

成绩:

# 江西科技师范大学

## 毕业设计 (论文)

题目(中文): 基于 Web 客户端技术个性化 UI 设计与编程

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

院系: 元宇宙产业学院

专业: 计算机科学与技术

学生姓名: 彭泱旸

学号: 20213624

指导教师: 李健宏

2024 年 6 月 17 日

# 目录

<b>1. 引言</b>	2
1.1 毕业设计任务分析	2
1.2 研学计划	2
1.3 研究方法	3
1.3.1 文献法	3
1.3.2 模型研究法	3
1.3.3 其他会误解的研究方法	4
<b>2. 技术总结与文献综述</b>	5
2.1 Web 平台和客户端技术概述	5
2.1.1 Web 平台	5
2.1.2 Web 编程	5
2.2 软件开发的过程管理	6
2.2.1 增量模型（ADIT）	6
2.2.2 瀑布模型（WDD）	7
2.3 有关技术介绍	8
2.3.1 Html	8
2.3.1 CSS	9
2.3.1 JavaScript	9
<b>3. “三段论”式的内容设计与实现</b>	9
3.1 分析与设计	9
3.1.1 需求分析	9
3.1.2 功能设计	10
3.2 代码实现	10
3.2.1 HTML 代码	10
3.2.2 CSS 代码	11
3.3 运行与测试	11
3.4 代码提交与版本控制	12
<b>4. 响应式设计与适应屏幕实现</b>	15
4.1 分析与设计	15
4.1.1 功能分析	15
4.1.2 功能设计	16
4.2 代码实现	16
4.2.1 HTML 代码	16
4.2.2 CSS 代码	17
4.2.3 JavaScript 代码	18
4.3 运行与测试	18
4.4 代码提交与版本控制	19
<b>5. 交互 UI 鼠标模型实现</b>	20
5.1 分析与设计	20
5.1.1 需求分析	20

5.1.2 功能设计.....	21
5.2 代码实现.....	21
5.2.1 HTML 代码.....	21
5.2.2 CSS 代码.....	22
5.2.3 JavaScript 代码 .....	23
5.3 运行与测试.....	24
5.4 代码提交与版本控制.....	25
<b>6. 探索 UI 拖动模拟触屏的操作模式 .....</b>	<b>26</b>
6.1 分析与设计.....	26
6.1.1 需求分析.....	26
6.1.2 功能设计.....	27
6.2 代码实现.....	27
6.2.1 HTML 代码.....	27
6.2.2 CSS 代码.....	28
6.2.3 JavaScript 代码 .....	30
6.3 运行与测试.....	32
6.4 代码提交与版本控制.....	33
<b>7. 通用 UI 设计为触屏和鼠标统一建模 .....</b>	<b>34</b>
7.1 分析与设计.....	34
7.1.1 需求分析.....	34
7.1.2 功能设计.....	35
7.2 代码实现.....	35
7.2.1 HTML 代码实现.....	35
7.2.2 CSS 代码.....	36
7.2.3 JavaScript 代码 .....	38
7.3 运行与测试.....	40
7.4 代码提交与版本控制.....	41
<b>8. 个性化 UI 监控键盘实现 .....</b>	<b>42</b>
8.1 分析与设计.....	42
8.1.1 需求分析.....	42
8.1.2 功能设计.....	43
8.2 代码实现.....	44
8.2.1 HTML 代码.....	44
8.2.2 CSS 代码.....	45
8.2.3 JavaScript 代码 .....	46
8.3 运行与测试.....	49
8.4 代码提交与版本控制.....	50
<b>9. 用 Git Bash 工具管理项目的代码仓库和 Http 服务器 .....</b>	<b>51</b>
9.1 跨世纪的经典 Bash 工具 .....	51
9.2 通过 Git Hub 平台实现本项目的全球域名。 .....	51
9.3 创建一个空的远程代码仓库.....	52
9.4 设置本地仓库和远程代码仓库的链接.....	52
<b>参考文献 .....</b>	<b>55</b>

# 基于 Web 客户端技术个性化 UI 设计与实现

**【摘要】：**近年来，以 HTML5 为核心的 Web 开发技术因其跨平台和开源的特性，在各个领域的应用软件开发中得到了广泛应用。本项目选择了 HTML5 作为 Web 客户端技术路线，对程序设计和软件开发进行了深入研究和实践。通过广泛阅读相关技术书籍、开发者论坛和文献，设计并开发了一个具有个性化用户界面（UI）的应用程序。在开发过程中，综合运用了 HTML 进行内容建模、CSS 进行 UI 设计、JavaScript 实现交互功能。除了直接使用 Web 客户端的底层 API 外，项目中的所有代码都是手工编写，没有使用任何第三方代码（包括框架和库）。本项目采用了响应式设计，能够智能适应移动互联网时代用户屏幕多样化的需求。同时，大量应用了面向对象的编程思想，例如构建了一个通用的指针模型，该模型通过一套代码实现了对鼠标和触屏的控制，确保了代码的高质量，这也是项目的一大特色。从工程管理的角度来看，本项目采用了增量式开发模式，通过逐步细化的方式进行了六次代码的增量式重构，包括分析（A）、设计（D）、实现（I）和测试（T），愉快地完成了项目的设计、开发和测试工作。在代码的开源和分享方面，本项目使用了 git 工具进行版本管理，在开发过程中进行了六次代码重构并进行了正式提交，另外在测试阶段还进行了两次代码修改提交。最终，利用 Git-Bash 工具将项目的代码仓库上传到了 GitHub，并利用 GitHub 提供的 HTTP 服务器，实现了 UI 应用在全球互联网的部署。用户可以通过多种方式访问程序，无论是通过网址直接访问还是扫描二维码，都能获得跨平台的高效互动体验。

**关键字：**Web 前端;JavaScript; GitHub; Git-Bash; Html;代码仓库管理。

## 1. 引言

### 1.1 毕业设计任务分析

毕业设计要求学生利用在本科阶段所学的计算机科学与技术知识，特别是程序设计和软件工程领域的理论和实践技能，来规划自己感兴趣的技术发展路径。学生需要结合自己的研究问题，明确软件需求，然后按照软件工程的传统流程，有条不紊地进行问题分析、建模、设计、实施和测试调试等工作。

计算机科学与技术专业的毕业设计是学生本科教育的综合性实践环节，它具有深远的教育意义和实际应用价值。毕业设计使学生能够将所学知识与技能综合运用，通过解决实际问题来巩固和深化对专业知识的理解和掌握。这一过程不仅锻炼了学生的编程能力、系统设计能力、项目管理能力和问题解决能力，而且培养了他们的专业素养、创新思维和职业精神。鼓励学生探索新技术、新方法，激发他们的创新思维，增强自主研究和创新的能力。它要求学生遵循软件工程的标准和规范，培养了学生的专业素养和职业精神。同时，毕业设计通常需要团队合作完成，这有助于学生学习如何在团队中发挥作用，提升团队协作和沟通能力。此外，毕业设计模拟了真实工作环境中的项目开发流程，为学生之后进入职场，打下牢固的基础，帮助学生更好地适应未来的工作。对于有意向继续深造的学生，毕业设计是进行学术研究的初步尝试，可以为他们未来的研究生学习或学术生涯奠定基础。通过撰写开发文档和毕业论文，学生培养了文档撰写和表达能力，这对于任何技术职位都是非常重要的技能。毕业设计过程中的自我学习和探索，有助于学生树立终身学习的观念，为未来的技术发展和个人成长打下坚实基础。毕业设计往往围绕解决现实问题，通过项目实施，学生能够为社会带来积极的影响和价值。总而言之，计算机科学与技术专业的毕业设计是学生本科学习成果的一次全面检验，也是对学生未来职业生涯的重要准备，具有重要的教育意义和实际应用价值。

### 1.2 研学计划

作为计算机科学技术专业的本科生，在学业即将结束时，设计并开发一个体现本专业特色的毕业作品是至关重要的。这样的作品不仅能够帮助我回顾和总结所学的知识体系，而且能够凸显我对课程核心内容的掌握，展示我的真实

技能。

在我的毕业设计中，我将运用一系列核心课程的理论知识，包括面向对象的程序设计、数据结构与算法、操作系统原理以及软件工程等。以往，这些课程的理论知识对我来说似乎有些抽象，主要因为它们偏重于理论而缺乏实践。在与导师的交流中，我达成共识，认为将这些核心课程的关键知识点应用到实践中，将极大地促进我在理解深度和技术层面上的专业成长。因此，我将毕业设计视为一个机会，它不仅是对大学理论学习的一次综合实践和回顾，也要求我学习并掌握一些当前流行的新技术。在形成了对计算机软硬件体系全面而深入的理解后，我将撰写毕业论文，这正是毕业论文的核心所在。对我这些即将成为国家现代化建设者的计算机专业学生来说，深刻理解计算机系统至关重要。这将使我与其他专业的毕业生区别开来，成为我独特的优势。在他人眼中，作为计算机专业的学生，我对计算机系统的理解不应停留在表面，而应力求深入到计算机的本质。同样，对任何技术的理解，我都应努力触及其实质和基本原理。

### 1.3 研究方法

#### 1.3.1 文献法

第一种文献法。文献法是一种研究方法，它主要涉及搜集、鉴别和整理文献资料，并通过研究这些文献形成对事实的科学认识。这种方法既古老又充满生命力。文献法的运用是一个继承与批判的过程，其根本目的在于通过系统的检索、收集、鉴别以及研究与运用，实现对某一时代或社会教育现象的某些特点进行描述和评论，分析其形成的客观原因，并可能形成新观点、创造出新理论。文献法的信息来源可以非常广泛，包括书籍、文章、报刊杂志、统计公报、年鉴、信息公报、政策法规、学术论文、调查报告等。在进行文献研究时，研究者需要对文献的真实性和可用性进行检验，确保研究的系统性和可靠性。高质量的文献资料应具备真实、新颖、全面和准确的特点。文献研究法的一般过程包括提出课题或假设、研究设计、搜集文献、整理文献和进行文献综述。文献法的提出课题或假设是指依据现有的理论、事实和需要，对有关文献进行分析整理或重新归类研究的构思。

#### 1.3.2 模型研究法

模型研究法。模型研究法，也称为模拟法，是一种通过创建和分析与原型相似的模型来间接研究原型的科学方法。这种方法适用于那些无法或难以直接进行研究的对象系统。研究者首先收集有关原型的信息，如测试数据、实验数据和调查资料，了解其特性、结构和功能。然后，在综合分析这些事实资料的基础上，通过创造性思维和逻辑推理建立模型。接着，对模型进行观察、实验和研究分析。最后，将结论外推到原型。模型研究法可以采用多种方法，包括物理模拟、数学模拟、功能模拟和智能模拟等，具体方法会根据对象系统的特性、复杂程度和用途而有所不同。模型分析需要采用一系列的方法和技术，如系统分析法、系统综合法、结构分析法、要素分析法、功能分析法、优化分析法以及计算机运算技术和有关逻辑演绎方法等。模型研究法在理论研究和工程研究中有着广泛的应用，已成为不可或缺的工具。它通过简化和理想化的形式再现原型的各种复杂结构、功能和联系，是连接理论和应用的桥梁。此外，模型实际上是假设的一种特殊形式，它需要在实践中接受检验，并在实践中不断扩展、补充和修正。

模型研究法的基本步骤包括：搜集有关原型的信息、建立模型、对模型进行观察和实验、分析研究模型，以及将结论外推到原型。建模时应使模型满足真实性、简明性（便于进行数学处理）、完整性（具备模型的基本要素）和规范化等要求。

### 1.3.3 其他会误解的研究方法

案例研究法是一种以深入分析特定实例为核心的研究方法，它在管理学等专业领域中非常适用，比如产品经理可以通过深入研究一个软件产品来应用这种方法。然而，这种方法对于以技术为主的计算机软件开发领域可能不太合适。

实验法通常与医学研究相关，它通过比较不同实验条件下的数据来探究因果关系，这与计算机科学中的程序和代码开发有所不同，后者更多是对现实世界现象的模拟和抽象。

至于实践法，实际上它并不是一种独立的研究方法。实践是将理论应用于现实世界的过程，是检验和验证理论有效性的手段。在计算机科学领域，所有的理论和技术最终都需要通过实践来检验，无论是代码编写还是系统构建，都需要在实际应用中证明其有效性和效率。对于计算科学的基础理论研究，包括

理论和数学研究，它们可能不直接涉及实践操作，但最终目的仍然是为了更好地理解和解释现实世界，为实践提供理论基础和指导。

## 2. 技术总结与文献综述

### 2.1 Web 平台和客户端技术概述

#### 2.1.1 Web 平台

万维网的创始人蒂姆·伯纳斯-李在确立了网络的基础技术之后，创建了 W3C 组织。W3C 在 2010 年之后推出了 HTML5 这一全球性标准，并与欧洲的 ECMA 组织共同维护 ECMAScript 标准，这两项标准极大地推动了全球开发平台的标准化。至今，科学家和网络行业的专家们仍在持续努力，以进一步优化和提升这一宏伟目标的实现<sup>[1]</sup>。

