesson/linuxCMD.jpg",
     "../lesson/Http.jpg"
   ];
   var pdf = [
     "../pdf/CS.pdf",
     "../pdf/CSS.pdf",
```

```
    "../pdf/linuxCMD.pdf",
    "../pdf/Http.pdf"
  ];
  //当前位置
  var currentImageIndex = 0;

  // 定义下一个图片的函数
  function showNextImage() {
    currentImageIndex++;
    if (currentImageIndex == images.length) currentImageIndex = 0;
    myDiv.style.backgroundImage = 'url(' + images[currentImageIndex]
+ ')';
  }

  // 定义上一个图片的函数
  function showPrevImage() {
    currentImageIndex--;
    if (currentImageIndex == -1) currentImageIndex = images.length
- 1;
    myDiv.style.backgroundImage = 'url(' + images[currentImageIndex]
+ ')';
  }

  // 将函数绑定到点击事件
  document.getElementById("next").onclick = showNextImage;
  document.getElementById("prev").onclick = showPrevImage;
  document.getElementById("about").onclick = function () {
    var pdfUrl = 'path/to/your/document.pdf';
    // 使用 window.open 打开 PDF 文件
    window.open(pdf[currentImageIndex], '_blank');
  }
</script>
```

6.3 项目的运行和测试

该项目在 PC 上的运行效果图如图 6-3 所示。

图 6-3 PC 运行效果图

该项目在窄屏终端 iPhone SE 上的运行效果图如图 6-4 所示。

图 6-4 iPhone SE 运行效果图

该项目的网页二维码如图 6-5 所示。

图 6-5 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。

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ cd webUI
$ touch 1.4.html
```

编写好 1.4.html 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：个性化 UI 设计中对鼠标交互的设计开发,完善了鼠标事件的监听，初步为页面添加了样式。
```

成功提交代码后，gitbash 的反馈如下图 6-6 所示：

```
Us@LAPTOP-NVCO6L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (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,完善了鼠标事件的监听，初步为页面添加了样式。
[master 6182de7] 项目第四版：个性化UI设计中对鼠标交互的设计开发,完善了鼠标事件的监听，初步为页面添加了样式。
 1 file changed, 237 insertions(+)
 create mode 100644 exp/1.4.html
```

图 6-6 代码提交反馈图

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图 6-7 所示：

图 6-7 代码提交日志

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

在本次更新迭代中，通过一套统一的逻辑，针对触屏和鼠标事件建立了对象模型，确保在不同设备上都能良好运行。在代码中设置了界面的宽度和高度，根据设备的屏幕尺寸进行响应式调整。通过计算基础字体大小(baseFont)并应用到整个页面，确保页面内容的大小在不同设备上都保持一致。代码定义了一个名为 Pointer 的对象，用于存储指针事件的状态和位置。Pointer 对象包含三个属性：isDown 表示指针是否按下，x 和 deltaX 分别表示指针的初始位置和移动距离。代码中包含三个主要的事件处理函数 handleBegin、handleEnd 和 handleMoving，这些函数分别处理指针按下、松开和移动事件。在 handleBegin 函数中，通过检测 ev.touches 来区分触屏和鼠标事件，并记录指针的初始位置。在 handleEnd 函数中，计算指针移动的距离，并根据移动距离判断是否为有效的滑动或拖动。handleMoving 函数则实时更新指针的移动距离，并动态调整页面元素的位置。并在每次有效拖动或者滑动操作后，根据左移还是右移决定切换上一本书还是下一本书。

通过给特定的 DOM 元素绑定这些事件处理函数，代码实现了对鼠标和触屏事件的统一处理。此外，代码还通过添加键盘事件处理函数，实现对键盘按键的实时响应。

这套逻辑确保在触屏设备和传统电脑上，都能通过相同的代码实现一致的用户交互体验。项目用例图如图 7-1 所示。

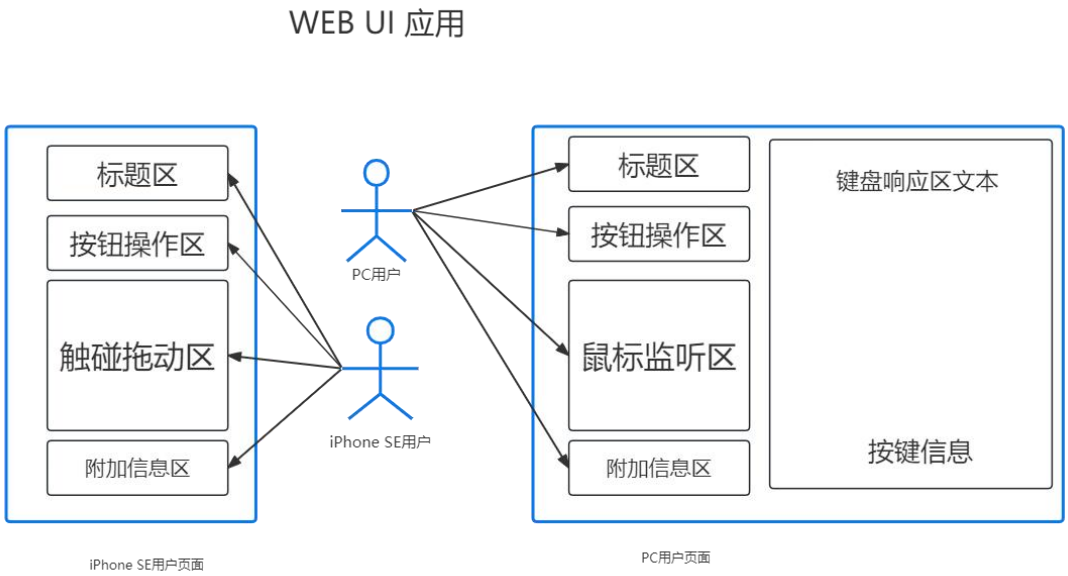

图 7-1 用例图

以及 DOM 结构如图 7-2 所示

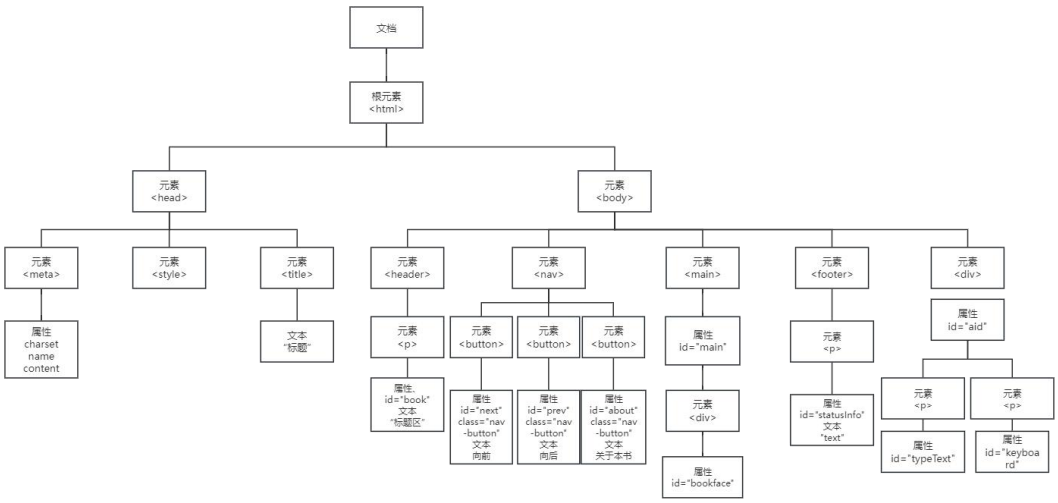

图 7-2 DOM 图

7.2 项目的实现和编程

一、HTML 代码

