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: 张荣军

学　号　: 20213623

指导教师: 李健宏

2024 年 6 月 17 日

# 目录

# 基于 Web 客户端技术个性化 UI 设计与实现

**摘要：**本项目专注于研究和应用 Web 客户端技术，以适应移动互联网时代的前端需求。通过广泛查阅技术资料和相关文献，特别是参考了 Mozilla 组织的 MDN 社区的实践文章，深入探索了 HTML 内容建模、CSS 样式设计和 JavaScript 功能编程的基本技术和技巧。项目的重点在于实现一个响应式用户界面（UI），以确保最佳适配不同设备的屏幕，包括 PC 端和移动端。在功能方面，项目采用了 DOM 技术和事件驱动模式，实现了对鼠标、触屏和键盘事件的底层响应和流畅支持。特别地，项目还创新性地设计和实现了一个对象模型，用于模拟鼠标和触屏设备的指向性操作，这是项目的一个亮点和研究创新。为了有效管理设计与开发过程，项目采用了工程思想和软件工程的增量式开发模式。经过 6 次迭代开发，每次迭代都严格按照分析、设计、实现和测试（A:Analysis, D:Design, I:Implementation, T:Testing）的经典开发阶段进行，以逐步完善和优化 UI 应用程序。为了促进代码的分享与合作，项目利用 Git 工具进行版本控制和开发过程的详细记录，共进行了 12 次代码提交。最终，项目将代码托管在 GitHub 上，通过设置为 HTTP 服务器，实现了全球范围内对 UI 应用的便捷访问。这一系列的技术实践和方法论，不仅提升了开发效率和用户体验，也为今后在 Web 开发领域的探索和应用奠定了坚实的基础。

**关键字：**Web 客户端技术;JavaScript;Git 工具;GitHub;HTTP 服务器

# 1. 引言

## 1.1 毕业设计任务分析

毕业设计要求学生运用本科期间所学的计算机科学与技术知识，尤其是编程和软件工程领域的理论与实践技能，来规划他们感兴趣的技术发展路径。学生需要结合个人的研究问题，明确软件需求，并按照软件工程的标准流程，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中的重要实践环节，具有深远的教育意义和实际应用价值。通过毕业设计，学生能够综合运用所学知识与技能，解决实际问题，巩固和深化对专业知识的理解与掌握。这一过程不仅锻炼了学生的编程、系统设计、项目管理和问题解决能力，还培养了他们的专业素养、创新思维和职业精神。毕业设计鼓励学生探索新技术、新方法，激发他们的创新思维和自主研究能力。同时，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。团队合作是毕业设计的常态，这有助于学生学习团队协作和沟通能力。毕业设计模拟了真实工作环境中的项目开发流程，为学生进入职场打下基础。对于有意向深造的学生，毕业设计是进行学术研究的初步尝试，为未来的研究生学习或学术生涯奠定基础。撰写开发文档和毕业论文培养了学生的文档撰写和表达能力。毕业设计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。毕业设计通过项目实施解决现实问题，为社会带来积极影响和价值。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全面检验，为学生未来职业生涯做好重要准备，具有重要的教育意义和实际应用价值。

## 1.2 研学计划

作为计算机科学与技术专业的本科生，毕业设计的完成是我学业旅程中的一个重要里程碑。这项作品不仅是对所学知识的一次全面回顾和总结，更是展示我掌握核心课程内容和实际技能的舞台。

在我的毕业设计中，我计划将面向对象编程、数据结构与算法、操作系统基础以及软件工程等核心课程的理论知识转化为实践应用。过去，这些课程的理论部分对我而言显得有些遥远，主要是因为它们过于理论化，缺乏与实践的

结合。通过与导师的深入讨论，我们一致认为，将这些关键知识点应用于实际项目中，将极大地加深我的理论理解并提升我的技术实践能力。

因此，我视毕业设计为一个宝贵的机会，它不仅要求我将大学期间学到的理论知识进行综合运用，还促使我去探索和掌握当前技术领域的新趋势。在全面而深入地理解了计算机软硬件的体系结构之后，我将着手撰写毕业论文，这将是我毕业设计的核心成果。