Web 平台，作为现代互联网的核心组成部分，提供了一个多功能、可访问的环境，让用户能够通过浏览器直接访问和使用各种在线服务和应用程序。这些平台的构建依赖于一系列开放的网络标准和协议，确保了跨平台的兼容性和可扩展性。Web 平台的架构通常采用客户端-服务器模型，其中客户端负责展示用户界面和处理用户输入，服务器端则负责处理业务逻辑、数据存储和提供 API 服务。这种分离的架构不仅提高了应用的可维护性，也使得 Web 平台能够灵活地应对不同的用户需求和设备类型。

随着技术的发展，Web 平台已经从简单的静态网页演变为支持复杂交互和动态内容的动态应用。现代 Web 平台利用了 Ajax、WebSocket 等技术实现与服务器的实时通信，为用户提供了接近本地应用程序的体验。同时，Web Assembly 等新兴技术进一步扩展了 Web 平台的能力，允许在浏览器中运行接近原生性能的代码。

#### 2.1.2 Web 编程

Web 编程是一个综合性的技术领域，它涵盖了从前端用户界面的设计到后端服务器逻辑的实现。前端开发使用 HTML、CSS 和 JavaScript 等技术，通过框架动态、响应式的用户界面<sup>[2]</sup>。而后端开发则涉及使用不同编程语言，结合框架，处理数据存储、业务逻辑和 API 的构建。在整个开发过程中，版本控制工具如 Git 发挥着至关重要的作用，它帮助团队成员高效协作，同时保持代码的版



本历史。响应式设计确保了 Web 应用程序能够在各种设备和屏幕尺寸上提供一致的体验。随着 Web 技术的不断发展,如 Web Assembly 和 PWA (渐进式 Web 应用程序),Web 编程的能力和范围也在不断扩展。不断学习新的工具、技术和最佳实践,以适应这个快速变化的领域。通过整合这些技术和实践,Web 编程能够创造出功能丰富、用户友好且安全的在线体验<sup>[3]</sup>。

## 2.2 软件开发的过程管理

### 2.2.1 增量模型 (ADIT)

增量式迭代开发模式是一种将软件开发过程分解为多个小的、可管理的增量或迭代的软件开发方法。它将整个开发过程划分为多个阶段,每个阶段都是一个完整的开发周期,包括需求分析、设计、实现、测试和部署等阶段。每个增量都是一个完整的、可交付的产品或功能,可以独立运行或与之前的增量集成。特点在于它的灵活性和适应性,允许开发团队在每个迭代周期中,根据用户反馈和市场变化快速调整方向。它强调持续集成,确保新开发的功能能够无缝集成到现有系统中,从而保持项目的稳定性和一致性。此外,增量式迭代开发模式还注重质量保证,每个增量都经过严格的测试,以确保最终产品的质量。通过这种方式,团队可以逐步交付价值,同时不断学习和改进,最终实现高质量的软件产品开发。优势在于其高度的灵活性和适应性,能够应对快速变化的需求和市场条件。通过将项目分解为一系列小的、可交付的增量,它允许开发团队持续地集成新功能,同时确保系统的稳定性和可维护性。这种模式下的快速迭代可以加速产品上市时间,同时提供频繁的用户反馈循环,使得产品能够更好地满足用户的实际需求。风险管理也更为有效,因为问题可以在早期被发现和解决,减少了后期大规模重构的可能性。此外,客户参与度的提高确保了开发的产品与用户期望的一致性,增强了最终用户的满意度。总的来说,增量式迭代开发模式通过持续的改进和交付,为软件开发提供了一种高效、响应迅速且用户中心化的方法<sup>[4]</sup>。

作为一项本科专业学生的毕业设计软件作品,本项目在复杂性上远超一般单一用途的程序。手写代码量庞大,远非简单程序可比,从问题分析到初步编码,整个过程不可能在短短几天内完成。因此,本项目可视为一个系统工程,需要运用软件工程的管理方法来规范开发过程。

在软件工程的开发模式中，有两大经典模型可供选择：瀑布模型（The Waterfall Model）和增量式迭代模型（The Incremental Model）。无论采用哪种模型，软件开发都必须经历分析（Analysis）、设计（Design）、实施（Implementation）和测试（Test）这四个基本阶段。

在当今开放的软件开发环境中，开发者在开发过程中总是在不断地优化设计、重构代码，以持续提升程序的功能和代码质量。因此，本项目选择了增量式迭代模型作为开发模式。如下图 3-1 增量式迭代模型所示。

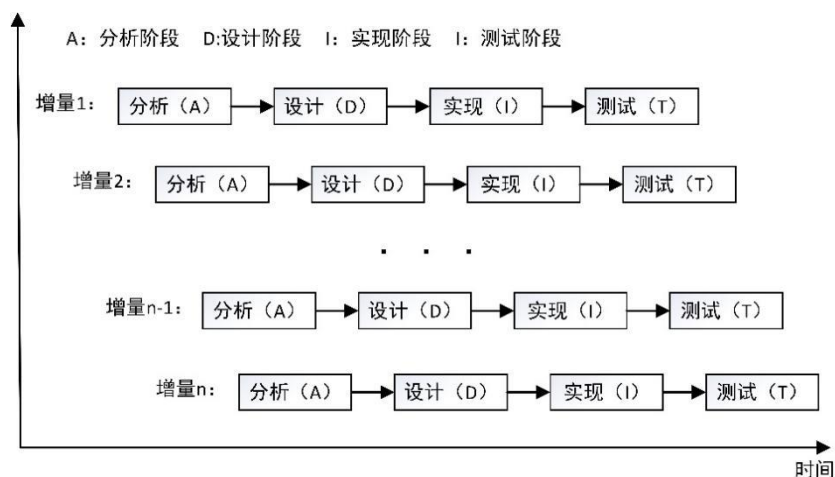


图 2-1 增量式迭代模型

增量式迭代模型允许开发者在每个迭代中逐步构建和完善产品，每个迭代都包括需求分析、设计、实现和测试等环节。这种模式更加灵活，能够适应需求的变化，允许开发者在开发过程中不断学习和改进，最终实现高质量的软件开发。通过这种方式，本项目能够更好地应对开发过程中的不确定性，提高开发效率和产品质量<sup>[5]</sup>。

## 2.2.2 瀑布模型（WDD）

瀑布模型（Waterfall Model）是一种传统的软件开发方法论，它将软件开发过程划分为一系列线性、顺序的阶段，每个阶段的输出是下一个阶段的输入。这种模型的流程从需求分析开始，经过系统设计、实现（编码）、测试、部署，最终到达维护阶段。瀑布模型要求团队成员之间有极高的协同性，每个阶段都必须在前一阶段完美完成后才能开始。然而，对于大多数普通开发者，尤其是小微团队的开发者来说，这几乎是不可能实现的。他们往往需要同时承担多个角色，很难一次性完美完成任何一个阶段。例如，在实施阶段，开发者可能会发现设计上存在的问题，这时就需要在下一轮迭代中对设计进行改进。

在需求分析阶段，开发者与客户紧密合作，明确软件需要满足的功能和性能要求。随后的系统设计阶段，开发者创建软件架构和详细设计文档，为编码阶段奠定基础。实现阶段是开发者根据设计文档编写代码，构建软件的各个组件。接下来，测试阶段确保软件满足需求并且没有缺陷。一旦软件通过测试，它将被部署到生产环境中供用户使用。最后，在维护阶段，开发者根据用户反馈进行必要的更新和修复。瀑布模型的优势在于其结构化和可预测性，使得项目管理相对简单。如图 2-2 瀑布式开发模型所示<sup>[6]</sup>。

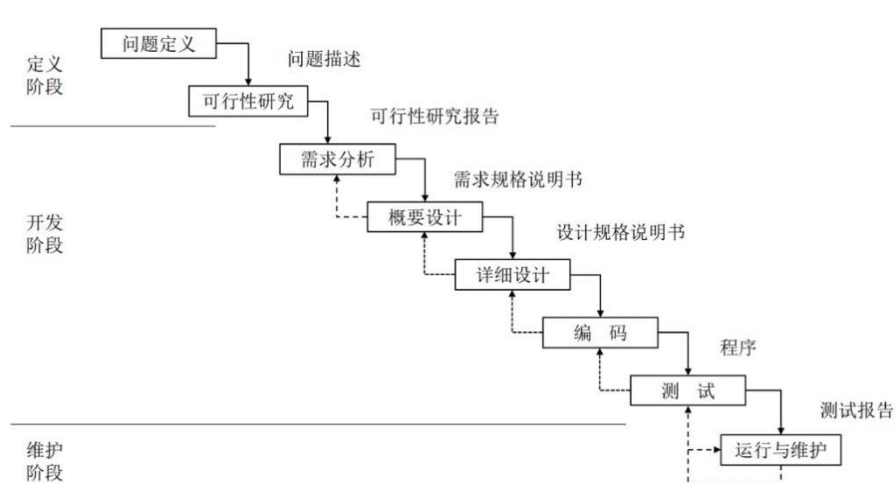


图 2-2 瀑布式开发模型

然而，它的主要缺点是缺乏灵活性，需求变更在项目后期可能非常昂贵和耗时。此外，用户直到开发周期的后期才能看到软件的实际运行效果，这可能导致最终产品与用户期望不符。随着软件开发实践的演进，更灵活的敏捷开发方法逐渐受到青睐，它们允许更频繁的反馈和迭代，以更好地适应需求变化。尽管如此，对于需求明确且变化不大的项目，瀑布模型仍然是一种有效的开发方法<sup>[7]</sup>。

## 2.3 有关技术介绍

### 2.3.1 Html

HTML 是一种用于构建网页和网络应用的标记语言，它通过标签定义网页结构，支持创建链接、嵌入多媒体内容、设计表单以及与 CSS 和 JavaScript 结合，实现网页的样式设计和交互功能。HTML5 作为其最新标准，引入了更多语义化标签和 Web API，支持响应式设计，增强了跨平台和设备兼容性，是现代 Web 开发不可或缺的基础技术。

### 2.3.1 CSS

CSS 是一种用于控制网页外观和布局的样式表语言，它通过分离内容与表现层，使用选择器和继承机制来应用样式，支持盒模型、响应式设计、动画效果和跨浏览器兼容性，为网页设计提供了强大的视觉表现力和灵活性。

### 2.3.1 JavaScript

JavaScript 是一种广泛使用的客户端脚本语言，它通过动态交互、DOM 操作、事件驱动、异步编程和丰富的 Web API，为网页带来交互性和动态效果，同时支持跨平台和全栈开发，拥有强大的框架和库生态，以及活跃的社区支持，是现代 Web 开发的核心语言之一。

## 3. “三段论”式的内容设计与实现

### 3.1 分析与设计

#### 3.1.1 需求分析

本项目的界面设计遵循了用户友好的“三段论”原则，首先映入眼帘的是醒目的标题性信息，无论是通过 logo 展示还是文字标题，都能迅速吸引用户的注意力。紧接着，用户的视线会转移到内容区，这是整个 UI 设计的核心，本项目坚信“内容为王”的理念，确保高质量的内容是吸引和留住用户的关键。最后，页面底部提供了一些附加信息，包括用户可能关心的细节和变化，以满足用户对信息完整性的需求<sup>[8]</sup>。

在这种设计思路下，本项目的 UI 布局清晰合理，既突出了重点内容，又兼顾了细节的展示。这种“三段论”的设计方法，不仅符合用户的视觉习惯，也有助于提升用户体验。通过精心设计每个部分，致力于打造一个既美观又实用的 UI 界面，让用户在使用过程中感到舒适和愉悦。在保持整体风格统一的同时，也为未来可能的功能扩展留出了空间。通过模块化的设计方法，本项目可以灵活地添加或调整功能模块，以满足不断变化的业务需求。如下图 3-1 第一次增量迭代用例图所示。

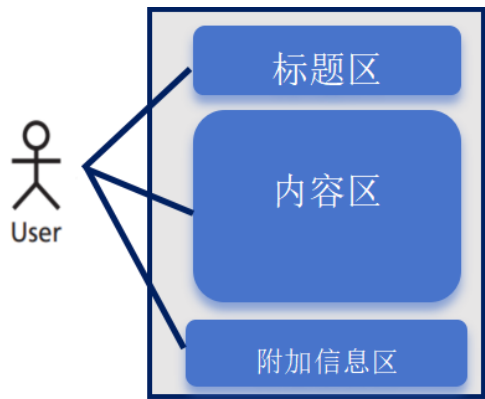
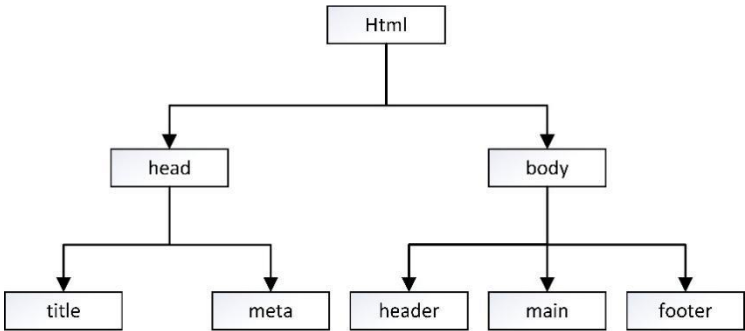