```
<header>
  <p id="book">
    《我的毕设题目》
  </p>
</header>
```

```
<nav>
  <button id="next" class="nav-button">向前</button>
  <button id="prev" class="nav-button">向后</button>
  <button id="about" class="nav-button">关于本书</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 夏豪 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>
  <hr>
  <p id="keyboard"></p>
</div>
```

二、CSS 代码

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

  header {
    border: 3px solid green;
    height: 10%;
    font-size: 1em;

  }

  nav {
    border: 3px solid green;
    height: 10%;
  }
```

```
main {
  border: 3px solid green;
  height: 70%;
  font-size: 0.8em;
  position: relative;
}

#box {
  position: absolute;
  right: 0;
  width: 100px;
}

footer {
  border: 3px solid green;
  height: 10%;
  font-size: 0.7em;
}

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%;
  background-image: url('../lesson/Http.jpg');
```

```
  background-repeat: no-repeat;
  background-size: cover;

}
.nav-button {
  margin: 6px 0;
  background-color: rgba(111, 112, 14, 0.2);
  border-radius: 6px;
  border: none;
  cursor: pointer;
  position: relative;
  overflow: hidden;
  transition: 0.7s;
}

.nav-button:hover {
  background-color: rgb(20, 130, 178);
}

.nav-button::before,
.nav-button::after {
  content: "";
  height: 90px;
  width: 100px;
  display: block;
  background-color: mediumaquamarine;
  opacity: 0.5;
  filter: blur(30px);
  position: absolute;
  overflow: hidden;
  left: 0;
  top: 0;
  transform: skewX(-20deg);
  transform: translateX(-100px);
}

.nav-button::after {
  filter: blur(6px);
  width: 40px;
  left: 60px;
  opacity: 0;
  background-color: rgba(20, 184, 167, 0.307);
}
```

```
   .nav-button:hover::before {
     transform: translateX(320px);
     transition: 1s;
     opacity: 1;

   }

   .nav-button:hover::after {
     transform: translateX(320px);
     transition: 1s;
     opacity: 1;
   }
 </style>
```