对于我们这些即将步入国家现代化建设行列的计算机专业学生来说，对计算机系统的深刻理解是至关重要的。这将使我们在众多专业毕业生中脱颖而出，成为我们独特的竞争优势。在他人眼中，作为计算机专业的毕业生，我们对计算机系统的理解不能仅仅停留在表面，而应该努力深入探索其内在本质。同样，对于任何技术，我们都应该力求理解其深层结构和基础原理。

## 1.3 研究方法

### 1.3.1 文献法

文献法，作为一种历史悠久且持续发展的学术研究手段，其核心在于对各类文献资料的搜集、评估、整理和分析。通过这一过程，研究者能够对特定历史时期或社会现象进行深入的描述、评论，并探究其成因，进而可能孕育出新的见解或理论。

文献法的实施是一个既继承传统又不断批判创新的过程。其目标是通过系统化的文献检索、搜集、评估与应用，对某一时代或社会教育现象的特定特征进行描述和分析，以期揭示其背后的客观动因，并可能在此基础上形成新的认识或理论。

文献法的信息来源极为广泛，不仅包括书籍、期刊文章、报纸、杂志、统计公告、年鉴、信息通报、政策法规文件、学术论文和调查报告等，而且要求研究者对这些文献资料的真实性与适用性进行严格审查，以确保研究的全面性和可信度。

高质量的文献资料应具备真实性、创新性和准确性。文献研究的一般流程包括：确定研究课题或假设、制定研究计划、搜集相关文献、对文献进行整理分类以及撰写文献综述。确定研究课题或假设，即是根据现有理论、事实和需求，对现有文献进行深入分析、整理或重新分类，以形成新的研究思路。

### 1.3.2 模型研究法

模型研究法，亦称为模拟法，是一种科学探究手段，它通过构建与实际对象相似的模型来进行间接研究。这种方法特别适用于那些直接研究存在困难或不可能的系统。研究者首先需要搜集关于原型的各类信息，包括测试结果、实验数据和调查资料，以理解其属性、结构和功能。

在对这些信息进行综合分析之后，研究者运用创造性思维和逻辑推理来构建模型。随后，对所建模型进行细致的观察、实验和分析研究。最终，研究者将从模型中得到的结论推广应用到实际原型上。

模型研究法可以采用多种形式，如物理模拟、数学模拟、功能模拟和智能模拟等，具体选择哪种方法取决于研究对象的特性、复杂度和研究目的。在进行模型分析时，需要运用多种方法和技术，包括系统分析、系统综合、结构分析、要素分析、功能分析、优化分析，以及计算机运算技术和逻辑推理方法等。

作为一种在理论研究和工程研究中广泛应用的工具，模型研究法通过简化和理想化的方式，再现原型的复杂结构、功能和相互作用，成为连接理论与实践的纽带。此外，模型本身也是一种特殊的假设形式，它需要在实际应用中不断接受验证，并根据实践经验进行调整和完善。

模型研究法的主要步骤包括：收集原型信息、构建模型、对模型进行观察和实验、对模型进行分析研究，以及将研究结果推广到原型。在建模过程中，应确保模型具备真实性、简洁性（便于数学处理）、完整性（包含所有基本要素）和规范性等特性。

### 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. Web 平台和客户端技术概述

## 2.1 HTML

HTML（HyperText Markup Language，超文本标记语言）是一种标准标记语言，用于创建网页和网页应用程序的结构。它由一系列标签组成，这些标签描述了页面的结构、文本、图像、链接等元素，并告诉浏览器如何显示这些内容。HTML 的基本任务是定义和组织网页内容，为用户提供一致的浏览体验。

一个典型的 HTML 文档以<!DOCTYPE html>声明开始，紧接着是<html>根元素，包含整个 HTML 文档的内容。在<html>中，有<head>部分和<body>部分。<head>包含元数据，例如文档标题、字符集声明、样式表链接等信息。而<body>包含了文档的实际内容，如文本、图像、链接等。

HTML 使用各种标签来定义不同类型的内容和结构。例如，<h1>至<h6>标签用于定义标题，<p>标签用于定义段落，<a>标签用于定义超链接等。此外，HTML 还可以包含属性，这些属性提供了关于元素的更多信息。例如，<a>标签可以具有 href 属性指定链接的 URL，<img>标签可以具有 src 属性指定图像的源文件等。

