

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 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：刘苗苗

学 号：20213665

指导教师：李健宏

2024 年 6 月 18 日

目录

| | |
|-------------------------------------|----|
| 1.前言 | 1 |
| 1.1 毕设任务分析..... | 1 |
| 1.2 研学计划..... | 2 |
| 1.3 研究方法..... | 3 |
| 2.技术总结和文献综述..... | 3 |
| 2.1Web 平台和客户端技术概述..... | 3 |
| 2.1.1 发展历程..... | 4 |
| 2.1.2Web 平台与 Web 编程 | 4 |
| 2.2 项目的增量式迭代开发模式..... | 5 |
| 2.2.1 瀑布模型..... | 5 |
| 2.2.2 增量模型..... | 6 |
| 3.内容设计概要..... | 7 |
| 3.1 分析和设计..... | 7 |
| 3.2 项目的实现和编程..... | 7 |
| 3.3 项目的运行和测试..... | 8 |
| 3.4 项目的代码提交和版本管理..... | 9 |
| 4.移动互联时代的 UI 开发初步——窄屏终端的响应式设计 | 10 |
| 4.1 分析和设计..... | 10 |
| 4.2 项目实施和编程..... | 11 |
| 4.3 项目运行和测试..... | 12 |
| 4.4 项目的代码提交和版本管理..... | 13 |
| 5.应用响应式设计技术开发可适配窄屏和宽屏 UI..... | 14 |
| 5.1 分析和设计..... | 14 |
| 5.2 项目实施和编程..... | 15 |
| 5.3 项目运行和测试..... | 18 |
| 5.4 项目的代码提交和版本管理..... | 19 |
| 6.个性化 UI 设计中对鼠标交互的设计开发..... | 20 |
| 6.1 分析和设计..... | 20 |

| | |
|--|----|
| 6.2 项目实现和编程..... | 20 |
| 6.3 项目运行和测试..... | 23 |
| 6.4 项目的代码提交和版本管理..... | 24 |
| 7.对触屏和鼠标的通用交互操作的设计开发..... | 24 |
| 7.1 分析和设计..... | 24 |
| 7.2 项目实现和编程..... | 25 |
| 7.3 项目运行和测试..... | 28 |
| 7.4 项目的代码提交和版本管理..... | 29 |
| 8.UI 的个性化键盘交互控制的设计开发..... | 30 |
| 8.1 分析和设计..... | 30 |
| 8.2 项目实现和编程..... | 31 |
| 8.3 项目运行和测试..... | 32 |
| 8.4 项目的代码提交和版本管理..... | 34 |
| 9.用 gitBash 工具管理项目的代码仓库和 http 服务器..... | 35 |
| 9.1 经典 Bash 工具介绍..... | 35 |
| 9.2 通过 gitHub 平台实现本项目的全球域名..... | 35 |
| 9.3 创建一个空的远程代码仓库..... | 36 |
| 9.4 设置本地仓库和远程代码仓库的链接..... | 36 |
| 参考文献: | 39 |

基于 Web 客户端技术的个性化 UI 的设计和实现

(Customized UI design and Programming based on Web client technology)

科师大元宇宙产业学院 2021 级 计算机科学与技术 3 班 刘苗苗

摘要：近十年来，以 html5 为核心的 web 标准软件开发技术被广泛应用，其以跨平台、开源的优势在多个领域受到青睐。通过对本次任务的分析，本项目选择 html5 的 web 客户端技术为技术路线，展开了对程序设计和软件开发的研究和实践。通过广泛研究相关书籍、论坛和文献，设计了一个个性化的用户界面（UI）应用程序。项目中，我运用 html 语言进行内容建模，css 语言设计 UI 外观，javascript 语言实现交互功能，除直接使用了 web 客户端最底层的 API 外，全部代码皆由手工逐条编写，未使用任何框架和库。项目采用响应式设计，适应了移动互联网时代多样化的屏幕需求。同时运用面向对象思想，构建通用 pointer 模型，实现对鼠标和触屏的统一控制，提升代码质量。从工程管理上看，项目采用增量式开发模式，经过六次代码重构（A:Analysis, D:Design, I:Implementation, T:Testing），愉快地完成了对该项目的设计、开发和测试。从代码开源的角度上看，项目使用 git 工具进行版本管理，利用 gitbash 工具将该项目的代码仓库提交至 github 平台上，最后借助 github 的 http 服务器，实现该 UI 应用在全球互联网的部署，用户仅需通过地址和二维码，便可实现对该程序的跨平台的高效访问。

关键字：html; web; pointer; gitbash; github

1.前言

1.1 毕设任务分析

毕业论文对于本科生而言，是一次全面展示其在大学期间所学知识和技能

机会。它要求学生针对一个具体的课题进行深入研究，从而培养独立分析和解决问题的能力。在撰写过程中，学生需要进行广泛的文献调研，掌握最新的研究动态，这有助于他们了解学科前沿，提高研究素养。同时，毕业论文也是学生进行数据收集、实验设计和结果分析的实践过程，这些技能对于未来的科研或工程工作至关重要。此外，毕业论文的撰写还能锻炼学生的组织和表达能力，包括如何清晰地阐述观点、逻辑严密地构建论文结构以及如何进行有效的学术交流。在这一过程中，学生还能学习到项目管理的技巧，如时间管理、任务分配和进度控制，这些能力对于任何职业发展都是宝贵的资产。

最后，一篇优秀的毕业论文不仅是学生学术成就的证明，也是他们专业能力的体现，能够在求职时增加竞争力，为学生打开更广阔的职业道路。因此，毕业论文在本科教育中扮演着至关重要的角色，是学生综合运用所学知识、技能和方法解决实际问题的重要途径。

1.2 研学计划

为了更有效地管理我的毕业设计进度，我制作了一个详细的计划表，即表格 1。这个表格不仅列出了整个毕业设计过程中的关键任务和时间安排，还为每个阶段设定了明确的时间节点和目标。通过这种方式，我可以清晰地追踪每个任务的完成情况，确保毕业设计的每一部分都能按时完成。

表格 1 研学计划安排表

| 时间安排 | 目标 |
|------|---|
| 第一周 | “三段式”内容的设计开发概要 |
| 第二周 | 实现网页在窄屏终端的响应式设计，使页面能适应小的屏幕 |
| 第三周 | 实现页面在窄屏和宽屏设备上的适应性显示 |
| 第四周 | 在初步的 UI 设计中实现鼠标模型的设计，动态捕捉鼠标移动和点击的操作，并精确显示鼠标坐标 |
| 第五周 | 实现鼠标的横向滑动效果，模拟了在手机中的触屏拖拽效果 |
| 第六周 | 实现与键盘之间交互控制功能，将键盘输入的字符动态的显示在项目指定界面中 |