图 3-1 第一次增量迭代用例图

3.1.2 功能设计

首先，使用 HTML5 来定义页面的布局框架。在这个框架中，需要添加 header、main 和 footer 这三个关键的布局元素。header 元素用于展示网站的标题，main 元素承载核心的艺术作品内容，而 footer 则用于展示版权信息和其他个人信息。随后，设计编写 CSS 对这些元素进行了细致的样式设计，以确保网页的视觉效果与如图 4.2 项目 Dom 树所示。



图表 3-2 第一次增量迭代 Dom 树

在 Dom 树里面可以清晰地了解项目结构。

3.2 代码实现

首先，我通过 HTML5 构建了网页的基础框架，定义了页面的各个主要部分，如页眉（header）、主体（main）和页脚（footer），分别用于展示页面标题、主要内容以及版权声明等信息。随后，我运用 CSS 对这些元素进行样式设计。具体的实现方法可以参考下面代码。

3.2.1 HTML 代码

```
1. <body>
2.   <header>
3.     作品展示
4.   </header>
```

```
5.     <main>
6.         内容展示区
7.     </main>
8.     <footer>
9.         © 彭泱旸 江西科技师范大学 2024-2025
10.    </footer>
11. </body>
```

### 3.2.2 CSS 代码

```
1. * {
2.     margin: px;
3.     text-align: center;
4.     align-content: center;
5.     justify-content: center;
6.     font-size: px;
7.     font-family: 'KaiTi', '楷体', 'SimKai', 'STKaiti', serif;
8. }
9. header {
10.    border: px solid black;
11.    height: px;
12. }
13. main {
14.    border: px solid black;
15.    height: px;
16. }
17. footer {
18.    border: px solid black;
19.    height: px;
20. }
21. a {
22.    display: inline-block;
23.    padding: px;
24.    color: black;
25.    box-shadow: px px px px rgba(, , , );
26.    background-color: #ECFC;
27.    background-image: linear-gradient(deg, #ECFC %, #ECFC %);
28.    text-decoration: none;
29. }
```

## 3.3 运行与测试

在浏览器里面打开本地的文件，运行代码查看效果图如下图 3-3 运行效果图。

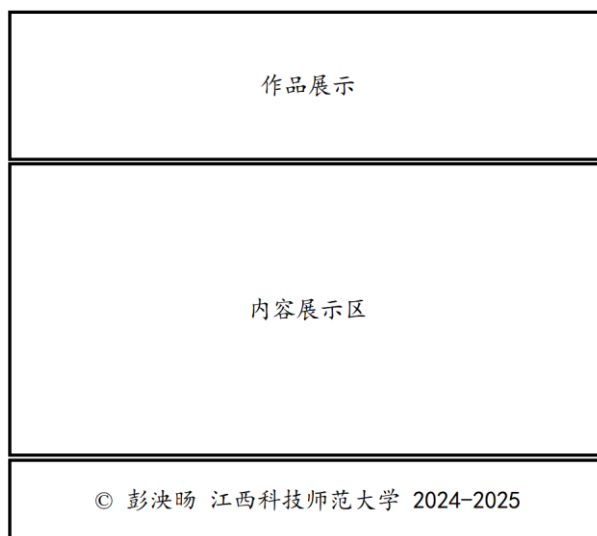


图 3-3 运行效果图

手机端可以扫描下图 3-4 阶段 1 增量迭代二维码，访问“三段论”式的内容设计与实现文件。

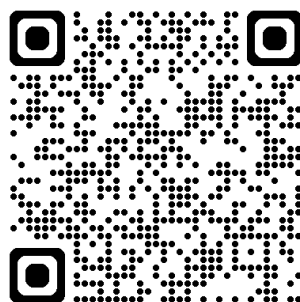


图 3-4 阶段 1 增量迭代二维码

### 3.4 代码提交与版本控制

先将代码用 Git Bash 对项目代码进行版本控制，具体过程如下面代码块所示。

```
1. cd /WebUIProject
2. # cd 后面输入要进入的项目文件夹
3. git init
4. # init 初始化本地代码仓库，用于代码版本控制
5. git status
6. # status 查看仓库代码状态
7. git add <phase1.html>
8. # add 后面输入要跟踪的文件名字
9. git commit -m "commit message"
10. # commit 后面输入本次提交代码的备注信息
11. git log
```

## 12. # log 查看仓库代码的提交信息

完成代码的版本控制后，可以查看本次提交代码的状况，如下图 3-5 提交代码图所示。

```
commit 7c0a88a2147fa14044ee76595f176b3af6def962 (origin/master)
Author: 20213624彭泱昀@科师大 <353365869@qq.com>
Date: Wed Jun 12 15:23:10 2024 +0800

    第一次增量迭代—完成了软件的开发环境的搭建，完成了系统的设计概要。本次代码提交index.html、Css.css、Js.js、Phase/phase1,htm等文件。把整个软件分成了上中下三个部分，header里面放标题用于页面的头部的展示，main里面放软件的内容，就是软件的主体内容与页面的展示，footer里面放软件的动态反馈，也就是javascript的代码。
```

图 3-5 提交代码图

在 4-5 里面 Commit Hash 这是一个独特的 40 位十六进制数，用于标识这次提交。它是这次更改的永久标识符；Author 显示了提交者的用户名、姓名、学校或机构以及电子邮件地址；Date 这是提交的时间和日期，根据协调世界时加上 8 小时的时区偏移。用命令行将代码推送到远程 GitHub 网站上。

```
git remote add origin https://github.com/pyy13635987691/ksdPyy.io
# 添加远程仓库 remote add 命令将 GitHub 仓库 URL 添加为远程仓库。
git remote -v
# 添加完成后，使用 remote -v 来验证远程仓库是否成功添加
git push --set-upstream origin master
# 设置远程仓库的 master 分支作为当前本地 master 分支的上游分支
git push
```

添加成功后的效果如下图 3-6 检查 GitHub 连接所示。

```
R9000X 2021@Lenovo MINGW64 /d/WebUIProject (master)
$ git remote -v
origin https://github.com/pyy13635987691/ksdPyy.io.git (fetch)
origin https://github.com/pyy13635987691/ksdPyy.io.git (push)
```

图 3-6 检查 GitHub 连接

使用 Git 设置远程仓库的 master 分支作为当前本地 master 分支的上游分支，这样以后就可以直接使用 git push 或 git pull 而无需指定分支，简化了后续的操作，提高了开发效率。一旦设置了上游分支，以后您再想要推送当前分支的时候，只需简单地执行 Git 会自动将您的更改推送到对应的上游分支。如下图 4-7 推送结果所示。



```
R9000X 2021@Lenovo MINGW64 /d/WebUIProjecct (master)
$ git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 10.43 KiB | 3.48 MiB/s, done.
Total 11 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/pyy13635987691/articleGithub/pull/new/master
remote:
To https://github.com/pyy13635987691/articleGithub
 * [new branch]      master -> master
```

图 4-7 推送结果

最后用命令行将代码推送到个人的 GitHub 项目仓库中，GitHub 会给个人生成的项目仓库生成 Http 访问路径，以实现全球用户访问本项目。具体过程如下面代码块所示。但是出现问题

```
R9000X 2021@Lenovo MINGW64 /d/WebUIProject (master)
$ git push --set-upstream origin master:main
To https://github.com/pyy13635987691/ksdPyy.io.git
 ! [rejected]        master -> main (fetch first)
error: failed to push some refs to 'https://github.com/pyy13635987691/ksdPyy.io.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

这是因为远程仓库与本地仓库的代码冲突了，通常为远程分支有新更改，远程的 `main` 分支上自上次拉取以来有新的提交；本地分支落后，本地 `master` 分支没有包含远程 `main` 分支上的所有更改；需要合并，需要先将远程分支的更改合并到本地，然后再尝试推送。并进行手动解决。解决过程代码如下面代码块所示。

```
git push --force origin master:main
# 使用以下命令，将本地的 master 分支强制推送到远程的 main 分支会将覆盖远程 main 分支上的所有历史。
git push --set-upstream origin master:main
# 将本地的分支推送到远程仓库的 main 分支，并设置远程 main 分支作为本地 master 分支的上游分支
```

执行结果如下图所示图表 1，推送完成后进入 GitHub 网页查看效果如下图所示图 2。

```
R9000X 2021@Lenovo MINGW64 /d/WebUIProject (master)
$ git push --force origin master:main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/pyy13635987691/ksdPyy.io.git
 + 6d2f0d0...7c0a88a master -> main (forced update)
R9000X 2021@Lenovo MINGW64 /d/WebUIProject (master)
$ git push --set-upstream origin master:main
Everything up-to-date
Branch 'master' set up to track remote branch 'main' from 'origin'.
```

图表 3



图表 4

## 4. 响应式设计 with 适应屏幕实现

### 4.1 分析与设计

#### 4.1.1 功能分析

响应式设计（Responsive Design）是一种网络设计方法，这种设计的目标是确保网站在所有设备上都能提供良好的用户体验，无论这些设备的屏幕尺寸、分辨率或方向如何。计算机使用的显示硬件种类繁多，显示设备的尺寸和分辨率取决于成本。设计师们并没有为每种显示设备设计不同版本的网页，而是选择让网页提供通用的布局指导，并允许浏览器决定如何在特定计算机上显示页面。因此，网页不会提供太多细节。允许浏览器选择显示细节有一个有趣的后果：当通过两个不同的浏览器或在两个具有不同硬件的计算机上查看时，网页的外观可能会有所不同。如果一个屏幕比另一个屏幕宽，可以显示的文本长度或图像大小就会有所不同。关键点是：网页提供了关于期望展示的一般指导；浏览器在选择显示页面的细节时会有所差异。因此，同一个网页在两台不同的计算机或不同浏览器上显示时可能会略有不同。

本次更新迭代代码要实现在不同用户的不同设备上正确运行，给用户最佳体验，因此要完成响应式设计并且适应屏幕的自动调节。如下图 4-1 第二次迭代项目用例图所示。

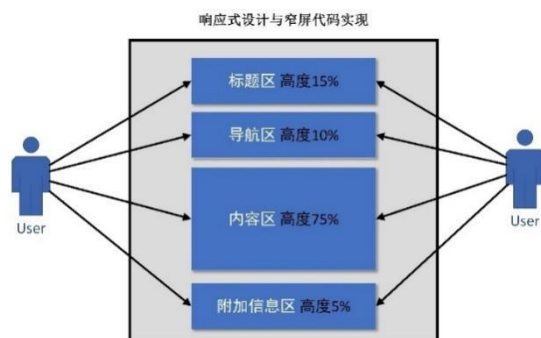


图 4-1 第二次增量迭代项目用例图

#### 4.1.2 功能设计

首先，使用 HTML5 来定义页面的布局框架。在前面的第一次迭代中分别在 header 中添加了 p 标签。新增了 nav 里面添加了 button，用于放置导航按钮。Main 里面添加了 p 标签用于内容显示。如图 4-2 项目 Dom 树所示。

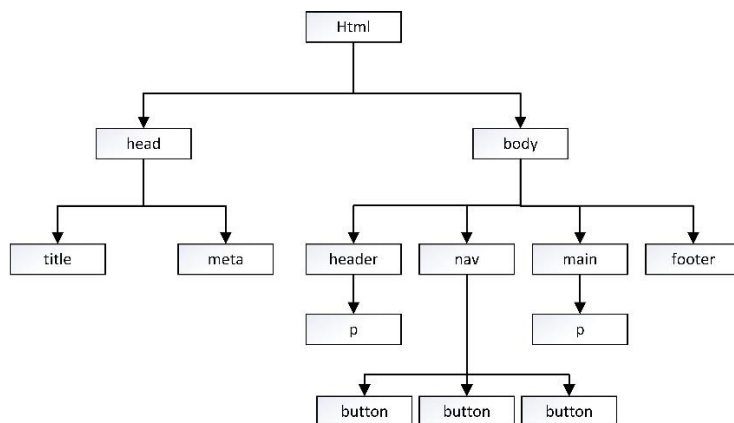


图 4-2 项目 Dom 树

### 4.2 代码实现

我通过 HTML5 构建了网页的基础框架，定义了页面的各个主要部分，如页眉（header）、主体（main）和页脚（footer），分别用于展示页面标题、主要内容以及版权声明等信息。我运用 CSS 对这些元素进行样式设计。具体的实现方法可以参考下面代码。

#### 4.2.1 HTML 代码

```
<body>
  <header>
    <p id="head">
      作品展示
    </p>
  </header>
  <nav>
    <button title="功能未实现">导航一</button>
```

```

<button title="功能未实现">导航二</button>
<button title="功能未实现">导航三</button>
</nav>
<main id='main'>
  <p class="Content">
    内容展示区
  </p>
</main>
</body>
<footer>
  © 彭泱旻江西科技师范大学 2024-2025
</footer>

```

#### 4.2.2 CSS 代码