三、JavaScript 代码

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

   UI.appHeight = window.innerHeight;

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

   //尝试对鼠标和触屏设计一套代码实现 UI 控制
   var Pointer = {};
   Pointer.isDown = false;
   Pointer.x = 0;
```

```
   Pointer.deltaX = 0;
   { //Code Block Begin
     let handleBegin = function (ev) {
       Pointer.isDown = true;

       if (ev.touches) {
         console.log("touches1" + ev.touches);
         Pointer.x = ev.touches[0].pageX;
         Pointer.y = ev.touches[0].pageY;
         console.log("Touch begin：" + "(" + Pointer.x + "," + Pointer.y
+ ")");
         $("bookface").textContent = "触屏事件开始，坐标：" + "(" +
Pointer.x + "," + Pointer.y + ")";
       } else {
         Pointer.x = ev.pageX;
         Pointer.y = ev.pageY;
         console.log("PointerDown at x: " + "(" + Pointer.x + "," +
Pointer.y + ")");
         $("bookface").textContent = "鼠标按下，坐标：" + "(" + Pointer.x
+ "," + Pointer.y + ")";
       }
     };
     let handleEnd = function (ev) {
       Pointer.isDown = false;
       ev.preventDefault()
       //console.log(ev.touches)
       let direction = Pointer.deltaX > 0 ? 'right' : 'left';
       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! ";
           $("bookface").style.left = '7%';
           if (direction === 'right' && Pointer.isDown) {
             showNextImage();
           }else{
             showPrevImage();
           }
         } 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! ";
          $("bookface").style.left = '7%';
          if (direction === 'right' && Pointer.isDown) {
            showNextImage();
          }else{
            showPrevImage();
          }
        } 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);
```

```
    //添加"keydown"和"keyup"这 2 个键盘底层事件处理后，keypress 这个高层键盘事件响应被系统忽略
     $("body").addEventListener("keypress", function(ev){
       $("typeText").textContent += ev.key ;
     });

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

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

```
'(', ')', '-', '_', '+', '=', ',', '.', '<', '>', '?', '/', ' '];
        if ((k >= 'a' && k <= 'z') || (k >= 'A' && k <= 'Z') || (k >=
'0' && k <= '9')) {
          console.log("letters");
          return true;
        }
        for (let p of puncs) {
          if (p === k) {
            console.log("puncs");
            return true;
          }
        }
        return false;
        //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 $
  var myDiv = document.getElementById('bookface');
  var images = [
    "../lesson/CS.jpg",
    "../lesson/CSS.jpg",
    "../lesson/linuxCMD.jpg",
    "../lesson/Http.jpg"
  ];
  var pdf = [
    "../pdf/CS.pdf",
```

```
    "../pdf/CSS.pdf",
    "../pdf/linuxCMD.pdf",
    "../pdf/Http.pdf"
  ];
  //当前位置
  var currentImageIndex = 0;

  // 定义下一个图片的函数
  function showNextImage() {
    currentImageIndex++;
    if (currentImageIndex == images.length) currentImageIndex = 0;
    myDiv.style.backgroundImage = 'url(' + images[currentImageIndex]
+ ')';
  }

  // 定义上一个图片的函数
  function showPrevImage() {
    currentImageIndex--;
    if (currentImageIndex == -1) currentImageIndex = images.length
- 1;
    myDiv.style.backgroundImage = 'url(' + images[currentImageIndex]
+ ')';
  }

  // 将函数绑定到点击事件
  document.getElementById("next").onclick = showNextImage;
  document.getElementById("prev").onclick = showPrevImage;
  document.getElementById("about").onclick = function () {
    var pdfUrl = 'path/to/your/document.pdf';
    // 使用 window.open 打开 PDF 文件
    window.open(pdf[currentImageIndex], '_blank');
  }
 </script>
```