1.3 研究方法

在这个项目的构建上，我采取了一种多维度的研究方法，确保了研究的实用性和严谨。首先，我通过翻阅了大量关于 HTML5、CSS、JavaScript 及其相关技术的资料，初步掌握这个领域的最新动态和研究成果，为技术的选择和开发提供坚实的理论基础。同时也使用了原型设计法，用专业工具设计了界面原型，确保设计方案切实可行。然后，项目采用了增量开发法，就是用迭代增量的方式开发，把整个项目拆分成一系列小模块，逐个开发和测试。每次迭代，都会对已有的功能进行优化，这样既能保证项目质量，又能确保开发进度。最后，项目使用了实验测试法和版本控制法。在浏览器的开发者工具里进行调试和修改，同时用 Git 来管理代码版本，确保代码的稳定性和可追溯性。本项目还通过 GitHub 这个平台，对项目进行托管和全球部署，让项目能够更好地服务于全球用户。

2.技术总结和文献综述

2.1Web 平台和客户端技术概述

Web 平台和客户端技术是互联网发展史上的两个重要支柱，它们共同推动了网络应用的多样化和用户体验的提升。客户端技术，主要指的是运行在用户设备上的软件，如桌面应用程序、移动应用等。它们通常提供更加丰富的用户界面和更强大的功能，能够更好地利用设备的硬件资源。客户端技术的发展与操作系统和硬件的进步紧密相关，随着智能手机和平板电脑的普及，移动客户端技术尤其得到了迅猛发展。

Web 平台和客户端技术的主要作用在于提供用户界面和交互方式，使用户能够访问和操作存储在服务器上的数据和服务。Web 平台的优势在于跨平台性，用户无需安装即可通过浏览器访问应用；而客户端技术则提供了更接近本地应用的体验，能够提供更丰富的功能和更好的性能。

2.1.1 发展历程

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”。

1995 年，JavaScript 的引入为 Web 页面添加了动态特性，使得用户可以在不刷新页面的情况下与 Web 应用交互。随后，DHTML（动态 HTML）和 CSS（层叠样式表）的出现进一步丰富了 Web 页面的表现力和布局能力。Ajax 技术的诞生标志着 Web 应用可以进行异步数据交互，极大地提升了用户体验。

进入 21 世纪，随着智能手机和移动设备的普及，Web 技术开始向移动优先（Mobile First）的方向发展。HTML5 和 CSS3 的推出，为 Web 平台带来了更多的功能，如地理位置、多媒体播放、本地存储等，使得 Web 应用能够更好地适应不同设备和场景。

2.1.2 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 编程的核心语言包括 HTML、CSS 和 JavaScript。HTML 定义了网页的结构和内容，CSS 负责样式和布局，而 JavaScript 则添加了交互性，允许用户与网页进行动态交互。随着 Web 技术的发展，出现了许多新的编程范式和框架，如响应式设计、前端框架（例如 React、Vue 和 Angular）和后端技术（如 Node.js、Django 和 Ruby on Rails）。这些技术使得 Web 编程不再局限于简单的页面展示，而是可以构建复杂的单页应用（SPAs）、Web 服务和全栈应用程序。Web 平台的另一个重要方面是 Web API，它们提供了访问浏览器功能（如地理位置、摄像头和存储）和服务器端资源的接口。

此外，Web 平台还包括了一系列安全协议和标准，如 HTTPS、TLS/SSL 和 CORS，它们确保了数据传输的安全性和隐私保护。Web 编程中，开发者需要遵循这些安全最佳实践，以防止诸如跨站脚本攻击（XSS）和 SQL 注入等安全威胁。

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）。

2.2.1 瀑布模型

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]。瀑布模型是最早出现的软件开发模型，是所有其他软件开发模型的基础框架。与软件的生命周期不同的是，它缺少了软件运行维护阶段，瀑布式开发模型如下图 2-1 所示：

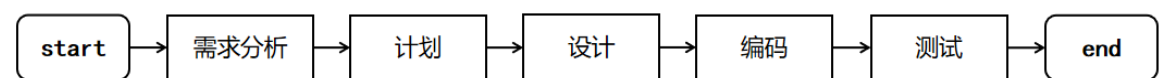


图 2-1 瀑布模型

2.2.2 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]。增量模型支持需求的逐步明确和变更适应，降低了项目风险，因为它允许在开发周期的早期阶段识别和解决问题。本项目使用的就是运动了增量式的开发模型，对项目进行了 6 次优化，最后得到一个完整的项目，增量式开发模型如下图 2-2 所示。

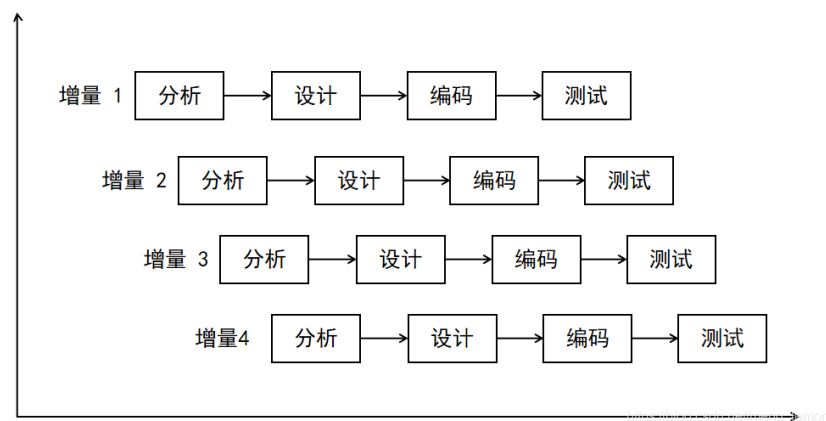


图 2-2 增量式模型

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。如图 3-1 用例图所示：

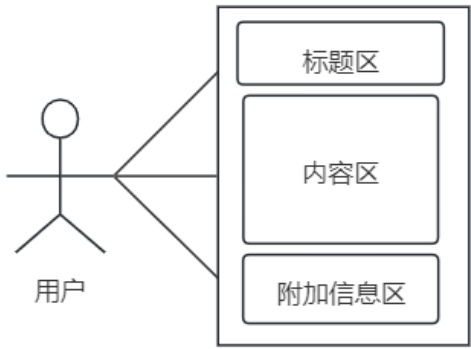


图 3-1 三段式设计开发与用例图

3.2 项目的实现和编程

(1) HTML 代码编

在本阶段的开发中，我采用了 HTML 语言来精心设计页面结构，将其划分为三个主要部分：标题区域、内容区域和脚注区域。在标题区域，我使用了<header>标签来定义，这不仅为页面提供了一个明确的顶部标识，还有助于用户快速了解页面的主题。内容区域作为页面的核心，使用了<main>标签进行包裹，确保了主要内容的突出显示。而脚注区域则通过<footer>标签来实现，它包含了我的个人信息，如代码块 3-1 所示。

```
<header>
    《我的毕设题目》
</header>
<main>
```

```
        我的毕设内容
    </main>
</footer>
    Copyright 刘苗苗 江西科技师范大学 2024-2025
</footer>
```