HTML5 是 HTML 的最新版本，它引入了许多新元素和属性，以增强多媒体支持和语义结构。例如，引入了诸如<video>和<audio>等多媒体标签，以及语义化的标签如<header>、<footer>、<article>等。

总之，HTML 是 Web 开发的基础，理解并掌握 HTML 对于任何 Web 开发者来说都是至关重要的。通过使用 HTML，开发者可以创建丰富多样的网页内容，从而为用户提供更加丰富、多样化的互联网体验。

## 2.2 CSS

CSS（Cascading Style Sheets，层叠样式表）是一种用于描述网页或应用

程序外观和样式的语言。HTML 负责定义网页的结构和内容，而 CSS 则负责控制这些内容的表现方式。它使开发者能够将样式应用于 HTML 和 XML 等标记语言创建的内容，从而实现页面的美观和一致性。

CSS 的工作原理是通过将样式规则应用于 HTML 文档中的元素来改变它们的外观。每个 CSS 规则由一个选择器和一组声明块组成。选择器用于选择要应用样式的 HTML 元素，而声明块则包含一个或多个属性-值对，用于指定元素的具体样式。

CSS 具有层叠（cascading）的特性，这意味着多个样式来源时，如浏览器默认样式、用户样式表和开发者样式表，会按照一定的优先级和特定的层叠顺序来确定最终应用的样式。

CSS 可以控制元素的各种视觉属性，包括文字颜色、背景颜色、边框样式、尺寸和位置等。此外，CSS 还支持响应式设计，使开发者能够根据设备的不同或视口大小动态调整页面的布局和样式。

随着 CSS 的发展，出现了许多新的功能和技术，如 CSS 网格布局（Grid Layout）、弹性布局（Flexbox）、CSS 变量（Variables）等，这些功能使得页面布局更加灵活和复杂的样式更易管理。

总之，CSS 是现代 Web 设计和开发中不可或缺的一部分。通过合理使用 CSS，开发者可以实现各种视觉设计效果，提升用户体验和页面性能。同时，深入理解 CSS 的工作原理和最佳实践，有助于开发者更高效地管理和维护复杂的 Web 项目。

## 2.3 JavaScript

JavaScript 是一种高级的、解释型的编程语言，主要用于在网页中实现复杂的交互功能和动态效果。它无需编译即可在浏览器中直接运行，这极大地提高了开发效率和网页的响应速度。JavaScript 的核心特性包括动态性、事件驱动、异步编程和文档对象模型（DOM）操作，这些特性使得它成为现代 Web 开发中不可或缺的一部分。

JavaScript 是一种动态类型的语言，这意味着在声明变量时不需要指定其类型，变量的类型会在运行时根据赋值自动确定。这种动态性使得 JavaScript 在处理数据时更加灵活，开发者可以根据需要随时改变变量的类型。

JavaScript 是事件驱动的，这意味着它不会主动执行代码，而是根据用户的交互操作或其他触发事件来执行相应的代码。例如，当用户点击一个按钮时，JavaScript 可以捕获这个点击事件并执行相应的函数。这种事件驱动的方式使得 JavaScript 能够实现丰富的交互效果。

异步编程是 JavaScript 的重要特性之一。在 Web 开发中，很多操作如网络请求、文件读写等都是异步的，需要等待一段时间才能完成。JavaScript 通过回调函数、Promise、async/await 等机制来实现异步编程，这使得开发者可以编写非阻塞的代码，提高页面的响应性和用户体验。

JavaScript 可以通过 DOM API 来操作网页中的元素，包括添加、删除、修改元素的内容和属性等。这使得开发者可以在网页加载后根据需要对页面进行动态修改，实现各种复杂的交互效果和动态页面效果。

总结来说，JavaScript 是一种功能强大的编程语言，它在 Web 开发中扮演着至关重要的角色。通过其动态性、事件驱动、异步编程和文档对象模型（DOM）操作等核心特性，JavaScript 不断推动着互联网技术的发展，为开发者提供了丰富的工具和手段来创建交互性强、功能丰富的网页和应用程序。