7.3 项目的运行和测试

该项目在 PC 上的运行效果图如图 7-3 所示。

图 7-3 PC 运行效果图

该项目在窄屏终端 iPhone SE 上的运行效果图如图 7-4 所示。

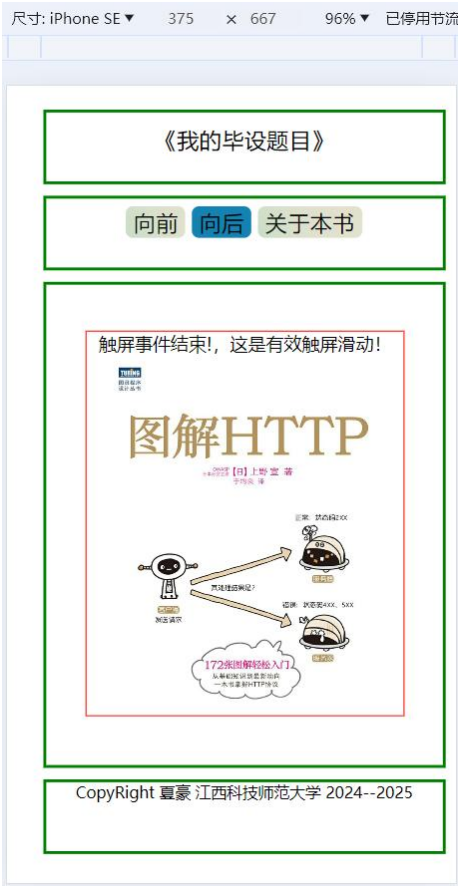

图 7-4 iPhone SE 运行效果图

该项目的网页二维码如图 7-5 所示。

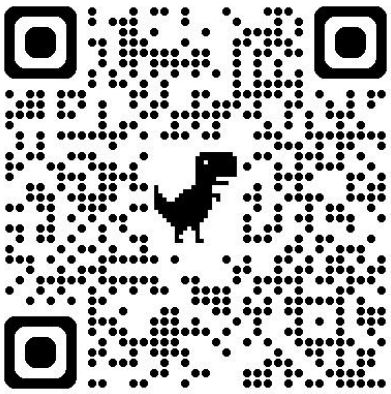

图 7-5 二维码

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。

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ cd webUI
$ touch 1.5.html
```

编写好 1.5.html 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：对触屏和鼠标的通用交互操作的设计开发，本次开发对触 屏和鼠标拖动都进行了实现，并再次优化样式和键盘响应区，并且有效滑动还能够切换封 面，就和小说翻页一样。
```

成功提交代码后，gitbash 的反馈如下图 7-6 所示：

```
Us@LAPTOP-NVCO6L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (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，本次开发对触屏和鼠标拖动都进行了实现，并再次优化样式和键盘响应区，并且有效滑动还能够切换封面，就和小说翻页一样。
[master af244b6] 项目第五版：对触屏和鼠标的通用交互操作的设计开发，本次开发对触屏和鼠标拖动都进行了实现，并再次优化样式和键盘响应区，并且有效滑动还能够切换封面，就和小说翻页一样。
 1 file changed, 381 insertions(+)
 create mode 100644 exp/1.5.html
```

图 7-6 代码提交反馈图

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图 7-7 所示：

```
commit af244b643917c18c386f3193a36cc81149e2accb (HEAD -> master)
Author: 江科师大夏豪 <19880041639@163.com>
Date:   Thu Jun 13 21:13:15 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，本次开发对触屏和鼠标拖动都进行了实现，并再次优化样式和键盘响应区，并且有效滑动还能够切换封面，就和小说翻页一样。
```