代码块 3-1

(2) CSS 代码编写

我使用了简单的 CSS 语言为页面设置了边框，通过 CSS 的简洁而强大的语法，我能够轻松地定义边框的宽度、样式和颜色，从而为页面的各个部分添加了清晰的视觉效果，增强了页面的美观性和层次感，如代码块 3-2 所示。

```
<style>
    body{
        font-size: 30px;
    }
    header{
        border: 2px solid purple;
        height: 200px;
    }
    main{
        border: 2px solid purple;
        height: 400px;
    }
    footer{
        border: 2px solid purple;
        height: 100px;
    }
    body{
        text-align:center;
    }
</style>
```

代码块 3-2

3.3 项目的运行和测试

项目的运行和测试至少要通过二类终端，本文此处仅给出 PC 端用 Chrome 浏览器打开项目的结果，如下图 3-2 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 3-3 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 3-2PC 端运行效果图



图 3-3 一阶段移动端二维码

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。

(1) 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 刘苗苗
$ git config user.email 910519723@qq.com
$ touch index.html myCss.css
```

(2) 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 的反馈如下图 3-4 所示：

```
86137@LAPTOP-ONNA3P69 MINGW64 /d/abc/exp (master)
$ git commit -m 项目第一版：“三段论”的内容设计概要开发
[master 205b363] 项目第一版：“三段论”的内容设计概要开发
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 exp/index.html
create mode 100644 exp/mycss.css
```

图 3-4 一阶段代码提交

(3) 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 3-5 所示所示：



```
86137@LAPTOP-ONNA3P69 MINGW64 /d/webUI (master)
$ git log
commit 5d9359748b276eae7adfe973e3be020993304acd (HEAD -> master)
Author: 刘苗苗 <910519723@qq.com>
Date: Fri Jun 7 19:15:12 2024 +0800

    项目第一版：“三段论”的内容设计概要开发
```

图 3-5 一阶段仓库日志

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

在计算机世界里，显示器的种类繁多，尺寸和分辨率这些特性，很大程度上取决于你愿意掏多少钱。所以网络设计师通常会制定一套通用的布局规则，然后让浏览器去决定如何在不同电脑上展示这些页面，而不会去为每种显示器都准备一个专门的网页版本，那样太不现实了^[1]。

这种让浏览器来决定细节的做法，有时候会导致一些有趣的现象：同样的网页，在不同的浏览器或者不同硬件配置的电脑上打开，可能会呈现出不同的效果。例如，如果一个屏幕比另一个宽，那么显示的文本行长度或者图像大小可能就会有所不同。关键在于，网页提供了一个基本的展示框架，而具体的展示细节则由浏览器来填充。

在上一阶段，我们专注于设计了一个只适应电脑端的 UI 界面。为了进一步满足界面在不同设备上的适应性需求，本阶段我计划开发一个能够适配各种窄屏设备的 UI 界面。这种设计能够确保用户无论使用何种窄屏设备，都能获得一致的用户体验。页面设计的具体细节和布局已在图 4-1 中展示。

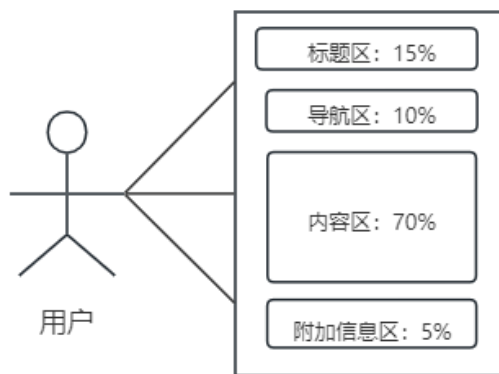


图 4-1 窄屏终端的响应式用例图

4.2 项目实现和编程

(1) CSS 代码编写

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:

```
*{  
  margin: 10px;  
  text-align: center;  
}  
  
header{  
  border: 2px solid purple;  
  height: 15%;  
  font-size: 1.66em;  
}  
  
main{  
  border: 2px solid purple;  
  height: 70%;  
  font-size: 1.2em;  
}  
  
nav{  
  border: 2px solid purple;  
  height: 10%;
```

```

    }
    nav button{
font-size: 1.1em;
}
    footer{
        border: 2px solid purple;
        height: 5%;
    }

```

代码块 4-1

(2) JavaScript 代码编写

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 = 33;
    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 项目运行和测试

在进行测试的过程中，我特别选取了 iPhone SE 和 Samsung Galaxy S8+这两款窄屏设备，用以测试和展示代码的运行效果，如图 4-2 和图 4-3 所示。实验结果证实，所编写的代码成功地实现了项目界面在这些不同窄屏设备上的响应式布局。此外，我还将项目在开发过程中的阶段性文件上传至 GitHub 网站。移动端用户现在可以通过扫描图 4-4 中的二维码，体验并测试该项目的第二次开发阶段

的成果。



图 4-2iPhoneSE 页面显示图



图 4-3Samsung Galaxy S8+页面显示图



图 4-4 二阶段移动端二维码

4.4 项目的代码提交和版本管理

- (1) 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：实现了网页在窄屏终端的响应式设计，使页面能适应小的屏幕



成功提交代码后，gitbash 的反馈如下图 4-5 所示：

```
86137@LAPTOP-ONNA3P69 MINGW64 /d/webUI (master)
$ git commit -m项目第二版：实现了网页在窄屏终端的响应式设计，使页面能适应不同尺寸的屏幕
[master 95476ba] 项目第二版：实现了网页在窄屏终端的响应式设计，使页面能适应不同尺寸的屏幕
1 file changed, 78 insertions(+)
create mode 100644 1.2.html
```

图 4-5 二阶段代码提交

(2) 输入日志命令查看项目代码仓库的提交记录

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：

```
commit 95476bafc7d495dbbb8ac6b8d45551c8f4a4884a (HEAD -> master)
Author: 刘苗苗 <910519723@qq.com>
Date: Tue Jun 11 14:13:19 2024 +0800