```

*{
  margin:2px;
  text-align:center;
  align-content:center;
  justify-content:center;
  font-family:'KaiTi','楷体','SimKai','STKaiti',serif;
  box-sizing:border-box;
  -webkit-user-select:none;
  -moz-user-select:none;
  -ms-user-select:none;
  user-select:none;
}
header{
  background-color:#74EBD5;
  background-image:linear-gradient(90deg,#74EBD50%,#9FACE6100%);
  border:2pxsolidblack;
  height:15%;
  font-size:1.66em;
}
#head{
  font-size:1.3em;
}
main{
  background-color:rgb(247,247,247);
  border:2pxsolidblack;
  height:70%;
  font-size:1.2em;
  display:flex;
}
.Content{
  font-size:1.1em;
}

```

```

width: 75%;
background-color: #FFDEE9;
background-image: linear-gradient(0deg, #FFDEE9 0%, #B5FFFC 100%);
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}
nav{
  background-color: rgb(247, 247, 247);
  border: 2px solid black;
  height: 10%;
}
navbutton{
  font-size: 0.8em;
  padding: 13px 13px;
  box-shadow: 2px 2px 5px 0px rgba(0, 0, 0, 0.3);
  background-color: #8EC5FC;
  background-image: linear-gradient(62deg, #8EC5FC 0%, #E0C3FC 100%);
  border-radius: 8px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}
footer{
  background-color: #FAD961;
  background-image: linear-gradient(90deg, #FAD961 0%, #F76B1C 100%);
  border: 2px solid black;
  height: 5%;
}

```

#### 4.2.3 JavaScript 代码

```

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 + "px";
document.body.style.height = UI.appHeight - 1 * baseFont + "px";

```

### 4.3 运行与测试

这一小节展示在不同的设备上面运行项目的效果图，这是增量迭代的第二版，分别有 iPhone SE、Surface Duo 在这两种设备上面运行本项目，实际效果

与预期效果别无二致。效果图如下图 4-3 iPhone SE 与 Samsun Galaxy S8 的效果对比。



图 4-3 不同设备运行效果对比图

分析图 4-3 不同设备运行效果对比图可知，代码运行正常，实际效果与预期效果一致。代码可以根据用户的设备屏幕大小，来自动调整最佳页面效果，展示给不同设备用户。

手机端可以扫描下图 4-4 阶段 2 增量迭代二维码，访问“三段论”式的内容设计与实现文件。



图 4-4 阶段 2 增量迭代二维码

经过一系列细致的开发和测试工作，我最终实现了与前期分析和设计阶段所规划的一致运行效果。软件的每个功能模块都严格遵循了既定的设计原则，确保了用户界面的直观性、交互的流畅性以及性能的高效性。

#### 4.4 代码提交与版本控制

利用 3.4 里面的过程，将代码用 Git Bash 对项目代码进行版本控制如下图 4-5 提交代码图所示。

commit 9502a74368771ff22c56c914e61221e915ea4c1e  
 Author: 20213624彭泱旻@科师大 <353365869@qq.com>  
 Date: Wed Jun 12 16:06:58 2024 +0800

第二次增量迭代—本次提交了'移动互联时代的响应式设计(窄屏)'代码,本次代码运行了应用程序的响应式设计:1.声明了一个全局UI对象,用来存放需要使用的属性,2,为四个区域设置了占整个用户界面的比例(以高度为准)。3,用Javascript获取了应用屏幕的宽度和高度,并依据需要来进行计算基础字体的大小。4,在不同的区域根据内容的重要程度设定了该区域字体的相对大小。5.改变了颜色和对比色,对组区域的背景图做了优化外观的设计。6.总之,初步为用户的应用程序做了响应式设计并部署了代码。

图 4-5 提交代码图

将本地代码推送至远端,也就是 GitHub 网址中的代码仓库,第二迭代更新推送过程如下图 4-6 所示推送结果如下图 4-7 远端 GitHub 代码仓库结构所示。

```
R9000X 2021@Lenovo MINGW64 /d/WebUIProject (master)
$ git push origin HEAD:main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 2.22 KiB | 1.11 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/pyy13635987691/ksdPyy.io.git
7c0a88a..9502a74 HEAD -> main
```

图 4-6 第二迭代更新推送过程



Name	Last commit message	Last commit date
..		
phase1.html	第一次增量迭代——完成了软件的开发环境...	2 hours ago
phase2.html	第二次增量迭代——本次提交了移动互联...	1 hour ago

图 4-7 远端 GitHub 代码仓库结构

## 5. 交互 UI 鼠标模型实现

### 5.1 分析与设计

#### 5.1.1 需求分析

在开发交互式用户界面的过程中,鼠标发挥着关键作用。因此,我深入研究了用户界面设计中的鼠标行为模型,特别关注了鼠标坐标在计算机界面上的表现。我的目标是实现一项功能:当用户在屏幕上的特定区域内点击鼠标时,能够展示出该点击事件的鼠标坐标。

第三次增量迭代在第二次增量迭代的基础上,添加了鼠标的监控模型。如下图 5-1 第三次增量迭代用例图所示。

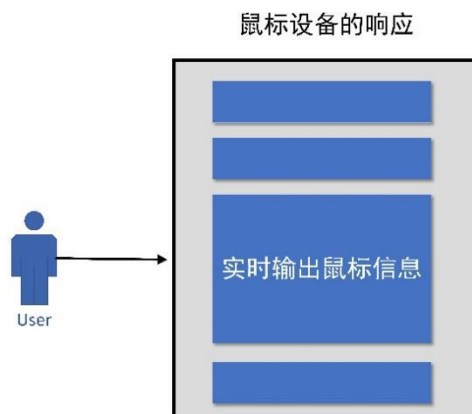


图 5-1 第三次增量迭代用例图

### 5.1.2 功能设计

在设计用户界面时，为了有效展示鼠标坐标信息，我选择了在主容器<main>的子元素<p>中显示这些数据。我首先定义了一个 `mouser` 对象来捕获和保存鼠标状态和位置信息，例如鼠标是否被按下（`mouseisdown`）、鼠标的当前 X 坐标（`mouse.x`）以及鼠标水平移动的增量（`mouse.deltaX`）。接着，我为鼠标事件注册了三个监听器：`mousedown` 用于鼠标按下时的事件捕获，`mousemove` 用于追踪鼠标移动，`mouseout` 用于处理鼠标移出特定区域的情况。最后，我利用<p>元素的 `textContent` 属性，将捕获到的鼠标信息动态更新到页面上。这样的设计允许用户在进行鼠标操作时实时看到其坐标变化。如下图 6-2 项目 Dom 树所示。

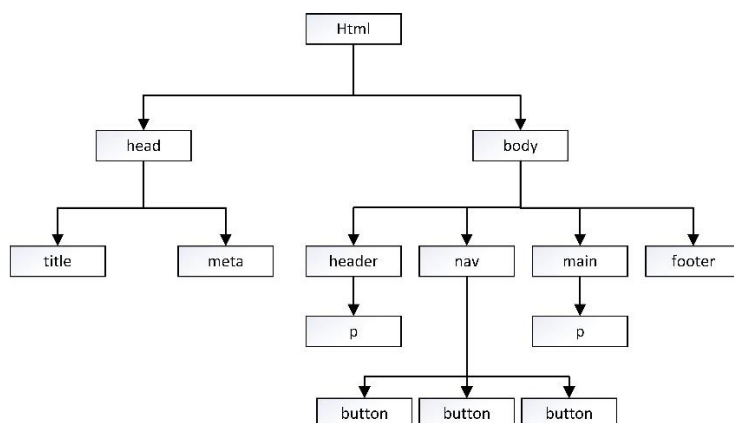


图 5-2 项目 Dom 树所示

## 5.2 代码实现

### 5.2.1 HTML 代码

```

<body>
  <header>
    <p id="head">
      作品展示
    </p>
  </header>

```



```
</p>
</header>
<nav>
  <button title="功能未实现">导航 1</button>
  <button title="功能未实现">导航 2</button>
  <button title="功能未实现">导航 3</button>
</nav>
<main id="main">
  <p class="Content">
    内容展示区
  </p>
</main>
<footer>
  ©彭泱旻江西科技师范大学 2024-2025
</footer>
</body>
```

### 5.2.2 CSS 代码

```
* {
  margin: 2px;
  text-align: center;
  align-content: center;
  justify-content: center;
  font-family: 'KaiTi', '楷体', 'SimKai', 'STKaiti', serif;
  box-sizing: border-box;
  -webkit-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
}
header {
  background-color: #74EBD5;
  background-image: linear-gradient(90deg, #74EBD5 0%, #9FACE6 100%);
  border: 2px solid black;
  height: 15%;
  font-size: 1.66em;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
main {
  background-color: rgb(247, 247, 247);
  border: 2px solid black;
  height: 70%;
  font-size: 1.2em;
  display: flex;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
```

```

}
.Content {
  font-size: 1.1em;
  width: 75%;
  background-color: #FFDEE9;
  background-image: linear-gradient(0deg, #FFDEE9 0%, #B5FFFC 100%);
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}
nav {
  background-color: rgb(247, 247, 247);
  border: 2px solid black;
  height: 10%;
}
nav button {
  font-size: 0.8em;
  padding: 13px 13px;
  box-shadow: 2px 2px 5px 0px rgba(0, 0, 0, 0.3);
  background-color: #8EC5FC;
  background-image: linear-gradient(62deg, #8EC5FC 0%, #E0C3FC 100%);
  border-radius: 8px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}
footer {
  background-color: #FAD961;
  background-image: linear-gradient(90deg, #FAD961 0%, #F76B1C 100%);
  border: 2px solid black;
  height: 5%;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}

```

### 5.2.3 JavaScript 代码

```

Var UI = {};

UI.appWidth = window.innerWidth>600?600:window.innerWidth;
UI.appHeight = window.innerHeight;
Const LETTERS = 22;
Const baseFont = UI.appWidth/LETTERS;
document.body.style.fontSize = baseFont+ "px";
document.body.style.width = UI.appWidth+"px";
document.body.style.height = UI.appHeight - 1 * baseFont + "px";
var mouse = {};

mouse.isDown = false;

mouse.x = 0;
mouse.deltaX = 0;

$(".Content").addEventListener("mousedown", function(ev){

```

```

    Letx = ev.pageX;
    Lety = ev.pageY;
    console.log("鼠标按下了，坐标为: "+"(+x+", "+y+)");
    $(".Content").textContent = "鼠标按下了，坐标为: "+"(+x+", "+y+)";
  });
  $(".Content").addEventListener("mousemove", function(ev){
    Letx = ev.pageX;
    Lety = ev.pageY;
    console.log("鼠标正在移动，坐标为: "+"(+x+", "+y+)");
    $(".Content").textContent="鼠标正在移动，坐标为: "+"(+x+", "+y+)";
  });
  $(".Content").addEventListener("mouseout", function(ev){
    $(".Content").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;
    }
  }
}
}

```

### 5.3 运行与测试

在本节内容中，我实现了对鼠标位置的实时跟踪，并在主容器 main 内的<p>元素中动态展示鼠标的坐标信息。这种实现不仅增强了用户对鼠标移动的可视化感知，而且通过将坐标数据直接嵌入到页面内容中，为用户提供了即时反馈，

提升了交互体验的直观性和互动性。页面效果如下图 5-3 所示

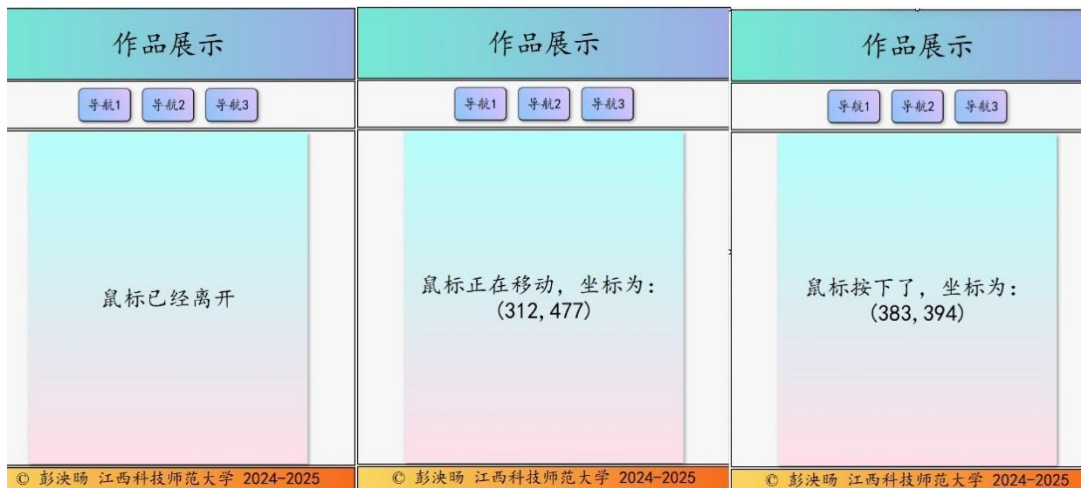


图 5-3 页面效果

分析图 5-3 页面效果可知，代码运行正常，实际效果与预期效果一致。代码可以根据鼠标基本详细的修改而修改内容区的文本。

手机端可以扫描下图 5-4 阶段 3 增量迭代二维码，访问交互 UI 鼠标模型实现文件。



图 5-4 阶段 3 增量迭代二维码

通过构建高效的鼠标模型实现，我确保了网页设计能够灵敏响应用户输入，无论在何种设备上。利用的编程技术，比如事件监听和坐标追踪，我的模型能够精准捕捉鼠标动作，并在不同分辨率和尺寸的屏幕上提供流畅的交互体验。这使得用户在进行网页浏览或执行任务时，享受到了更加直观和一致的操作感受。

### 5.4 代码提交与版本控制

利用 3.4 里面的过程，将代码用 Git Bash 对项目代码进行版本控制如下图 5-5 提交代码图所示。