## 2.4 Web 平台和 Web 编程

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[1]。

Web 平台，作为互联网基础设施的关键构成，为用户通过浏览器提供了一个易于访问且功能丰富的环境，使用户能够直接利用各种在线服务和应用程序。这些平台的建设基于一系列开放的网络标准和协议，确保了跨不同平台的兼容性与可扩展性。

Web 平台的架构通常基于客户端-服务器模型，客户端主要负责展示界面和响应用户操作，而服务器端则承担业务逻辑处理、数据存储和 API 服务提供的任务。这种分离式架构不仅提升了应用的维护性，还使 Web 平台能够灵活适应多样化的用户需求和设备。

随着技术的不断进步，Web 平台已经由最初的静态网页发展为能够支持复杂交互和动态内容的动态应用。现代 Web 平台通过 Ajax、WebSocket 等技术实现了与服务器的即时通信，为用户提供了类似本地应用的体验。此外，Web Assembly 等新兴技术进一步增强了 Web 平台的功能，使得浏览器能够执行接近原生性能的代码。

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。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[3]。

## 2.5 软件开发的过程管理

### 2.5 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[4]。

作为本科专业学生的毕业设计软件项目，其复杂度远超一般单一功能的应用程序。该项目涉及大量的手写代码，其工作量之大不是一般小型程序所能比拟的，从问题的分析到初步的编码实现，整个过程不可能在短时间内迅速完成。因此，该项目应被视为一项系统工程，需要采用软件工程的管理方法来规范整个开发流程。

在软件工程的开发模式中，可以选择两种经典的模型：传统的瀑布模型和现代的增量式迭代模型。无论选择哪种模型，软件开发都需要经过分析、设计、实施和测试这四个基本阶段。

在当前开放的软件开发环境中，开发者在开发过程中持续进行设计优化和代码重构，以不断提高程序的功能和代码质量。基于这些考虑，本项目选择了增量式迭代模型作为其开发策略。这种模型允许开发者在开发过程中逐步完善软件，通过一系列的小步迭代来逐步构建和改进产品。

如下图 2-1 增量式迭代模型所示：

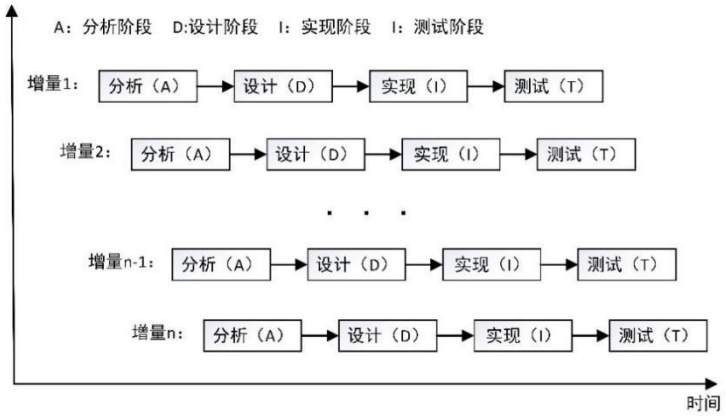


**图 2-1 增量式迭代模型**

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

产品开发。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[5]。

2.3 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。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[6]。