 项目第二版：实现了网页在窄屏终端的响应式设计，使页面能适应不同尺寸的屏幕
```

图 4-6 二阶段仓库日志

## 5.应用响应式设计技术开发可适配窄屏和宽屏 UI

### 5.1 分析和设计

在上一阶段，我们成功实现了用户界面(UI)在窄屏设备上的响应式布局。然而，为了满足现代化 UI 界面的需求，我们的目标是确保它不仅在窄屏设备上，也能在 PC 端提供出色的适应性。因此，在当前阶段，我专注于开发了一套能够同时适应窄屏和宽屏设备的 UI 界面。这一阶段的用例图已在图 5-1 中展示，它展示了我们如何确保用户界面在不同设备上都能提供一致的用户体验。

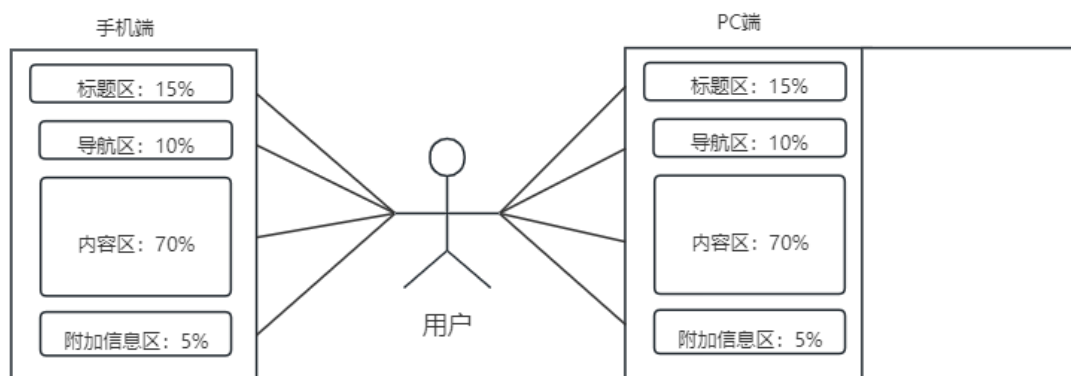


图 5-1 响应式设计用例图

## 5.2 项目实现和编程

### (1) html 代码编写

该阶段我增加页面导航按钮，用于项目后续的翻页功能，并且在页面右边设置了一个空白区，用于项目后续添加键盘交互功能。

```
<html>
<body>
 <header>
 <p id="book">
 我的毕设题目
 </p>
 </header>
 <nav>
 <button>导航一</button>
 <button>导航二</button>
 <button>导航三</button>
 </nav>
 <main id = 'main'>
 <div id="bookface">
 软件内容区域
 </div>
 </main>

 <footer>
 <p id="statusInfo">
```

CopyRight from 刘苗苗 江西科技师范大学 2021--2025

```
</p>
</footer>
<div id="aid">
 <p>用户键盘响应区</p>
 <p id="keyboard"></p>
</div>
</body>
</html>
```

代码块 5- 1

## (2) CSS 代码编写

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

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

 main{
 border: 2px solid purple;
 height: 70%;
 font-size: 1.2em;
 }

 nav{
 border: 2px solid purple;
 height: 10%;
 }

 nav button{
 font-size: 1.1em;
 }

 footer{
 border: 1px solid purple;
 height: 5%;
 }

 body{
 position: relative ;
 }
```

```

#aid{
position: absolute;
border: 3px solid purple;
top: 0.5em;
left: 600px;
}
#bookface{
width: 80%;
height: 100%;
border: 1px solid purple;
background-color: blanchedalmond;
margin: auto;
}
</style>

```

代码块 5-2

### (3) JavaScript 代码编写

代码首先设置了 `UI.appWidth` 的值，它是窗口内宽度和 600 像素之间的较小值，确保应用的宽度不会超过 600 像素。`UI.appWidth` 则直接设置为窗口内的高度。接着，代码定义了两个常量 `LETTERS` 和 `baseFont`。`LETTERS` 是一个数字常量，值为 33，而“`baseFont`”则是根据“`UI.appWidth`”和“`LETTERS`”计算得出的基础字体大小。代码通过直接修改 `document.body.style` 属性，将 `body` 元素的字体大小设置为 `baseFont` 像素，实现字体大小的全局设置。同时，将 `body` 元素的宽度和高度设置为减去一定 `baseFont` 值后的窗口宽度和高度，以实现全屏效果，并配合 CSS 的百分比布局实现响应式设计。

接下来，代码检查窗口的内宽度是否小于 900 像素，如果是，则隐藏 ID 为 `aid` 的元素。然后，为 `aid` 元素设置宽度和高度，宽度是窗口内宽度减去 `UI.appWidth` 和两倍 `baseFont` 的值，高度则是 `body` 元素的客户端高度，如代码块 5-3 所示。

```

<script>
var UI = {};
UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth;
UI.appHeight = window.innerHeight;
const LETTERS = 33;
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";
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';
}

```

代码块 5-3

### 5.3 项目运行和测试

在进行项目测试时，我特别选取了 PC 端服务器作为宽屏设备的代表，以及 iPhone SE 作为窄屏设备的代表，对项目代码进行了全面的测试。测试结果如图 5-2 和图 5-3 所示，证明了代码能够成功实现 UI 界面在宽屏和窄屏设备上的适应性布局。此外，为了便于移动端用户访问和体验，该项目在开发过程中的阶段性文件已经上传至 GitHub。现在，用户只需扫描图 5-4 中的二维码，即可直接在移动设备上体验并测试项目在第三次开发阶段所取得的成果。



图 5-2PC 端运行效果图



图 5-3phone S 设备运行效果图



图 5-4 三阶段移动端二维码

## 5.4 项目的代码提交和版本管理

(1) 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：实现了页面在窄屏和宽屏设备上的适应性显示
```

成功提交代码后，gitbash 的反馈如下图 5-5 所示：

```
86137@LAPTOP-ONNA3P69 MINGW64 /d/webUI (master)
$ git commit -m 项目第三版：实现了页面在窄屏和宽屏设备上的适应性显示
[master 2898817] 项目第三版：实现了页面在窄屏和宽屏设备上的适应性显示
1 file changed, 20 insertions(+), 21 deletions(-)
```

图 5-5 三阶段代码提交

(2) 输入日志命令查看项目代码仓库的提交记录

```
$ git log
```

gitbash 反馈代码的仓库日志如下图 5-6 所示：



图 5-6 三阶段仓库日志

## 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

在设计用户界面时，鼠标的作用至关重要。因此，我深入研究了用户界面设计中的鼠标模型，特别是鼠标在屏幕上的坐标系。其目标是，当用户在屏幕上的特定区域点击和移动时鼠标时，界面能够即时显示鼠标的精确坐标，如下图 5-1 所示。这样的功能不仅提高了交互的直观性，还能在需要精确操作时提供帮助。

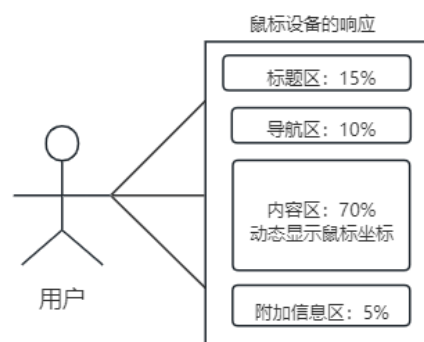


图 6-1 鼠标模型

### 6.2 项目实现和编程

#### (1) CSS 代码编写