```
commit 53ba97918354efb1602bd8dbc476b88b6bb20628
Author: 20213624彭洪阳@科师大 <353365869@qq.com>
Date: Wed Jun 12 19:22:05 2024 +0800
```

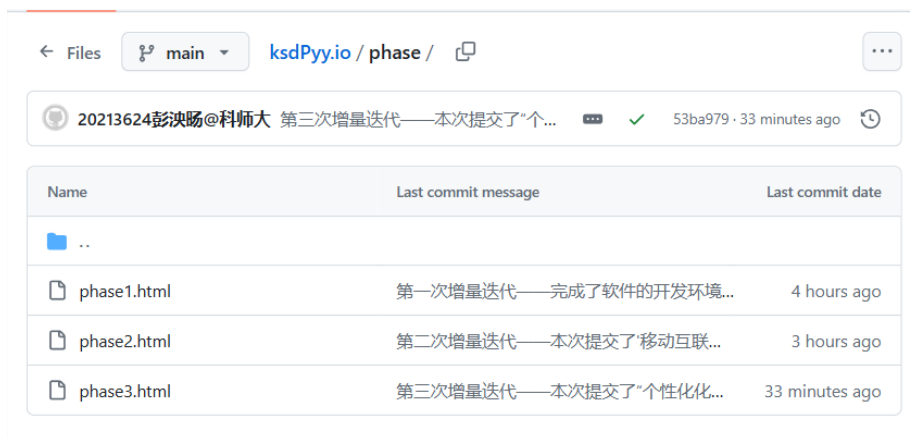
第三次增量迭代——本次提交了“个性化交互UI设计—鼠标模型”代码。对用户设备的页面考虑了不同的大小并思考如何兼容设备页面在页面上准确显示。进行了鼠标与键盘的跟踪，并在页面实时显示出来。用户错误使用系统会在浏览器的控制台抛出具体的错误内容。

图 5-5 提交代码图

将本地代码推送至远端，也就是 GitHub 网址中的代码仓库，第三迭代更新推送过程如下图 5-6 所示推送结果如下图 6-7 远端 GitHub 代码仓库结构所示。

```
R9000X 2021@Lenovo MINGW64 /d/WebUIProject (master)
$ git push origin HEAD:main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 2.62 KiB | 2.62 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/pyy13635987691/ksdPyy.io.git
9502a74..53ba979 HEAD -> main
```

图 5-6 第二迭代更新推送过程



Name	Last commit message	Last commit date
..		
phase1.html	第一次增量迭代——完成了软件的开发环境...	4 hours ago
phase2.html	第二次增量迭代——本次提交了“移动互联...	3 hours ago
phase3.html	第三次增量迭代——本次提交了“个性化化...	33 minutes ago

图 5-7 远端 GitHub 代码仓库结构

## 6. 探索 UI 拖动模拟触屏的操作模式

### 6.1 分析与设计

#### 6.1.1 需求分析

在之前的内容中，我集中讨论了基于鼠标的交互模型，并通过捕获鼠标点击事件来追踪其屏幕位置。这种方法存在局限性，因为它不能转换到触屏设备上。为了解决这个问题，本节我转向了一种新的交互模拟技术：我通过模拟鼠标的横向拖拽动作来复制触屏设备上的滑动操作，从而在移动设备上提供一种替代的交互方式，这一过程在图 6.1 中有所描述。

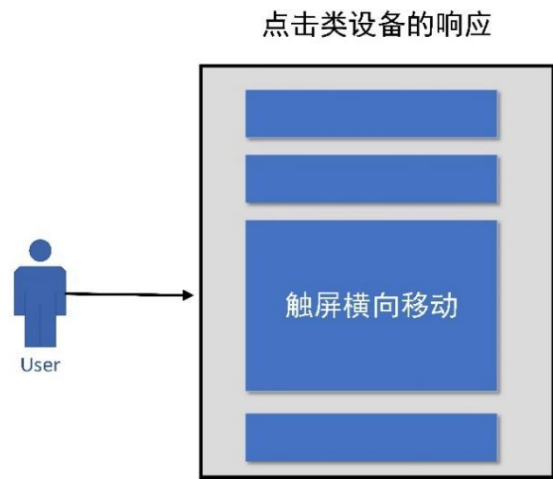


图 6.1 第 4 次增量用例图

6.1.2 功能设计

在设计用户界面时，在页面的部署上面添加了用户键盘响应区，这些需要添加前端元素 `p`、`div` 等元素，本次迭代主要是修改 JavaScript 里面内容，对项目的逻辑进行修改。在逻辑过程里面，我定义了一个 `touch` 对象来追踪触摸点的起始坐标和水平偏移量。接着，通过在 `.Content` 元素上绑定 `touchstart`、`touchmove` 和 `touchend` 事件，代码能够捕获并响应用户的触摸开始、移动和结束动作。在触摸开始时，记录坐标位置；在触摸移动时，计算并显示水平偏移量，并允许内容元素随手指拖动而移动；在触摸结束时，根据偏移量判断滑动是否有效，并给出相应的文本反馈。代码中还包括了一个自定义的 `$` 函数，用于简化 DOM 元素选择过程，增强代码的健壮性和易用性。如下图 6-2 项目 Dom 树所示。

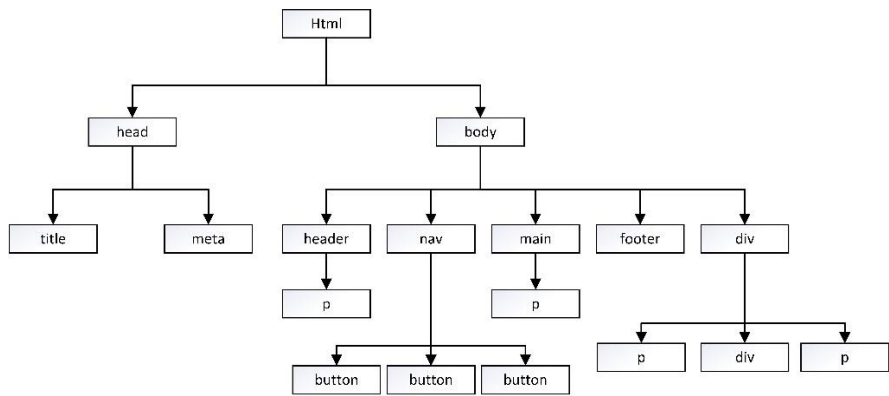


图 6-2 项目 Dom 树

6.2 代码实现

6.2.1 HTML 代码

```
<body>
<header>
```

```
<p id="head">
  作品展示
</p>
</header>
<nav>
  <button title="功能未实现">导航 1</button>
  <button title="功能未实现">导航 2</button>
  <button title="功能未实现">导航 3</button>
</nav>
<main id="main">
  <p class="Content">
    内容展示区
  </p>
</main>
<footer>
  ©彭泱江西科技师范大学 2024-2025
</footer>
<div id="aid">
  <p id="userResponseArea">用户键盘响应区</p>
  <div id="keyboard">
    <p id="displaytxt">
      &nbsp;
    </p>
  </div>
  <p id="displayCode">
    &nbsp;
  </p>
</div>
</body>
```

### 6.2.2 CSS 代码

```
* {
  margin: 2px;
  text-align: center;
  align-content: center;
  justify-content: center;
  font-family: 'KaiTi', '楷体', 'SimKai', 'STKaiti', serif;
  box-sizing: border-box;
  -webkit-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
}
header {
  border: 2px solid black;
```

```

height: 15%;
font-size: 1.66em;
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
main {
  background-color: rgb(247, 247, 247);
  border: 2px solid black;
  height: 70%;
  font-size: 1.2em;
  display: flex;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
.Content {
  font-size: 1.1em;
  position: relative;
  width: 75%;
  background-color: #FFDEE9;
  background-image: linear-gradient(0deg, #FFDEE9 0%, #B5FFFC 100%);
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}
nav {
  background-color: rgb(247, 247, 247);
  border: 2px solid black;
  height: 10%;
}
nav button {
  font-size: 0.8em;
  padding: 13px 13px;
  box-shadow: 2px 2px 5px 0px rgba(0, 0, 0, 0.3);
  background-color: #8EC5FC;
  background-image: linear-gradient(62deg, #8EC5FC 0%, #E0C3FC 100%);
  border-radius: 8px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}
footer {
  border: 2px solid black;
  height: 5%;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
#aid {
  top: 0.3em;
  left: 650px;
  position: absolute;

```



```

align-content: normal;
border: 2px solid black;
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
#userResponseArea {
  margin: 0.8em;
  font-size: 1.66em;
}
#keyboard {
  border: 3px solid black;
  margin-left: 10%;
  width: 80%;
  height: 60%;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}
#displayCode {
  border: 3px solid black;
  margin-left: 10%;
  width: 80%;
  position: absolute;
  bottom: 0.3em;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}

```

### 6.2.3 JavaScript 代码

```

var UI = {};
UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth;
UI.appHeight = window.innerHeight;
const LETTERS = 22;
const baseFont = UI.appWidth / LETTERS;
document.body.style.fontSize = baseFont + "px";
document.body.style.width = UI.appWidth + "px";
document.body.style.height = UI.appHeight - 1 * baseFont + "px";
UI.appHeight = window.innerHeight;
if (window.innerWidth < 1000) {
  $("aid").style.display = 'none';
}
$("aid").style.width = window.innerWidth - UI.appWidth - baseFont * 3 + 'px';
$("aid").style.height = UI.appHeight - baseFont * 1 + 'px';
var mouse = {};
mouse.isDown = false;
mouse.x = 0;
mouse.y = 0;
mouse.deltaX = 0;
$(".Content").addEventListener("mousedown", function(ev) {

```

```

mouse.isDown = true;
mouse.x = ev.pageX;
mouse.y = ev.pageY;
console.log("mouseDown at x: " + "(" + mouse.x + "," + mouse.y + ")");
$(".Content").textContent = "鼠标按下, 坐标: " + "(" + mouse.x + "," + mouse.y + ")";
});
$(".Content").addEventListener("mouseup", function(ev) {
    mouse.isDown = false;
    $(".Content").style.left = 0 + 'px';
    $(".Content").textContent = "鼠标松开!";
    if (Math.abs(mouse.deltaX) > 100) {
        $(".Content").textContent += " 这是有效拖动! ";
    } else {
        $(".Content").textContent += " 本次算无效拖动! ";
    }
});
$(".Content").addEventListener("mouseout", function(ev) {
    ev.preventDefault();
    mouse.isDown = false;
    $(".Content").textContent = "鼠标离开!";
    if (Math.abs(mouse.deltaX) > 100) {
        $(".Content").textContent += " 这次是有效拖动! ";
    } else {
        $(".Content").textContent += " 本次算无效拖动! ";
    }
});
$(".Content").addEventListener("mousemove", function(ev) {
    ev.preventDefault();
    if (mouse.isDown) {
        console.log("mouse isDown and moving");
        mouse.deltaX = parseInt(ev.pageX - mouse.x);
        $(".Content").textContent = "正在拖动鼠标, 距离: " + mouse.deltaX + "px 。";
        $(".Content").style.left = mouse.deltaX + 'px';
    }
});
var touch = {
    startX: 0,
    startY: 0,
    deltaX: 0
};
$(".Content").addEventListener("touchstart", function(ev) {
    var touchEvent = ev.touches[0];
    touch.startX = touchEvent.pageX;
    touch.startY = touchEvent.pageY;

```