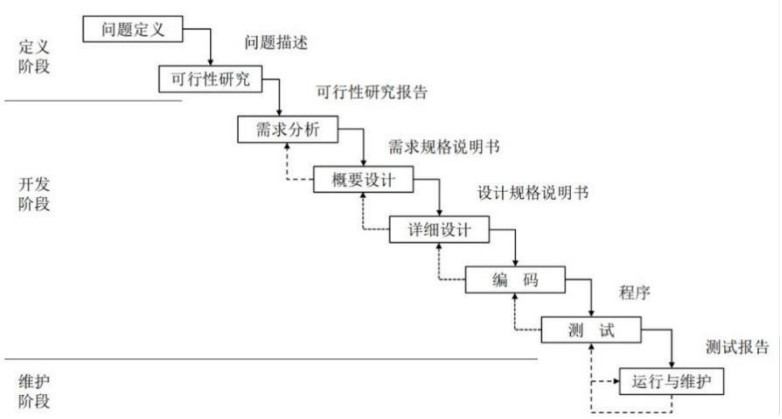


**图 2-2 瀑布式开发模型**

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[7]。

# 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[8]。

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 所示：

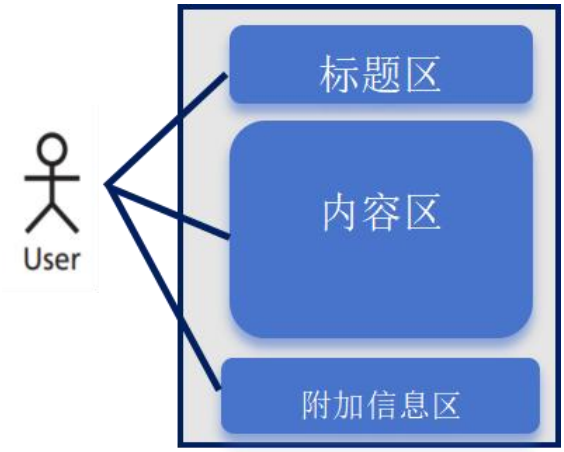


**图 3-1 第一次增量迭代用例图**

### 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 3-2 项目 Dom 树所示：

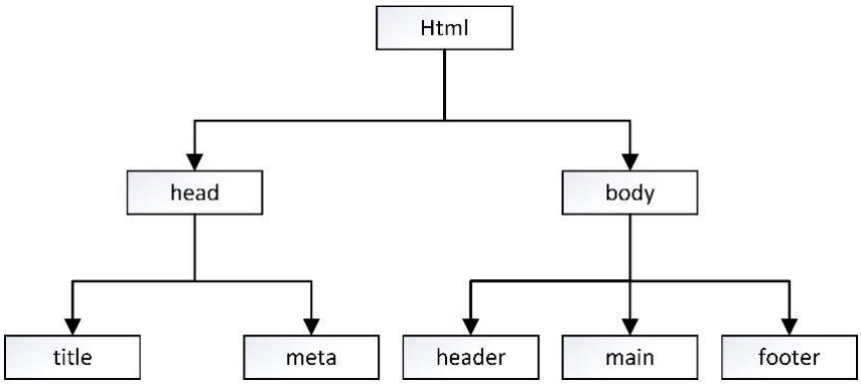


**图 3-2 项目 Dom 树**

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

```
<body>
    <header>
        作品展示
    </header>
    <main>
        内容展示区
    </main>
    <footer>
        © 张荣军 江西科技师范大学 2024-2025
    </footer>
</body>
```

### 3.2.2 CSS 代码

```
* {
    margin: px;
    text-align: center;
    align-content: center;
    justify-content: center;
    font-size: px;
    font-family: 'KaiTi', '楷体', 'SimKai', 'STKaiti', serif;
}
header {
```

```
    border: px solid black;
    height: px;
}
main {
    border: px solid black;
    height: px;
}
footer {
    border: px solid black;
    height: px;
}
a {
    display: inline-block;
    padding: px;
    color: black;
    box-shadow: px px px px rgba(, , , );
    background-color: #ECFC;
    background-image: linear-gradient(deg, #ECFC %, #ECFC %);
    text-decoration: none;
}
```