```
<style>
 body{
 position:relative ;
 }
 #aid{
```

```

position: absolute;
border: 3px solid purple;
top: 0.5em;
left: 600px;
}
#bookface{
width: 80%;
height: 100%;
border: 1px solid purple;
background-color: blanchedalmond;
margin: auto;
}
</style>

```

代码块 6-1

(2) JavaScript 代码如下:

首先, 定义了一个名为 `mouse` 的对象, 包含三个属性: `isDown` 用于标记鼠标按钮是否被按下, 初始值为 `false`; `x` 记录鼠标的当前 X 坐标; `deltaX` 记录鼠标水平移动的差值, 初始值为 0。然后, 它为页面上的 `bookface` 元素添加了三个鼠标事件监听器: 当鼠标按下时, 记录并显示坐标、当鼠标移动时, 实时更新并显示当前坐标、当鼠标离开时, 显示鼠标已离开的消息, 如代码块 6-2 所示。

```

<script>
var UI = {};
UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth;
UI.appHeight = window.innerHeight;
const LETTERS = 33;
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";
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";

});

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 ;
 }
 }
}

```

代码块 6-2

6.3 项目运行和测试

在进行 UI 界面的测试过程中，我主要关注了鼠标与界面交互的几个关键动作：移动、点击以及离开。测试结果显示，UI 界面能够成功地捕捉到鼠标在界面中的这些不同状态，并且能够准确显示鼠标的坐标位置。这一功能的测试结果分别在图 6-2、图 6-3 和图 6-4 中进行了展示。此外，为了便于用户更直观地体验项目在第三阶段的开发成果，我们提供了图 6-5 中的二维码。用户只需使用移动设备扫描该二维码，即可直接体验项目在第四阶段的运行效果。



图 6-2 鼠标移动效果图

图 6-3 鼠标点击效果图

图 6-4 鼠标移开效果图



图 6-5 四阶段移动端二维码

## 6.4 项目的代码提交和版本管理

(1) 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：在初步的 UI 设计中实现了鼠标模型的设计，能动态捕捉鼠标移动和点击的操作，并精确显示鼠标坐标
```

成功提交代码后，gitbash 的反馈如下图 6-6 所示：

```
86137@LAPTOP-ONNA3P69 MINGW64 /d/webUI (master)
$ git commit -m 项目第四版：在初步的UI设计中实现了鼠标模型的设计，能动态捕捉鼠标移动和点击的操作，并精确显示鼠标坐标
[master c485b3c] 项目第四版：在初步的UI设计中实现了鼠标模型的设计，能动态捕捉鼠标移动和点击的操作，并精确显示鼠标坐标
1 file changed, 145 insertions(+)
create mode 100644 1.4.html
```

图 6-6 四阶段代码提交

(2) 输入日志命令查看项目代码仓库的提交记录

```
$ git log
```

gitbash 反馈代码的仓库日志如下图 6-7 所示：

```
commit c485b3c138a7c74037beb1a6d5887fdbe958924b (HEAD -> master)
Author: 刘苗苗 <910519723@qq.com>
Date: Tue Jun 11 22:00:12 2024 +0800

 项目第四版：在初步的UI设计中实现了鼠标模型的设计，能动态捕捉鼠标移动和点击的操作，并精确显示鼠标坐标
```

图 6-7 四阶段仓库日志

## 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

在上一章节，我们集中讨论了基于鼠标点击和移动模型，通过这种方式能够在电脑上捕捉到鼠标指针的确切位置。但这种方法并不适用于移动设备，因为在手机上我们通常使用触控操作。为了解决这个问题，本章节我们转向研究鼠标的横向滑动效果，尝试用它来模拟在手机屏幕上的拖拽动作。该阶段用例图如下

图 7-1 所示。

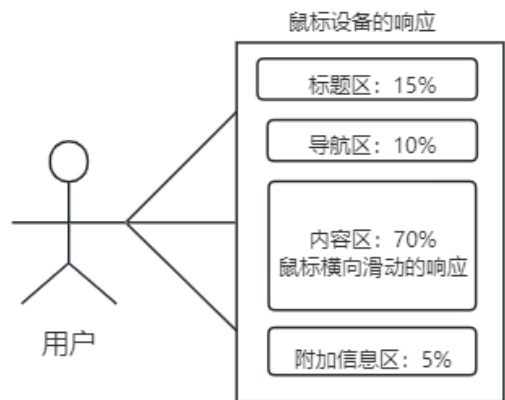


图 7-1 触屏与鼠标统一操作用例图

## 7.2 项目实现和编程

### (1) CSS 代码编写

使用 `position: relative` 属性;使得 `body` 元素的子元素在不脱离文档流的情况下进行偏移定位。如代码块 7-1 所示。

```
<style>
 body{
 position:relative ;
 }
 button{
 font-size:1em;
 }
 #aid{
 position: absolute;
 border: 3px solid purple;
 top: 0em;
 left: 600px;
 }
 #bookface{
 position: absolute;
 width: 80%;
 height: 80%;
 }
```

```

 border: 1px solid purple;
 background-color: blanchedalmond;
 left: 7%;
 top: 7%;
 }
</style>

```

代码块 7-1

## (2) JavaScript 代码编写

此段 JavaScript 代码通过绑定鼠标事件，实现了对 bookface 元素的拖拽功能。定义了一个 mouse 对象来监控鼠标的点击状态、当前位置以及水平方向上的偏移。当鼠标按下时触发 mousedown 事件，此时记录鼠标的起始坐标，并将 mouse.isDown 标记为激活状态，同时在 bookface 上显示鼠标按下的坐标。mousemove 事件在鼠标持续按下状态下触发，计算并更新鼠标的当前位置与起始坐标之间的水平偏移，实时反映在 bookface 元素的位移和文本描述上。mouseup 和 mouseout 事件在鼠标释放或移出元素时触发，评估鼠标的移动距离，若超过 100 像素则判定为有效拖拽，否则将元素位置重置并提示无效拖拽，如代码块 7-2 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 - baseFont + "px";
 document.body.style.height = UI.appHeight - baseFont*4 + "px";
 if(window.innerWidth<1000){
 $("aid").style.display='none';
 }

```

```

$("aid").style.width=window.innerWidth-UI.appWidth - baseFont*3 +'px';
$("aid").style.height= UI.appHeight - baseFont*3 +'px';