```

    $(".Content").textContent = "触屏开始, 坐标: " + "(" + parseInt(touch.startX, 10) + ", " +
    parseInt(touch.startY, 10) + ")";
  });
  $(".Content").addEventListener("touchmove", function(ev) {
    ev.preventDefault();
    var touchEvent = ev.touches[0];
    touch.deltaX = parseInt(touchEvent.pageX - touch.startX);
    $(".Content").textContent = "正在触屏滑动, 距离: " + touch.deltaX + "px 。";
    $(".Content").style.left = touch.deltaX + 'px';
  });
  $(".Content").addEventListener("touchend", function(ev) {
    $(".Content").textContent = "触屏结束!";
    if (Math.abs(touch.deltaX) > 100) {
      $(".Content").textContent += ", 这是有效滑动! ";
    } else {
      $(".Content").textContent += " 本次算无效滑动! ";
    }
  });
  function $(ele) {
    if (typeof ele !== 'string') {
      throw ("自定义的$函数参数的数据类型错误, 实参必须是字符串!");
      return
    }
    let dom = document.getElementById(ele);
    if (dom) {
      return dom;
    } else {
      dom = document.querySelector(ele);
      if (dom) {
        return dom;
      } else {
        throw ("执行$函数未能在页面上获取任何元素, 请自查问题!");
        return;
      }
    }
  }
} //end of $

```

## 6.3 运行与测试

在本节内容中, 我开发了一个模拟触屏操作的功能, 实时捕捉并显示触摸点的坐标信息。通过在主容器 `main` 内的 `<p>` 元素上动态反映触摸的位置数据, 这种实现方法不仅提升了用户对触摸动作的空间感知能力, 而且通过即时在页面上展示触摸反馈, 增强了用户界面的交互性和直观性, 使用户能够更直观地

与内容进行互动。页面效果如下图 6-3 所示

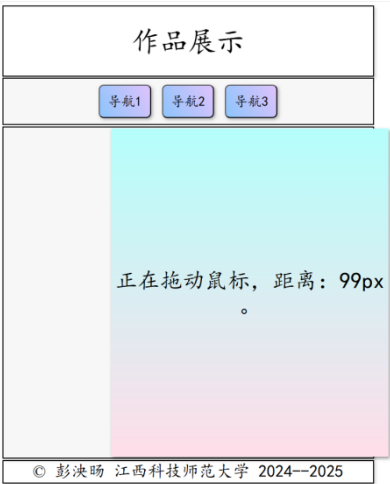


图 6-3 页面效果

分析图 6-3 页面效果可知，代码运行正常，实际效果与预期效果一致。代码可以根据模拟触屏调节内容区的位置，运行效果良好。

手机端可以扫描下图 7-4 阶段 4 增量迭代二维码，访问 UI 拖动模拟触屏后的操作模式页面。



图 6-4 阶段 4 增量迭代二维码

在本节中，我开发了一个的触屏模拟系统，该系统通过精心设计的算法和用户界面技术，确保了网页在各种设备上都能提供卓越的触摸交互体验。通过集成复杂的事件处理程序和动态坐标跟踪机制，系统能够准确识别和响应用户的每一个触摸动作，无论是轻触、滑动还是长按。此外，系统还特别优化了在不同屏幕尺寸和分辨率下的显示效果和交互逻辑，确保用户在浏览网页或完成特定任务时，都能获得一致、流畅且直观的触觉反馈。这一创新方法大大提升了用户与数字内容之间的互动质量，使得触屏操作更加自然和直观。

## 6.4 代码提交与版本控制

利用 3.4 里面的过程，将代码用 Git Bash 对项目代码进行版本控制如下图 6-5 提交代码图所示。

```
commit d0201ee733a71dd85bfd2cef06774ce4781bfb08
Author: 20213624彭决阳@科师大 <353365869@qq.com>
Date:   Wed Jun 12 20:59:50 2024 +0800
```

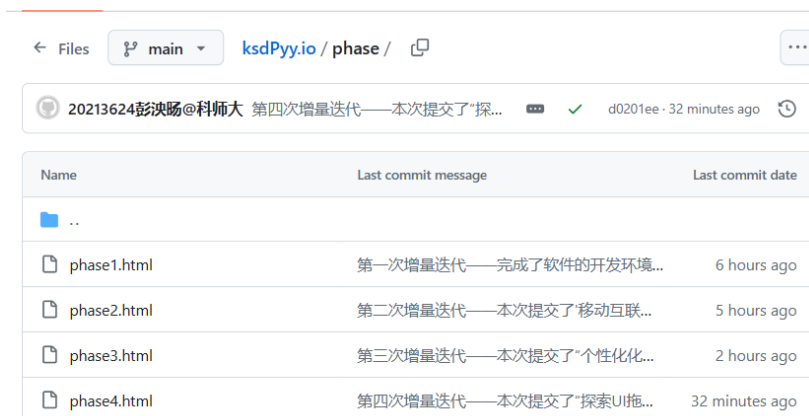
第四次增量迭代—本次提交了“探索UI拖动模型模拟触屏的操作模式”代码，本次添加了内容区的可移动功能，可以在计算机上面利用鼠标进行拖动，离实现移动端的内容区触屏移动更进一步，单次的内容区移动有判断条件，单次移动必须超过100px、才判定为有效移动，否则是无效移动。

图 6-5 提交代码图

将本地代码推送至远端，也就是 GitHub 网址中的代码仓库，第四迭代更新推送过程如下图 6-6 所示推送结果如下图 6-7 远端 GitHub 代码仓库结构所示。

```
R9000X 2021@Lenovo MINGW64 /d/WebUIProject (master)
$ git push origin HEAD:main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 3.05 KiB | 3.05 MiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/pyy13635987691/ksdPyPy.io.git
53ba979..d0201ee HEAD -> main
```

图 6-6 第四次迭代更新推送过程



Name	Last commit message	Last commit date
..		
phase1.html	第一次增量迭代——完成了软件的开发环境...	6 hours ago
phase2.html	第二次增量迭代——本次提交了移动互联...	5 hours ago
phase3.html	第三次增量迭代——本次提交了“个性化...	2 hours ago
phase4.html	第四次增量迭代——本次提交了“探索UI拖...	32 minutes ago

图 6-7 远端 GitHub 代码仓库结构

## 7. 通用 UI 设计为触屏和鼠标统一建模

### 7.1 分析与设计

#### 7.1.1 需求分析

为了提高 Web 应用开发的效率和软件质量，统一建模被应用于确立一致性标准和最佳实践。这种方法旨在增强代码的可维护性、可读性和可扩展性，同时促进团队协作和代码复用。为了简化调试和问题解决，统一建模还致力于创建易于理解和维护的代码结构。如下图 7-1 项目第 5 次增量用例图所示。

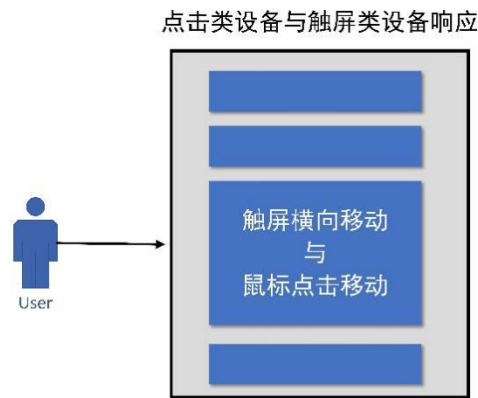


图 7-1 第 5 次增量项目用例图

### 7.1.2 功能设计

第 5 次迭代更新代码主要实现了一整套基于触摸和鼠标事件的交互逻辑，允许用户通过拖动或滑动来控制页面上的元素。此外，代码中还包含了一个自定义的\$函数，用于简化 DOM 元素的选择过程。如下图 7-2 项目 Dom 树所示。

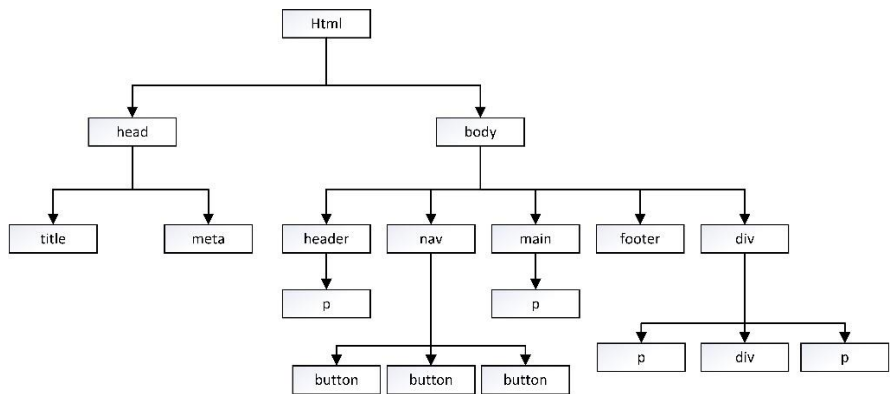


图 7-2 项目 Dom 树

## 7.2 代码实现

### 7.2.1 HTML 代码实现

```
<body>
  <header>
    <p id="head">
      作品展示
    </p>
  </header>
  <nav>
    <button title="功能未实现">导航 1</button>
    <button title="功能未实现">导航 2</button>
    <button title="功能未实现">导航 3</button>
  </nav>
  <main id="main">
    <p class="Content">
```

```

        内容展示区
    </p>
</main>
<footer>
    ©彭映旻江西科技师范大学 2024-2025
</footer>
    <div id="aid">
        <p id="userResponseArea">用户键盘响应区</p>
        <div id="keyboard">
            <p id="displaytxt">
                &nbsp;
            </p>
        </div>
        <p id="displayCode">
            &nbsp;
        </p>
    </div>
</body>

```

### 7.2.2 CSS 代码

```

* {
    margin: 2px;
    text-align: center;
    align-content: center;
    justify-content: center;
    font-family: 'KaiTi', '楷体', 'SimKai', 'STKaiti', serif;
    box-sizing: border-box;
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}
header {
    border: 2px solid black;
    height: 15%;
    font-size: 1.66em;
    box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
main {
    background-color: rgb(247, 247, 247);
    border: 2px solid black;
    height: 70%;
    font-size: 1.2em;
    display: flex;
    box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}

```

```

}
.Content {
  font-size: 1.1em;
  position: relative;
  width: 75%;
  background-color: #FFDEE9;
  background-image: linear-gradient(0deg, #FFDEE9 0%, #B5FFFC 100%);
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}
nav {
  background-color: rgb(247, 247, 247);
  border: 2px solid black;
  height: 10%;
}
nav button {
  font-size: 0.8em;
  padding: 13px 13px;
  box-shadow: 2px 2px 5px 0px rgba(0, 0, 0, 0.3);
  background-color: #8EC5FC;
  background-image: linear-gradient(62deg, #8EC5FC 0%, #E0C3FC 100%);
  border-radius: 8px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}
footer {
  border: 2px solid black;
  height: 5%;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
#aid {
  top: 0.3em;
  left: 650px;
  position: absolute;
  align-content: normal;
  border: 2px solid black;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
#userResponseArea {
  margin: 0.8em;
  font-size: 1.66em;
}
#keyboard {
  border: 3px solid black;
  margin-left: 10%;
}

```



```

width: 80%;
height: 60%;
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}
#displayCode {
border: 3px solid black;
margin-left: 10%;
width: 80%;
position: absolute;
bottom: 0.3em;
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}

```

### 7.2.3 JavaScript 代码

```

var UI = {};
if (window.innerWidth > 600) {
  UI.appWidth = 600;
} else {
  UI.appWidth = window.innerWidth;
}
UI.appHeight = window.innerHeight;
let baseFont = UI.appWidth / 20;
document.body.style.fontSize = baseFont + "px";
document.body.style.width = UI.appWidth + "px";
document.body.style.height = UI.appHeight - baseFont * 1 + "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 * 1 + 'px';
var Pointer = {};
Pointer.isDown = false;
Pointer.x = 0;
Pointer.deltaX = 0; {
  let handleBegin = function(ev) {
    Pointer.isDown = true;
    if (ev.touches) {
      console.log("touches1" + ev.touches);
      Pointer.x = parseInt(ev.touches[0].pageX);
      Pointer.y = parseInt(ev.touches[0].pageY);
      console.log("Touch begin : " + "(" + Pointer.x + "," + Pointer.y + ")");
    }
    $(".Content").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 + ")");
$(".Content").textContent = "鼠标按下, 坐标: " + "(" + Pointer.x + ", " + Pointer.y + ")";
}
};

let handleEnd = function(ev) {
  Pointer.isDown = false;
  ev.preventDefault()
  if (ev.touches) {
    $(".Content").textContent = "触屏事件结束!";
    $(".Content").style.left = 0 + 'px';
    if (Math.abs(Pointer.deltaX) > 100) {
      $(".Content").textContent += ", 这是有效触屏滑动! ";
    } else {
      $(".Content").textContent += " 本次算无效触屏滑动! ";
      $(".Content").style.left = Pointer.deltaX + 'px';
    }
  } else {
    $(".Content").textContent = "鼠标松开!";
    $(".Content").style.left = 0 + 'px';
    if (Math.abs(Pointer.deltaX) > 100) {
      $(".Content").textContent += ", 这是有效拖动! ";
    } else {
      $(".Content").textContent += " 本次算无效拖动! ";
      $(".Content").style.left = Pointer.deltaX + 'px';
    }
  }
}
};

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);
      $(".Content").textContent = "正在滑动触屏, 滑动距离: " + Pointer.deltaX + "px 。 ";
      $(".Content").style.left = Pointer.deltaX + 'px';
    }
  } else {
    if (Pointer.isDown) {
      console.log("Pointer isDown and moving");
      Pointer.deltaX = parseInt(ev.pageX - Pointer.x);
      $(".Content").textContent = "正在拖动鼠标, 距离: " + Pointer.deltaX + "px 。 ";
      $(".Content").style.left = Pointer.deltaX + 'px';
    }
  }
}
}

```