## 3.3 运行与测试

在在 PC 端现代浏览器（Edge）里面打开本地的文件，运行代码查看效果图如下图 3-3 所示：

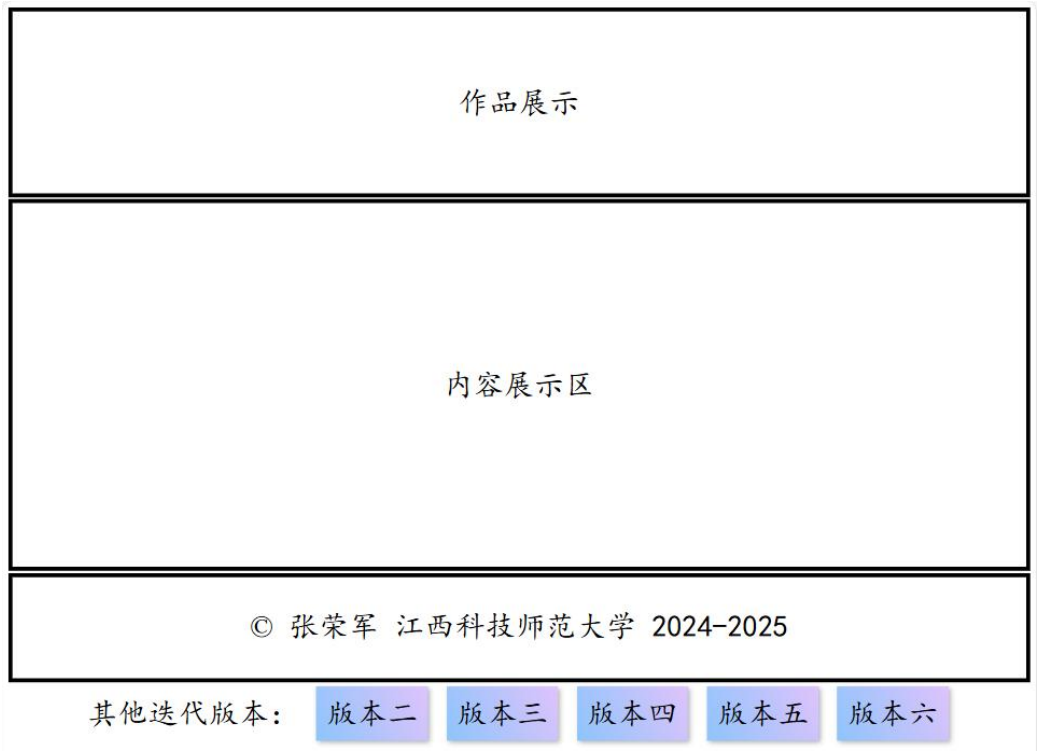


图 3-3 PC 页面效果图

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。

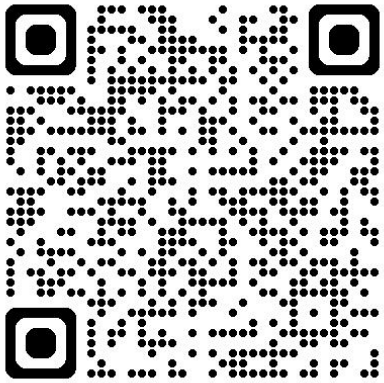

**图 3-4 阶段 1 增量迭代二维码**

## 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：

```
cd /WebUIProject
# cd 后面输入要进入的项目文件夹
git init
# init 初始化本地代码仓库，用于代码版本控制
git status
# status 查看仓库代码状态
git add <phase1.html>
# add 后面输入要跟踪的文件名字
git commit -m "commit message"
# commit 后面输入本次提交代码的备注信息
git log
# 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 39625b55dc64e2bddfd610bb21f22641ca9f6703 (HEAD -> master)
Author: 20213623张荣军@科师大 <314744262@qq.com>
Date:   Fri Jun 14 16:46:44 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.html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**

# 4. 响应式设计与适应屏幕实现

## 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 所示：

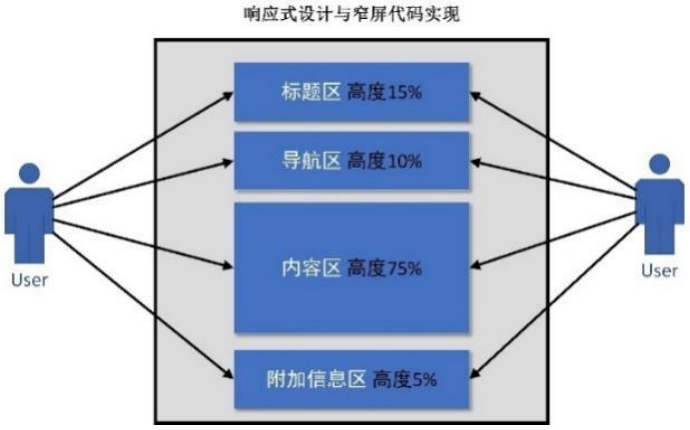


**图 4-1 第二次增量迭代项目用例图**

### 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：

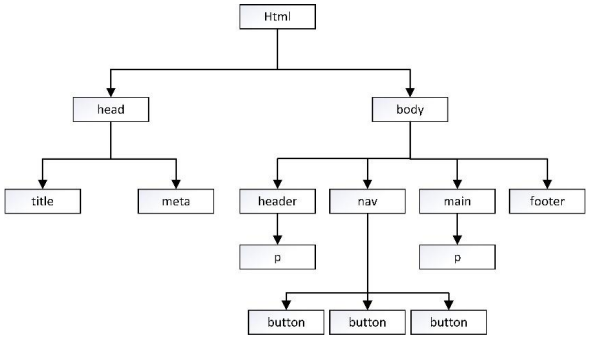


**图 4-2 项目 Dom 树**

## 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>
© 张荣军 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,#FFDEE90%,#B5FFFC100%);
    box-shadow:2px3px5px0pxrgba(0,0,0,0.4);
}
nav{
    background-color:rgb(247,247,247);
    border:2pxsolidblack;
    height:10%;
}
navbutton{
    font-size:0.8em;
    padding:13px13px;
    box-shadow:2px2px5px0pxrgba(0,0,0,0.3);
    background-color:#8EC5FC;