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

 $("bookface").textContent= "鼠标松开!";
 if(Math.abs(mouse.deltaX) > 100){
 $("bookface").textContent += "，这是有效拖动！ " ；
 }else{
 $("bookface").textContent += " 本次算无效拖动！ " ；
 $("bookface").style.left = '7%';
 }

});
$("bookface").addEventListener("mouseout",function(ev){
 ev.preventDefault();
 mouse.isDown=false;

 $("bookface").textContent= "鼠标松开!";
 if(Math.abs(mouse.deltaX) > 100){
 $("bookface").textContent += " 这次是有效拖动！ " ；
 }else{
 $("bookface").textContent += " 本次算无效拖动！ " ；
 $("bookface").style.left = '7%';
 }

});
$("bookface").addEventListener("mousemove",function(ev){
 ev.preventDefault();
 if (mouse.isDown){

```

```

 console.log("mouse isDown and moving");
 mouse.deltaX = parseInt(ev.pageX - mouse.x);
 $(".bookface").textContent= "正在拖动鼠标，距离： " + mouse.deltaX + "px 。 ";
 $(".bookface").style.left = mouse.deltaX + 'px' ;
 }

});

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

```

代码块 7-2

### 7.3 项目运行和测试

在本阶段的测试中，我特别关注了鼠标在页面上的拖动行为，包括页面水平方向的拖动效果，以及有效拖动与无效拖动的区分。测试结果表明，这些交互效果已经达到了预期目标，具体展示在图 7-2、图 7-3、图 7-4 和图 7-5 中。这些图像详细展示了不同拖动状态下的界面反馈。此外，本阶段的代码更新上传至 GitHub。用户现在可以通过扫描图 7-6 中的二维码，体验项目在第五阶段的运行效果。

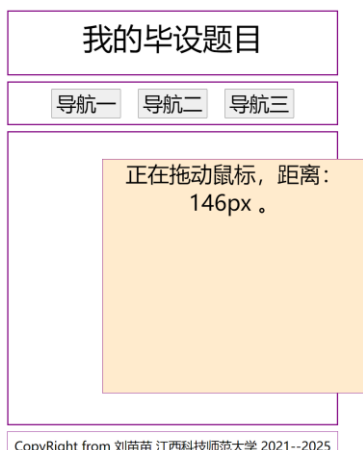


图 7-2 鼠标往右拖拽效果图

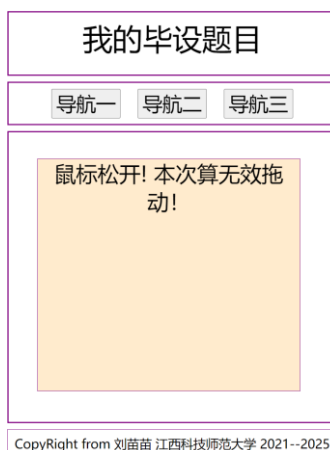


图 7-3 无效拖拽效果图

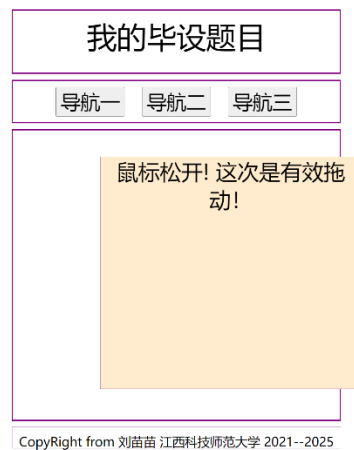


图 7-4 有效拖拽效果图

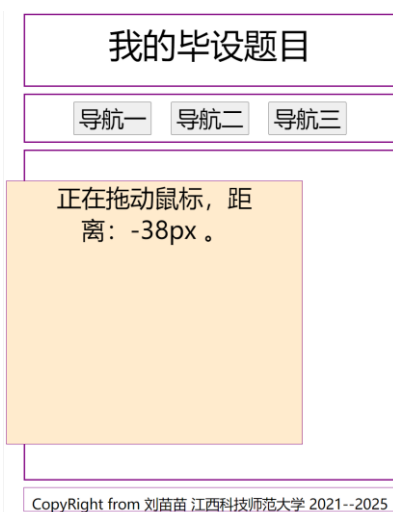


图 7-5 鼠标往左拖拽效果图



图 7-6 五阶段移动端二维码

## 7.4 项目的代码提交和版本管理

(1) 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：实现了鼠标的横向滑动效果，成功模拟了在手机中的触屏拖拽效果

成功提交代码后，gitbash 的反馈如下图 7-7 所示：



```
86137@LAPTOP-ONNA3P69 MINGW64 /d/webUI (master)
$ git commit -m 项目第五版：实现了鼠标的横向滑动效果，成功模拟了在手机
中的触屏拖拽效果
[master 641317c] 项目第五版：实现了鼠标的横向滑动效果，成功模拟了在手
机中的触屏拖拽效果
1 file changed, 188 insertions(+)
create mode 100644 1.5.html
```

图 7-7 五阶段代码提交

(2) 输入日志命令查看项目代码仓库的提交记录

```
$ git log
```

gitbash 反馈代码的仓库日志如下图 7-8 所示：

```
commit 641317c0b6921ead8c2b44e9ef45326e628a0f93 (HEAD -> master)
Author: 刘苗苗 <910519723@qq.com>
Date: Tue Jun 11 23:46:51 2024 +0800

 项目第五版：实现了鼠标的横向滑动效果，成功模拟了在手机中的触屏拖拽
 效果
```

图 7-8 五阶段仓库日志

## 8.UI 的个性化键盘交互控制的设计开发

### 8.1 分析和设计

在电脑的日常使用中，键盘的作用不可或缺，它作为主要的输入手段，与计算机之间进行着精细的信息传递。每当用户敲击键盘上的一个键，键盘便迅速向计算机发送信号。这个信号被计算机的键盘控制器捕获，并转换成一串数字或字符编码。之后，这些编码被发送到操作系统，操作系统依据其键盘布局或映射规则，将编码转换为可识别的字符或指令。完成转换后，操作系统将这些信息传递给当前正在运行的应用程序。应用程序接收到这些信息，会依据其设计逻辑执行相应的操作，例如在文本编辑器中将字符显示出来。

在这个阶段，我们的目标是开发一个具有个性化交互功能的键盘 UI，它不仅能够响应用户的输入，还能在界面上的特定区域实时显示用户所按下的字符，并且展示该字符对应的键值。这样的设计可以提高用户与界面的互动性，使用户能够直观地看到他们的操作结果，该阶段用例图如下图 8-1 所示。

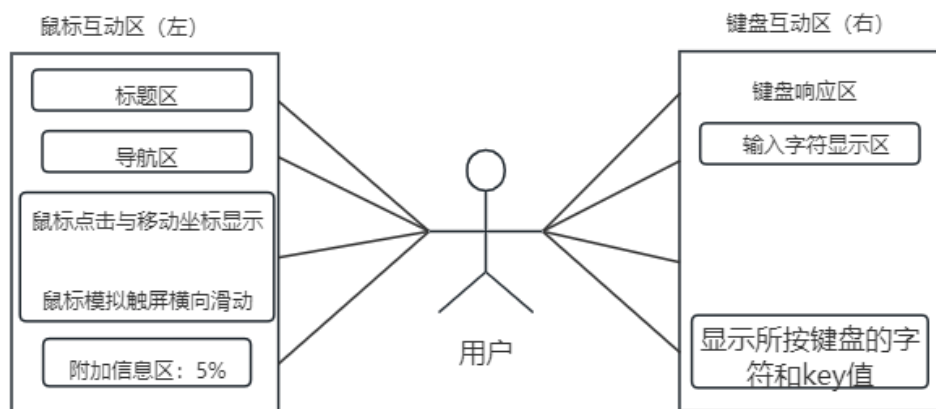


图 8-1 个性化键盘响应用例图

## 8.2 项目实现和编程

### (1) JavaScript 代码编写

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，如代码块 8-1 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)
});
} //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;
 }
 }
}