```

};
$(".Content").addEventListener("mousedown", handleBegin);
$(".Content").addEventListener("touchstart", handleBegin);
$(".Content").addEventListener("mouseup", handleEnd);
$(".Content").addEventListener("touchend", handleEnd);
$(".Content").addEventListener("mouseout", handleEnd);
$(".Content").addEventListener("mousemove", handleMoving);
$(".Content").addEventListener("touchmove", handleMoving);
}
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;
    }
  }
}
}

```

### 7.3 运行与测试

在本节内容中，我将上一次迭代实现的实时捕捉并显示触摸点的坐标信息与触屏类响应的实时信息显示在屏幕上的功能后端代码整合成一套代码，也就是通用代码的实现，减少 JavaScript 的运行压力。下面分别展示 iPhone 12 Pro 设备与 Nest Hub Max 设备运行效果与差异，页面效果如下图 7-3 所示。



图 7-3 页面效果

分析图 7-3 页面效果可知，代码运行正常，实际效果与预期效果一致。代码可以根据模拟触屏调节内容区的位置，运行效果良好。

手机端可以扫描下图 7-4 阶段 5 增量迭代二维码，访问 UI 拖动模拟触屏后的操作模式页面。



图 7-4 阶段 5 增量迭代二维码

在本节中，我开发了一个触屏模拟系统，该系统通过设计的算法和用户界面技术，确保了网页在各种设备上都能提供卓越的触摸交互体验。通过集成复杂的事件处理程序和动态坐标跟踪机制，系统能够准确识别和响应用户的每一个触摸动作，无论是轻触、滑动还是长按。此外，系统还特别优化了在不同屏幕尺寸和分辨率下的显示效果和交互逻辑，确保用户在浏览网页或完成特定任务时，都能获得一致、流畅且直观的触觉反馈。这一创新方法大大提升了用户与数字内容之间的互动质量，使得触屏操作更加自然和直观。

#### 7.4 代码提交与版本控制

利用 3.4 里面的过程，将代码用 Git Bash 对项目代码进行版本控制如下图 7-5 提交代码图所示。

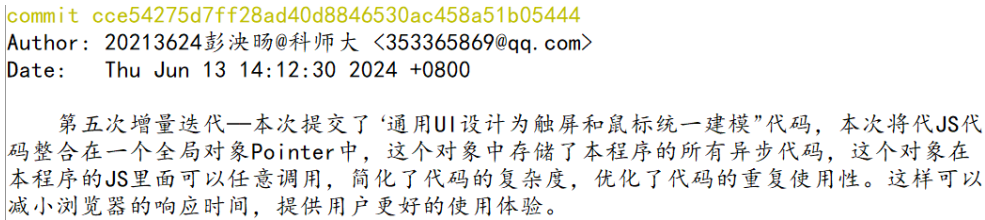


图 6-5 提交代码图

将本地代码推送至远端，也就是 GitHub 网址中的代码仓库，第四迭代更新推送过程如下图 6-6 所示推送结果如下图 6-7 远端 GitHub 代码仓库结构所示。

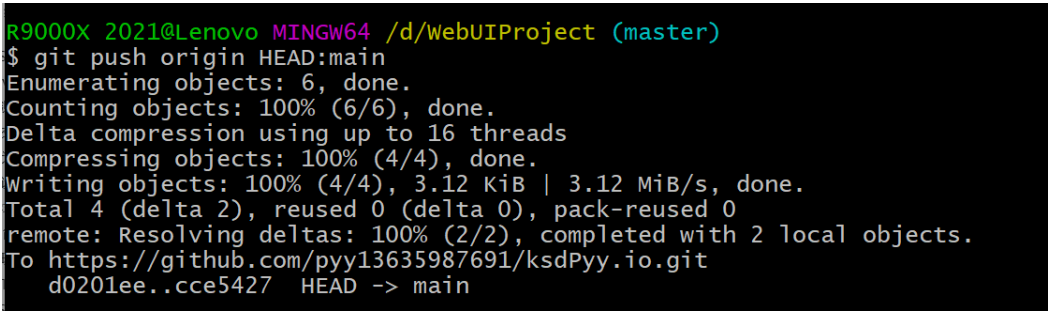


图 6-6 第四次迭代更新推送过程

20213624彭决阳@科... 第五次增量迭代——本次提交了“通用U... cce5427 · 3 minutes ago		
Name	Last commit message	Last commit date
..		
phase1.html	第一次增量迭代——完成了软件的开发环境...	yesterday
phase2.html	第二次增量迭代——本次提交了“移动互联...	yesterday
phase3.html	第三次增量迭代——本次提交了“个性化化...	18 hours ago
phase4.html	第四次增量迭代——本次提交了“探索UI拖...	17 hours ago
phase5.html	第五次增量迭代——本次提交了“通用UI设...	3 minutes ago

图 6-7 远端 GitHub 代码仓库结构

## 8. 个性化 UI 监控键盘实现

### 8.1 分析与设计

#### 8.1.1 需求分析

键盘作为计算机及多种电子设备的核心输入工具，扮演着至关重要的角色。它不仅为用户提供了一个直观、高效的文本和指令输入手段，而且通过功能键和快捷键的辅助，极大提升了操作的灵活性和工作效率。

为了深入掌握键盘的工作原理及其在用户界面中的实现方式，我专门添加了一个交互式的用户键盘响应区域，用以探究和演示键盘输入如何影响页面信息的展示。该设计包括了详尽的用户界面布局和响应机制，如图 8.1 描绘，旨在通过直观的展示和动态反馈，增强用户对键盘事件处理流程的认识。通过这一实践，用户不仅能够观察到按键行为如何被系统捕获和解析，还能够看到这些行为如何触发特定的页面反应，从而在提升用户体验的同时，也加深了对键盘交互背后技术的理解。

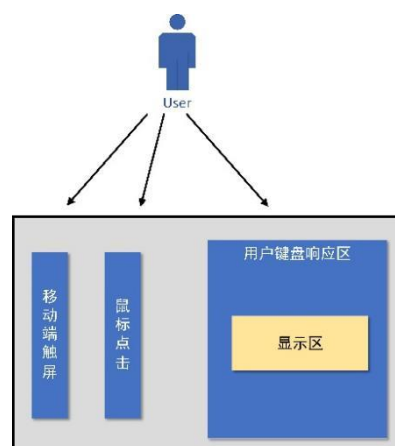
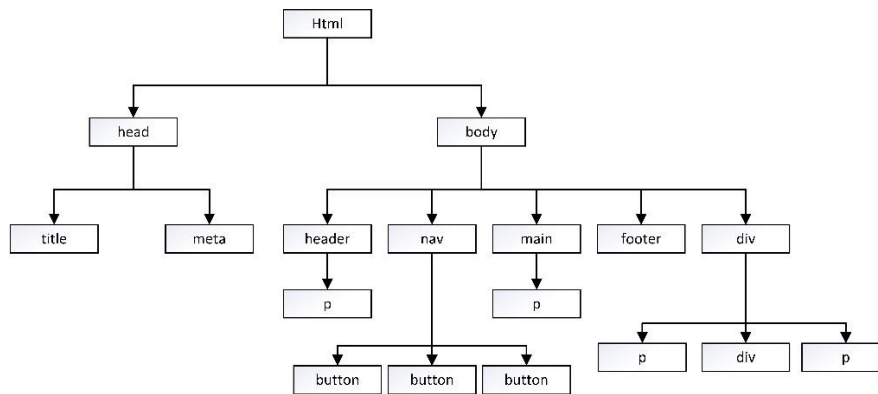


图 8.1 键盘响应用例图

### 8.1.2 功能设计

这段代码通过监听键盘的按下和释放事件，添加了一个实时响应用户输入的 Web 界面。它能够捕捉并显示用户按下的键及其编码，同时在页面上的特定元素中动态展示输入内容。系统特别设计了对 Enter 键的换行和 Backspace 键的删除功能，同时确保只有有效的字符（包括字母、数字和选定的标点符号）被添加到显示区域。此外，自定义的\$函数简化了 DOM 元素的选择过程，增强了代码的健壮性。实现过程不仅提升了用户交互体验，还通过即时反馈增强了界面的直观性和互动性。如下图 8-2 项目 Dom 树所示。



图如下图 8-2 项目 Dom 树所示

## 8.2 代码实现

### 8.2.1 HTML 代码

```

<body>
  <header>
    <p id="head">
      作品展示
    </p>
  </header>
  <nav>
    <button title="功能未实现">导航 1</button>
    <button title="功能未实现">导航 2</button>
    <button title="功能未实现">导航 3</button>
  </nav>
  <main id="main">
    <p class="Content">
      内容展示区
    </p>
  </main>
  <footer>
    ©彭泱旸江西科技师范大学 2024-2025
  </footer>
  <div id="aid">
    <p id="userResponseArea">用户键盘响应区</p>
    <div id="keyboard">
      <p id="displaytxt">
        &nbsp;
      </p>
    </div>
    <p id="displayCode">
      &nbsp;
    </p>
  </div>
  
```

<body>

## 8.2.2 CSS 代码

```
{
  margin: 2px;
  text-align: center;
  align-content: center;
  justify-content: center;
  font-family: 'KaiTi', '楷体', 'SimKai', 'STKaiti', serif;
  box-sizing: border-box;
  -webkit-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
}
header {
  border: 2px solid black;
  height: 15%;
  font-size: 1.66em;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
main {
  background-color: rgb(247, 247, 247);
  border: 2px solid black;
  height: 70%;
  font-size: 1.2em;
  display: flex;
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}
.Content {
  font-size: 1.1em;
  position: relative;
  width: 75%;
  background-color: #FFDEE9;
  background-image: linear-gradient(0deg, #FFDEE9 0%, #B5FFFC 100%);
  box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}
nav {
  background-color: rgb(247, 247, 247);
  border: 2px solid black;
  height: 10%;
}
nav button {
  font-size: 0.8em;
  padding: 13px 13px;
```



```

box-shadow: 2px 2px 5px 0px rgba(0, 0, 0, 0.3);
background-color: #8EC5FC;
background-image: linear-gradient(62deg, #8EC5FC 0%, #E0C3FC 100%);
border-radius: 8px;
cursor: pointer;
transition: background-color 0.3s ease;
}

footer {
border: 2px solid black;
height: 5%;
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}

#aid {
top: 0.3em;
left: 650px;
position: absolute;
align-content: normal;
border: 2px solid black;
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.2);
}

#userResponseArea {
margin: 0.8em;
font-size: 1.66em;
}

#keyboard {
border: 3px solid black;
margin-left: 10%;
width: 80%;
height: 60%;
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}

#displayCode {
border: 3px solid black;
margin-left: 10%;
width: 80%;
position: absolute;
bottom: 0.3em;
box-shadow: 2px 3px 5px 0px rgba(0, 0, 0, 0.4);
}

```

### 8.2.3 JavaScript 代码

```

var UI = {};
UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth;
UI.appHeight = window.innerHeight;
const Letters = 22; // 字母数量

```

```

const baseFont = UI.appWidth / Letters; // 基准字体大小
// 设置页面的字体大小为默认字体大小
document.body.style.fontSize = baseFont + 'px';
// 通过动态 CSS 设置页面全屏显示
document.body.style.width = UI.appWidth + 'px';
document.body.style.height = UI.appHeight - 1 * baseFont + "px";
if (window.innerWidth < 1000) {
  $("#aid").style.display = 'none';
}
$("#aid").style.width = window.innerWidth - UI.appWidth - baseFont * 3 + 'px';
$("#aid").style.height = UI.appHeight - 1 * baseFont + 'px';
var mouse = {};
mouse.isDown = false;
mouse.x = 0;
mouse.y = 0;
mouse.deltaX = 0;
$(".Content").addEventListener("mousedown", function(ev) {
  mouse.isDown = true;
  mouse.x = ev.pageX;
  mouse.y = ev.pageY;
  console.log("mouseDown at x: " + "(" + mouse.x + "," + mouse.y + ")");
  $(".Content").textContent = "鼠标按下, 坐标: " + "(" + mouse.x + "," + mouse.y + ")";
});
$(".Content").addEventListener("mouseup", function(ev) {
  mouse.isDown = false;
  $(".Content").textContent = "鼠标松开!";
  $(".Content").style.left = 0 + 'px';
  if (Math.abs(mouse.deltaX) > 100) {
    $(".Content").textContent += " 这是有效拖动! ";
  } else {
    $(".Content").textContent += " 本次算无效拖动! ";
  }
});
$(".Content").addEventListener("mouseout", function(ev) {
  ev.preventDefault();
  mouse.isDown = false;
  $(".Content").textContent = "鼠标离开!";
  if (Math.abs(mouse.deltaX) > 100) {
    $(".Content").textContent += " 这次是有效拖动! ";
  } else {
    $(".Content").textContent += " 本次算无效拖动! ";
  }
});
$(".Content").addEventListener("mousemove", function(ev) {

```