图 7-7 代码提交日志

8. UI 的个性化键盘交互控制的设计开发

8.1 分析和设计

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)
});
```

在本次迭代中，将会对键盘响应进行优化，用户可以输入字母、数字、符号等，并且 enter 键可以换行，backspace 键可以删除。不管是手机用户还是 PC 用户，都可以通过滑动来切换书的封面。用例图如图 8-1 所示。

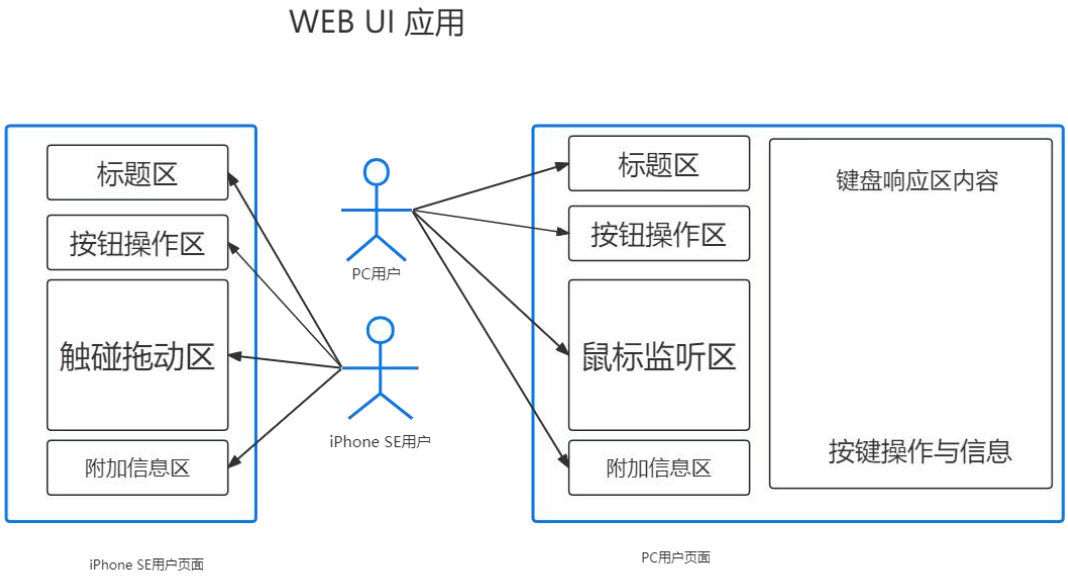

图 8-1 用例图

以及 DOM 结构如图 7-2 所示

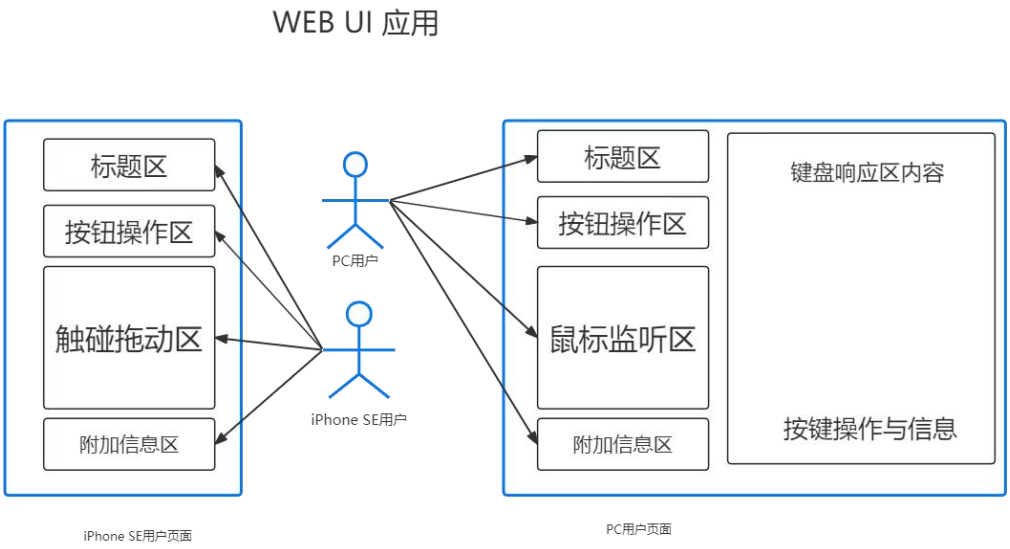

图 8-1 DOM 图

8.2 项目的实现和编程

JavaScript 部分代码：

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

    UI.appHeight = window.innerHeight;

    let baseFont = UI.appWidth / 20;
    //通过改变 body 对象的字体大小，这个属性可以影响其后代
    document.body.style.fontSize = baseFont + "px";
    //通过把 body 的高度设置为设备屏幕的高度，从而实现纵向全屏
    //通过 CSS 对子对象百分比（纵向）的配合，从而达到我们响应式设计的目标
    document.body.style.width = UI.appWidth - baseFont + "px";
    document.body.style.height = UI.appHeight - baseFont * 5 + "px";

    if (window.innerWidth < 1000) {
        $("aid").style.display = 'none';
    }
    $("aid").style.width = window.innerWidth - UI.appWidth - baseFont * 3 + 'px';
    $("aid").style.height = UI.appHeight - baseFont * 3 + 'px';
```

```
    //尝试对鼠标和触屏设计一套代码实现 UI 控制
    var Pointer = {};
    Pointer.isDown = false;
    Pointer.x = 0;
    Pointer.deltaX = 0;
    { //Code Block Begin
      let handleBegin = function (ev) {
        Pointer.isDown = true;

        if (ev.touches) {
          console.log("touches1" + ev.touches);
          Pointer.x = ev.touches[0].pageX;
          Pointer.y = ev.touches[0].pageY;
          console.log("Touch begin : " + "(" + Pointer.x + "," + Pointer.y + ")");
          $("bookface").textContent = "触屏事件开始，坐标： " + "(" + Pointer.x + "," +
Pointer.y + ")";
        } else {
          Pointer.x = ev.pageX;
          Pointer.y = ev.pageY;
          console.log("PointerDown at x: " + "(" + Pointer.x + "," + Pointer.y + ")");
          $("bookface").textContent = "鼠标按下，坐标：" + "(" + Pointer.x + "," + Pointer.y +
")";
        }
      };
      let handleEnd = function (ev) {
        Pointer.isDown = false;
        ev.preventDefault()
        let direction = Pointer.deltaX > 0 ? 'right' : 'left';
        //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！";
            $("bookface").style.left = '7%';
            if (direction === 'right') {
              showNextImage();
            } else {
              showPrevImage();
            }

          } 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！";
                $("bookface").style.left = '7%';
                if (direction === 'right') {
                    showNextImage();
                } else {
                    showPrevImage();
                }
            } 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);

        //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
        var text = $("typeText");
        $("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;
            if (ev.keyCode == 13) {
                let pChild = document.createElement("p");
                $("aid").appendChild(pChild);
                pChild.className = 'newText';
                text = pChild;
            }
            $("keyStatus").textContent = key + "  键已弹起";
            if (key === "Backspace") {
                text.textContent = text.textContent.slice(0, -1);
            }
            if (printLetter(key)) {
                text.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 $
var myDiv = document.getElementById('bookface');
var images = [
    "../lesson/CS.jpg",
    "../lesson/CSS.jpg",
    "../lesson/linuxCMD.jpg",
    "../lesson/Http.jpg"
];
var pdf = [
    "../pdf/CS.pdf",
    "../pdf/CSS.pdf",
    "../pdf/linuxCMD.pdf",
    "../pdf/Http.pdf"
];
//当前位置
var currentImageIndex = 0;
```

```
    // 定义下一个图片的函数
    function showNextImage() {
        currentImageIndex++;
        if (currentImageIndex == images.length) currentImageIndex = 0;
        myDiv.style.backgroundImage = 'url(' + images[currentImageIndex] + ')';
    }

    // 定义上一个图片的函数
    function showPrevImage() {
        currentImageIndex--;
        if (currentImageIndex == -1) currentImageIndex = images.length - 1;
        myDiv.style.backgroundImage = 'url(' + images[currentImageIndex] + ')';
    }

    // 将函数绑定到点击事件
    document.getElementById("next").onclick = showNextImage;
    document.getElementById("prev").onclick = showPrevImage;
    document.getElementById("about").onclick = function () {
        var pdfUrl = 'path/to/your/document.pdf';
        // 使用 window.open 打开 PDF 文件
        window.open(pdf[currentImageIndex], '_blank');
    }
</script>
```