```

代码块 8-1

### 8.3 项目运行和测试

在测试过程中，我进行了一系列的随机按键操作，以观察并记录界面对于键盘输入的响应。我特别关注了两个关键的交互时刻：一是按下键盘时界面的即时反馈，二是松开键盘后界面的状态变化。这些测试结果分别在图 8-2 和图 8-3 中得到了展示。同时用户也可以通过扫描图 8-4 中的二维码，查看项目第六阶段的

的运行效果。

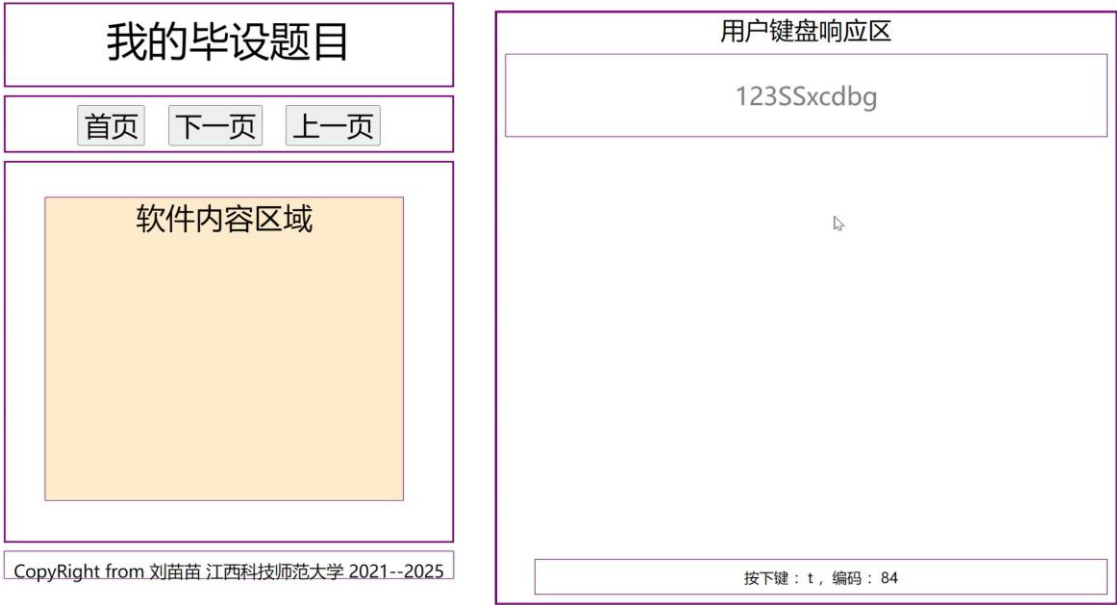


图 8-2 用户键盘响应区按下键盘运行效果图

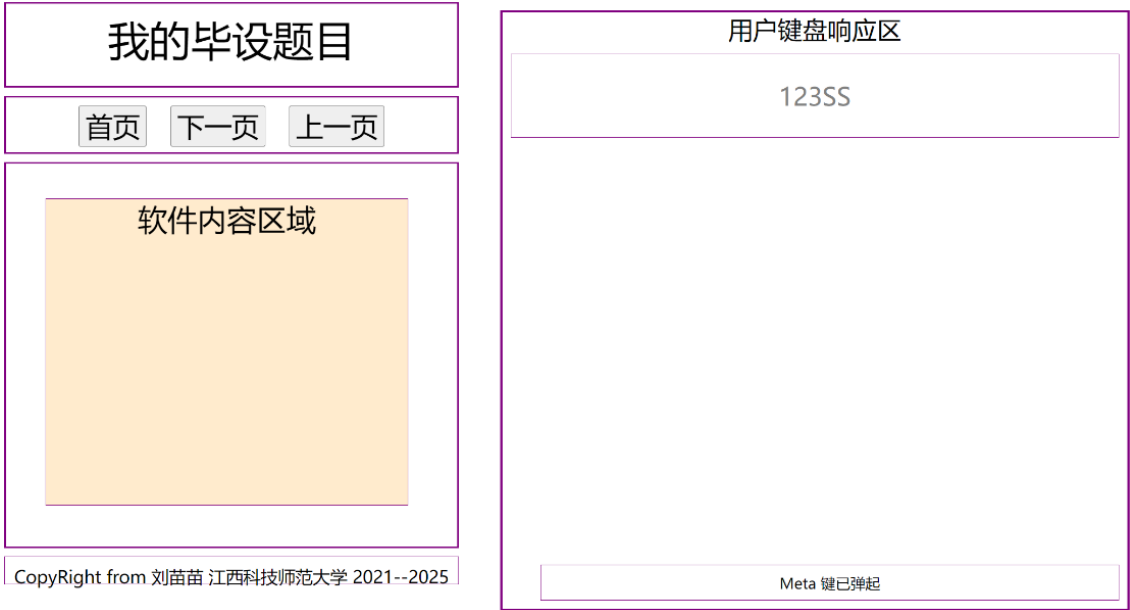


图 8-3 用户键盘响应区松开键盘运行效果图



图 8-4 六阶段移动端二维码

## 8.4 项目的代码提交和版本管理

(1) 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：实现了与键盘之间交互控制功能，成功将键盘输入的字符动态的显示在项目指定界面中

成功提交代码后，gitbash 的反馈如下图 8-5 所示：

```
86137@LAPTOP-ONNA3P69 MINGW64 /d/webUI (master)
$ git commit -m 项目第六版：实现了与键盘之间交互控制功能，成功将键盘输入的字符动
态的显示在项目界面中
[master dc5611b] 项目第六版：实现了与键盘之间交互控制功能，成功将键盘输入的字符
动态的显示在项目界面中
1 file changed, 259 insertions(+)
create mode 100644 1.6.html
```