```

ev.preventDefault();
if (mouse.isDown) {
    console.log("mouse isDown and moving");
    mouse.deltaX = parseInt(ev.pageX - mouse.x);
    $(".Content").textContent = "正在拖动鼠标, 距离: " + mouse.deltaX + "px 。";
    $(".Content").style.left = mouse.deltaX + 'px';
}
});
$("body").addEventListener("keydown", function(ev) {
    let k = ev.key;
    let c = ev.keyCode;
    $(".displayCode").textContent = "您已按下键 : " + k + " , " + "字符编码 : " + c;
});
$("body").addEventListener("keyup", function(ev) {
    let k = ev.key;
    let c = ev.keyCode;
    $(".displayCode").textContent = "松开按键 : " + k + " , " + "字符编码 : " + c;
    if (k === "Enter") {
        let p = document.createElement("p");
        $(".keyboard").appendChild(p);
    } else if (k === "Backspace") {
        if ($(".keyboard").lastElementChild.textContent === "") {
            if ($(".keyboard").childElementCount > 1) {
                $(".keyboard").removeChild($(".keyboard").lastElementChild);
            }
        } else {
            $(".keyboard").lastElementChild.textContent = $(".keyboard").lastElementChild.textContent.slice(0, -1);
        }
    } else if (printLetter(k)) {
        $(".keyboard").lastElementChild.textContent += k;
    }
}
function printLetter(k) {
    //判断字符串长度是否大于 1
    if (k.length > 1) {
        return false;
    }
    let puncs = ['~', ' ', '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '_', '+', '=', ',', '.', '<', '>', '?', '/', '"', ' '];
    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;
}
});
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;
        }
    }
}
```

### 8.3 运行与测试

在本章节，我们深入探索了键盘交互的内在机制，开发了一个创新的键盘响应展示平台。该平台位于网页的侧面展示区域 **main**，通过一个精心设计的 `<div>` 标签，动态捕捉并可视化用户的键盘敲击动作。这种直观的展示不仅加深了用户对键盘输入过程的理解，而且通过实时的页面更新，提供了一种新颖的反馈形式，极大地丰富了人机交互的深度和广度。用户现在可以直观地观察到每个按键如何触发页面元素的变化，从而在提升操作透明度的同时，也增加了交互的趣味性。如下图 8-3 页面效果所示。

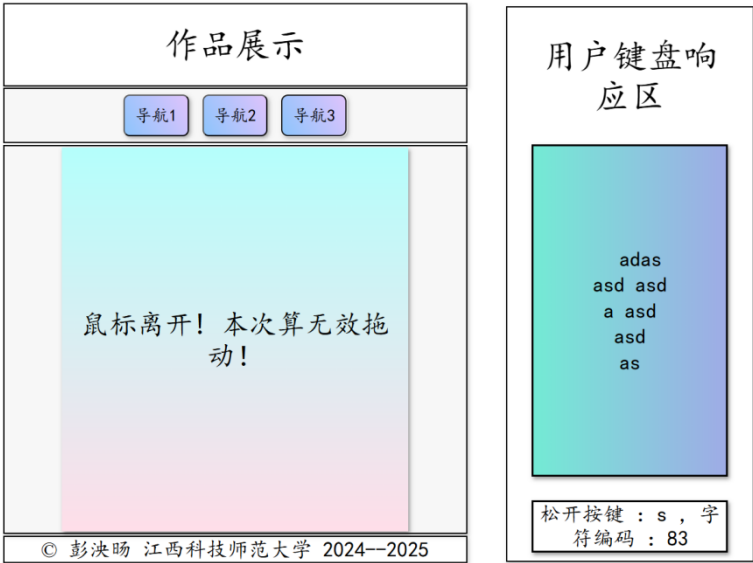


图 8-3 页面效果

分析图 8-3 页面效果可知，代码运行正常，实际效果与预期效果一致。代码可以根据模拟触屏调节内容区的位置，运行效果良好。

手机端可以扫描下图 8-4 阶段 6 增量迭代二维码，访问 UI 拖动模拟触屏后的操作模式页面。



图 8-4 阶段 6 增量迭代二维码

#### 8.4 代码提交与版本控制

利用 4.4 里面的过程，将代码用 Git Bash 对项目代码进行版本控制如下图 8-5 提交代码图所示。

```
commit 4a0c13931212fa4c78d311268773bb1c33ec9f0a (HEAD -> master, origin/main)
Author: 20213624彭泱昀@科师大 <353365869@qq.com>
Date: Thu Jun 13 14:19:37 2024 +0800

第六次增量迭代—本次提交了“个性化UI监控键盘实现”代码，这段代码通过监听键盘的按下和释放事件，创建了一个实时响应用户输入的Web界面。它能够捕捉并显示用户按下的键及其编码，同时在页面上的特定元素中动态展示输入内容。系统特别设计了对Enter键的换行和Backspace键的删除功能，同时确保只有有效的字符（包括字母、数字和选定的标点符号）被添加到显示区域。此外，自定义的$函数简化了DOM元素的选择过程，增强了代码的健壮性。实现过程不仅提升了用户交互体验，还通过即时反馈增强了界面的直观性和互动性。
```

图 8-5 提交代码图

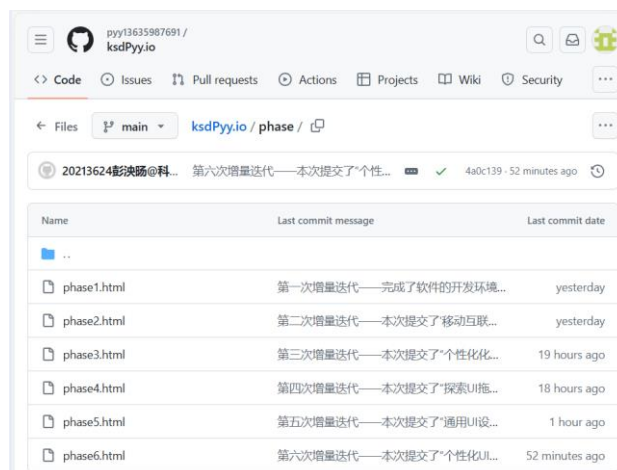
将本地代码推送至远端，也就是 GitHub 网址中的代码仓库，第四迭代更新推送过程如下图 8-6 所示推送结果如下图 8-7 远端 GitHub 代码仓库结构所示。

```

R9000X 2021@Lenovo MINGW64 /d/WebUIProject (master)
$ git push origin HEAD:main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 3.05 KiB | 3.05 MiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/pyy13635987691/ksdPyy.io.git
53ba979..d0201ee HEAD -> main

```

图 8-6 第四次迭代更新推送过程



Name	Last commit message	Last commit date
..		
phase1.html	第一次增量迭代——完成了软件的开发环境...	yesterday
phase2.html	第二次增量迭代——本次提交了移动互联...	yesterday
phase3.html	第三次增量迭代——本次提交了“个性化化...	19 hours ago
phase4.html	第四次增量迭代——本次提交了“探索UI施...	18 hours ago
phase5.html	第五次增量迭代——本次提交了“通用UI设...	1 hour ago
phase6.html	第六次增量迭代——本次提交了“个性化UI...	52 minutes ago

图 8-7 远端 GitHub 代码仓库结构

## 9. 用 Git Bash 工具管理项目的代码仓库和 Http 服务器

### 9.1 跨世纪的经典 Bash 工具


当我们讨论命令行界面，我们实际上是在描述一种用户界面，它允许用户通过文本命令与计算机系统交互。这种界面通常由Shell程序提供，Shell是一个中介软件，它接收用户的文本输入（命令），然后向操作系统发出相应的执行请求。在Linux系统中，Bash（Bourne Again Shell）是最常见的Shell，它继承自最初的Unix Shell程序，由Steve Bourne开发，并提供了更多的功能和改进。

Linux和类Unix系统采用了一种树状的文件系统结构，这种结构将文件和目录以层次化的方式组织起来，类似于一个家族树。在这个结构中，存在一个最顶层的目录，被称为根目录，它是所有文件和目录的起始点。从根目录延伸出的每个分支都可能包含文件或者指向其他目录的链接，这些目录又可以包含更多的文件和目录，形成一个复杂的层次网络。这种组织方式使得文件管理有序且易于导航<sup>[9]</sup>。

### 9.2 通过 Git Hub 平台实现本项目的全球域名。

## GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://pvy13635987691.github.io/ksdPyy.io/>  
Last deployed by  pvy13635987691 4 days ago

[Visit site](#)

...

图 9.1 本项目的全球域名

如上图 9.1 所示，为本项目在Git Hub平台上的全球域名，可以在PC端的微软Edge浏览器中输入该域名进行访问。

## 9.3 创建一个空的远程代码仓库

Required fields are marked with an asterisk (\*).

Owner \*

 pvy13635987691

Repository name \*

/ KsdPyy

✔ KsdPyy is available.

Great repository names are short and memorable. Need inspiration? How about [miniature-chainsaw](#) ?

Create repository

图 9.2 创建空的远程代码仓库

如上图 9.2 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。

## 9.4 设置本地仓库和远程代码仓库的链接

进入本地web UI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

```
$ git init
$ git commit -m "把代码仓库上传至 git hub 平台"
$ git branch -m master main
$ git remote add origin
https://github.com/pvy13635987691/ksdPyy.github.io.git
$ git push -u origin main
```

本项目使用window平台，git bash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 9.3 所示：

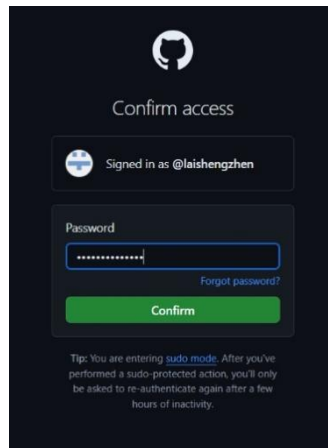


图 9.3 开发者授权

如下图 9.4 再次确认授权git bash拥有访问改动远程代码的权限，如下图所示：

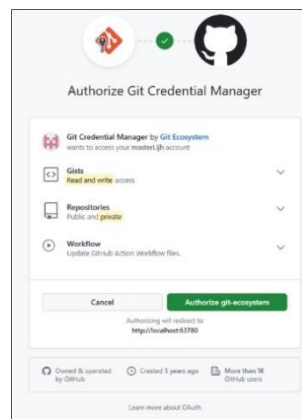


图 9.4 再次确认授权

最后，GitHub平台反馈：git bash和git hub平台成功实现远程链接如下图 9.5 所示。

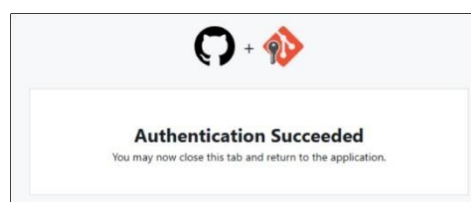


图 9.5 成功实现远程连接

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git hub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，如下图 9.6 所示：



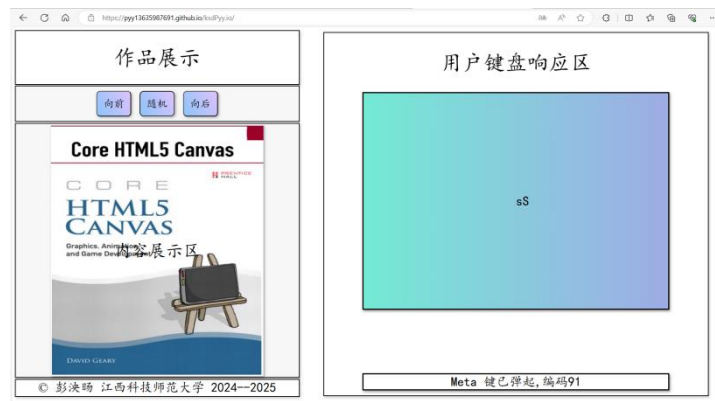


图 9.6 项目在互联网的部署

## 参考文献

- [1] Douglas E. Comer. The Internet Book [M] (Fifth Edition). CRC Press Taylor & Francis Group, 2019: 217-218
- [2] Douglas E. Comer. The Internet Book [M] (Fifth Edition). CRC Press Taylor & Francis Group, 2019: 217-218
- [3] 朱晓峰, 王忠军, 张卫. 大数据分析指南[M]. 南京大学出版社, 2021.
- [4] 杨洋, 刘全. 软件系统分析与体系结构设计[M]. 南京东南大学出版社, 2017.
- [5] 耿红琴. 软件工程案例教程[M]. 电子工业出版社, 2018.
- [6] 钟珞, 袁胜琼, 袁景凌, 等. 软件工程[M]. 人民邮电出版社, 2017.
- [7] 宋明秋. 软件安全开发[M]. 电子工业出版社, 2016.
- [8] 张伟. 多学科视域中的 MOOC 研究[M]. 中国人民大学出版社, 2022.
- [9] 张敬, 李江涛, 张振峰, 等. 企业网络安全建设最佳实践[M]. 电子工业出版社, 2021.