8.3 项目的运行和测试

该项目在 PC 上的运行效果图如图 8-3 所示。

图 8-3 PC 运行效果图

该项目在窄屏终端 iPhone SE 上的运行效果图如图 8-4 所示。

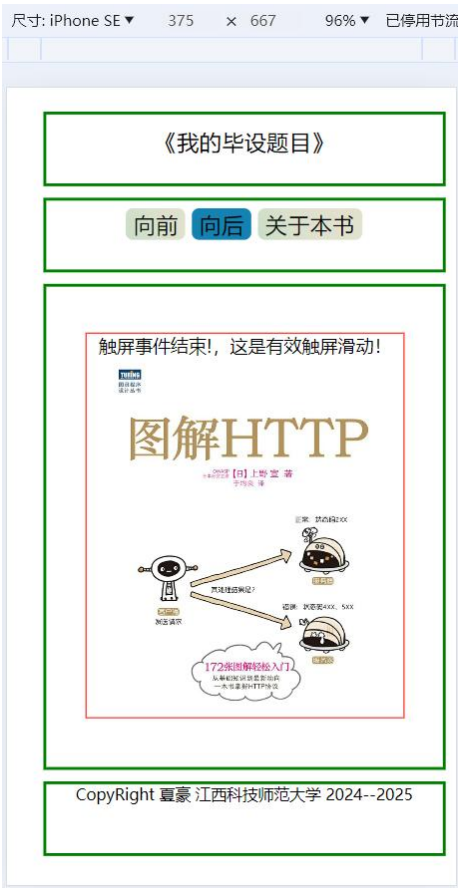

图 8-4 iPhone SE 运行效果图

该项目的网页二维码如图 8-5 所示。

图 8-5 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。

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ cd webUI
$ touch 1.6.html
```

编写好 1.6.html 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 的个性化键盘交互控制的设计开发，对键盘响应操作进 行优化，enter 键可以实现换行，backspace 可以实现删除。
```

成功提交代码后，gitbash 的反馈如下图 8-6 所示：

```
Us@LAPTOP-NVC06L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (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，对键盘响应操作进
行优化，enter键可以实现换行，backspace可以实现删除。
[master b9e8cc2] 项目第六版：UI的个性化键盘交互控制的设计开发，对键盘响应操作进
行优化，enter键可以实现换行，backspace可以实现删除。
 1 file changed, 398 insertions(+)
 create mode 100644 exp/1.6.html
```

图 8-6 代码提交反馈图

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图 8-7 所示：

```
Us@LAPTOP-NVC06L57 MINGW64 /d/学习资料/MrLi论文/code/new/WebUI (master)
$ git log
commit b9e8cc201fbc3f2d6d33e391975ee51272526f37 (HEAD -> master)
Author: 江科师大夏豪 <19880041639@163.com>
Date:   Thu Jun 13 21:55:30 2024 +0800

    项目第六版：UI的个性化键盘交互控制的设计开发，对键盘响应操作进行优化，enter
键可以实现换行，backspace可以实现删除。
```