图 8-5 六阶段代码提交

(2) 输入日志命令查看项目代码仓库的提交记录

```
$ git log
```

gitbash 反馈代码的仓库日志如下图 8-6 所示：

```
commit dc5611b0c8abd99ef32f14f8800a1ca0441bb7ae (HEAD -> master)
Author: 刘苗苗 <910519723@qq.com>
Date: Wed Jun 12 08:59:19 2024 +0800

 项目第六版：实现了与键盘之间交互控制功能，成功将键盘输入的字符动态的显示在项
目界面中
```

图 8-6 六阶段仓库日志

## 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

Bash 是一种强大的命令行工具，它允许用户通过脚本自动化各种任务。使用 Bash，你可以定义变量来存储数据，编写条件语句来做出决策，以及创建循环来重复执行命令。Bash 的管道功能可以将一个命令的输出直接作为另一个命令的输入，实现高效的数据流处理。

此外，Bash 支持输入输出重定向，让你能够控制数据的流向，比如将命令的输出保存到文件或从文件读取数据作为输入。通配符的使用使得文件搜索和操作更加灵活，而数组则允许你存储和操作多个值。Bash 还提供了函数功能，让你能够编写可重用的代码块，并且支持信号和陷阱，使得脚本能够响应系统信号。命令别名可以简化复杂的命令，而命令历史则让你能够快速回顾和执行之前的命令。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]。Bash 支持子 shell 的概念，允许在隔离的环境中执行命令，不影响主 shell 的状态。最后，Bash 提供了一些调试工具，帮助开发者在开发过程中跟踪和解决问题。

简而言之，Bash 是一个功能全面、易于使用的命令行工具，适用于从简单的文件操作到复杂的脚本自动化。

### 9.2 通过 gitHub 平台实现本项目的全球域名

使用 github 平台实现本项目的全球域名如下步骤如下：

（1）创建 GitHub 仓库：

登录你的 GitHub 账户。

点击页面右上角的 "+" 号，选择 "New repository" 创建一个新的仓库。

(2) 初始化本地仓库：

在本地项目文件夹中打开命令行或终端。

```
(1) 导航到项目目录
$ cd /d/webUI
(2) 执行 git init 初始化 Git 仓库:
$ git init
//这个命令会在当前目录下创建一个名为.git的隐藏文件夹，Git 将使用这个文件夹来存储所有的仓库数据和元数据。查看初始化结果：初始化后，终端通常会显示一些信息，告诉你 Git 仓库已经成功初始化。
(3) 配置用户信
$ git config user.name "刘苗苗"
$ git config user.email "910519723@qq.com"
```

## 9.3 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 *                 |
|  liumiaomiao20213665 | / lmm2021.github.io               |
|                                                                                                         | ✓ lmm2021.github.io is available. |
| <b>Create repository</b>                                                                                |                                   |

图 9-1 在 github 上创建一个空的仓库

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。将项目托入网站中托管，GitHub 会自动创建一个以 username.github.io/repositoryname 格式的子域名，其中 username 是你的 GitHub 用户名。通过访问 GitHub 自动生成的网站，能得到该网页的二维码。

## 9.4 设置本地仓库和远程代码仓库的链接

(1)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/liumiaomiao20213665/userName.github.io.git
$ git push -u origin main
```

代码块 9-1

(2)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：

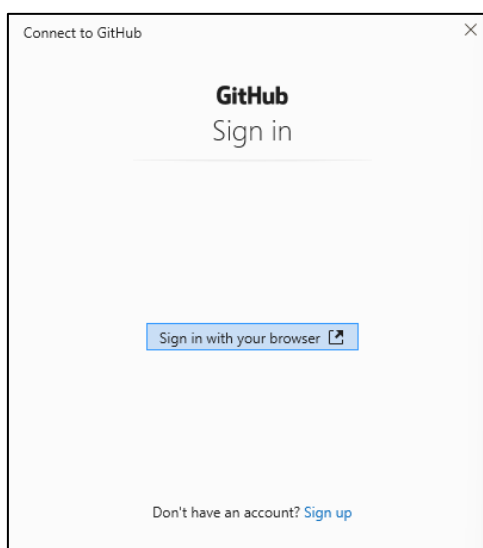


图 9-2 开发者授权

(3)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：



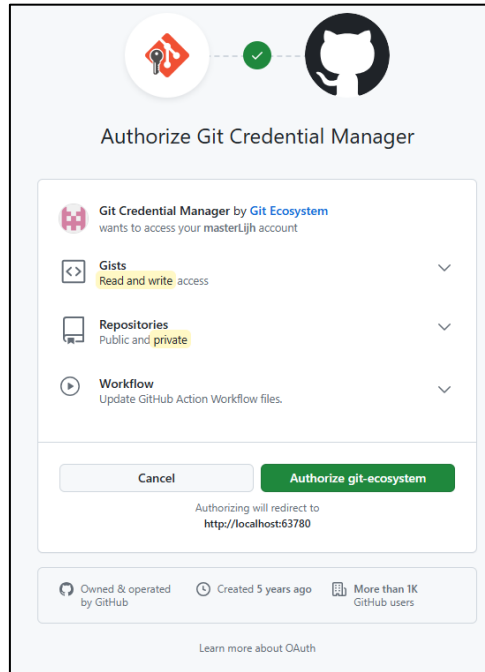


图 9-3 确认授权访问改动远程代码权限

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

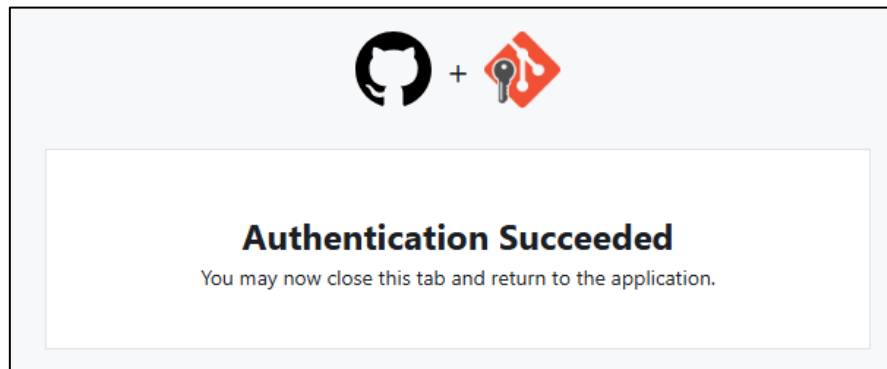


图 9-4 实现远程连接页面

(5)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。

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