图 8-7 代码提交日志

9. 谈谈本项目中的高质量代码

这是一本关于指导计算机操作的书籍。如今计算机几乎与螺丝刀一样普及，但它们要复杂得多，让它们按照你的意愿行事并不总是那么容易。 如果你的任

务对计算机来说是常见且已被充分理解的，比如查看电子邮件或充当计算器，你可以打开相应的应用程序并开始工作。但对于独特或开放式任务，很可能并没有现成的应用程序可用。 这时，编程或许就能派上用场了。编程是指构建程序的行为——一套精确的指令，告诉计算机该做什么。由于计算机是愚钝且死板的机器，编程从根本上说是单调乏味且令人沮丧的。幸运的是，如果你能接受这一事实，甚至可能享受以计算机能够处理的严格逻辑进行思考的乐趣，那么编程也会是值得的。它使你能够在几秒钟内完成手工永远无法完成的事情，是一种让你的计算机工具执行之前无法实现任务的方式，并且提供了一种抽象思维的绝佳锻炼[6]。

9.1 MVC 设计模式的践行

在这个项目中，我们创建了一个 Pointer 对象，并通过函数封装了对鼠标和触屏事件的处理逻辑。这种方法符合 MVC（Model-View-Controller）设计模式的原则，将事件处理逻辑（Controller）与 UI 展示（View）分离开来。

Model：数据和状态的管理，例如 Pointer 对象的创建和管理。

View：UI 部分，包括 HTML 和 CSS 部分，展示页面的结构和样式。

Controller：事件处理逻辑，通过事件监听器处理用户交互。

这种分离提高了代码的可读性和可维护性。

9.2 面向对象思想

项目中的 Pointer 对象封装了与指针相关的数据和方法，将复杂的逻辑分解为小的、独立的单元。这种面向对象的编程思想使代码更加模块化，易于理解和扩展。

```
var Pointer = {};
Pointer.isDown = false;
Pointer.x = 0;
Pointer.deltaX = 0;
```

9.3 函数封装与局部变量

项目中使用了多个函数来封装特定的操作逻辑，例如 handleBegin、handleEnd 和 handleMoving 等函数。这些函数通过闭包或局部变量的使用，避免了全局变量的滥用，减少了命名冲突的可能性。

```
let handleBegin = function (ev) {
```

```
  Pointer.isDown = true;
};
```

这种方式不仅提高了代码的模块化程度，还增强了其可测试性和可维护性。

9.4 逻辑清晰、抽象明确

在处理键盘事件时，项目通过 printLetter 函数对按键的合法性进行检查，并且实现了对不同按键（如字母、数字、标点符号）的统一处理。这种抽象化的逻辑设计使得代码更具通用性和复用性。

```
function printLetter(k) {
  if (k.length > 1) {
    return false;
  }
  let puncs = ['~', '`', '!', '@', '#', '$', '%', '^', '&', '*', '(',
')', '-', '_', '+', '=', ',', '.', ';', ';', '<', '>', '?', '/', '
', '\'', '\"'];
  if ((k >= 'a' && k <= 'z') || (k >= 'A' && k <= 'Z') || (k >= '0'
&& k <= '9')) {
    return true;
  }
  for (let p of puncs) {
    if (p === k) {
      return true;
    }
  }
  return false;
}
```

9.5. 响应式设计

通过计算并设置 body 的高度和字体大小，代码实现了简单的响应式设计，使页面在不同设备上都能有良好的显示效果。

```
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";
```

这种设计方式提高了代码的灵活性和适应性。

本项目通过合理的 MVC 设计模式、面向对象的编程思想、函数封装与局部变量的使用，以及清晰的逻辑抽象，实现了对鼠标和触屏事件的高效处理。这些高质量代码的特点不仅提高了代码的可读性和可维护性，还增强了其扩展性和复用性。通过这些设计原则，项目能够在多种设备和场景下顺利运行，展示了良好的编程实践。

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

当我们提到命令行时，实际上指的是壳（Shell）。壳是一个程序，它接收键盘输入的命令并将其传递给操作系统执行。几乎所有的 Linux 发行版都提供了一个来自 GNU 项目的壳程序，名为 bash。这个名字是一个首字母缩写，意为“bourne-again shell”，指的是 bash 是 sh（由 Steve Bourne 编写的原始 Unix 壳程序）的一个增强替代品[7]。

与 Windows 类似，类 Unix 操作系统如 Linux 以一种称为层次目录结构的方式来组织其文件。这意味着它们以树状模式组织目录（在其他系统中有时称为文件夹），这些目录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，这些子目录又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

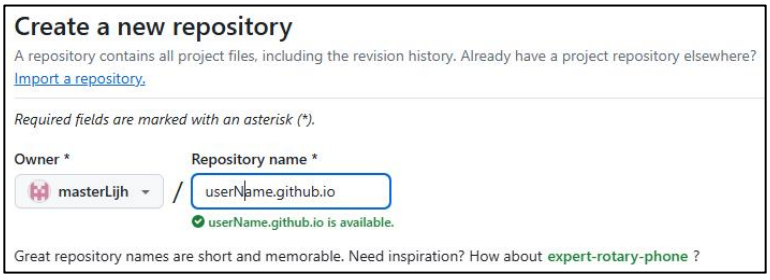

图 10-1 创建仓库界面图

图 10-2 创建按钮图

如上图 10-1 和图 10-2 所示，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 git@github.com:1TzUs/us001.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 10-3 所示：

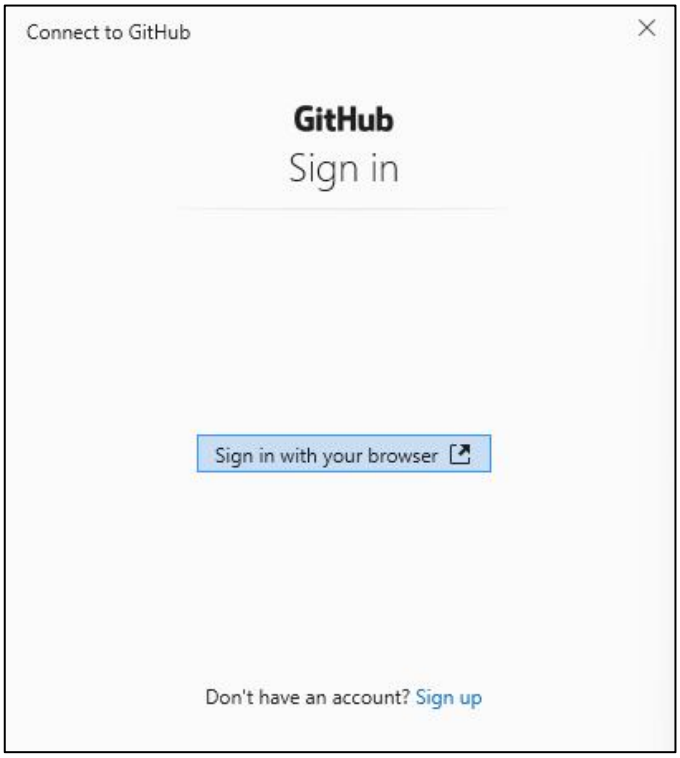

图 10-3 GitHub 注册页面图

再次确认授权 gitBash 拥有访问改动远程代码的权限，如下图 10-4 所示：

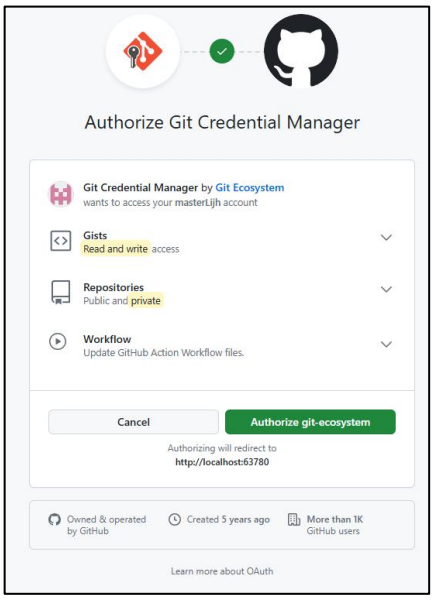

图 10-4 授权图

最后，如图 10-5 GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。

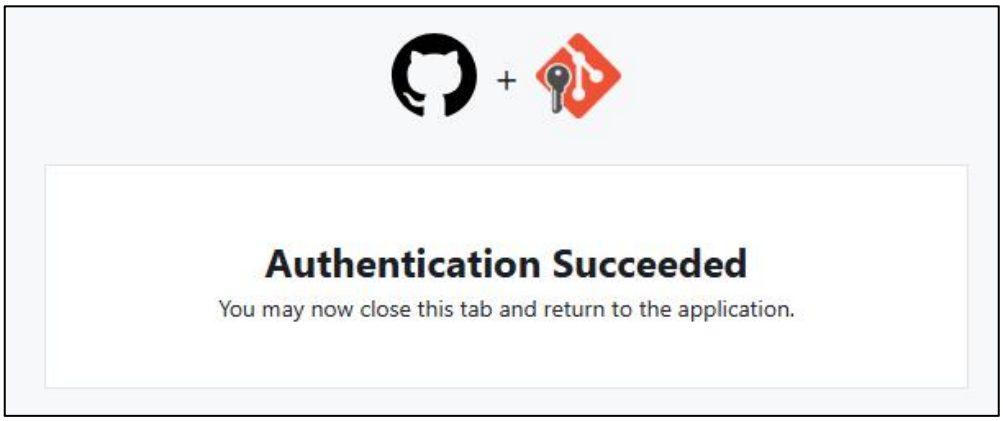

图 10-5 链接成功图

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 Chrome 浏览器打开，本文截取操作中间的效果图，如下图 10-6 所示：

图 10-6 效果图

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