    background-image:linear-gradient(62deg,#8EC5FC0%,#E0C3FC100%);
    border-radius:8px;
```

```
  cursor:pointer;
  transition:background-color0.3sease;
}
footer{
  background-color:#FAD961;
  background-image:linear-gradient(90deg,#FAD9610%,#F76B1C100%);
  border:2pxsolidblack;
  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，分别在 iPhone SE 和 Samsung Galaxy S8+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。效果对比图如下图 4-3 所示：


**图 4-3 不同设备运行效果对比图**

响应式设计与适应屏幕实现在 PC 端现代浏览器（Edge）中的效果图如下图 4-4 所示：


**图 4-4 PC 端页面效果图**

分析图 4-4 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 4-5 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 4-5 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的运行效果。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 4-6 提交代码图所示：

```
commit dfafca0944b6743a949bdc82b049ccb165482b91 (HEAD -> master)
Author: 20213623张荣军@科师大 <314744262@qq.com>
Date:   Fri Jun 14 16:46:44 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 4-6 提交代码图**

# 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 所示：

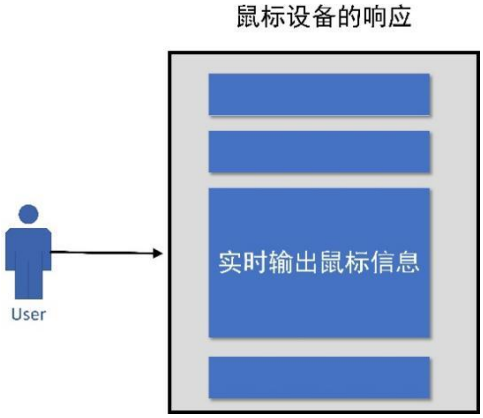


**图 5-1 第三次增量迭代用例图**

### 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：

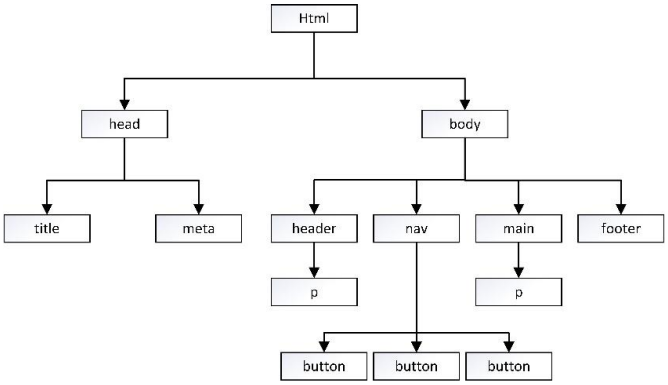


**图 5-2 项目 Dom 树所示**

## 5. 2 代码实现

### 5. 2. 1 HTML 代码

```
<body>
  <header>
    <p id="head">
      作品展示
    </p>
  </header>
  <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>
    © 张荣军 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：

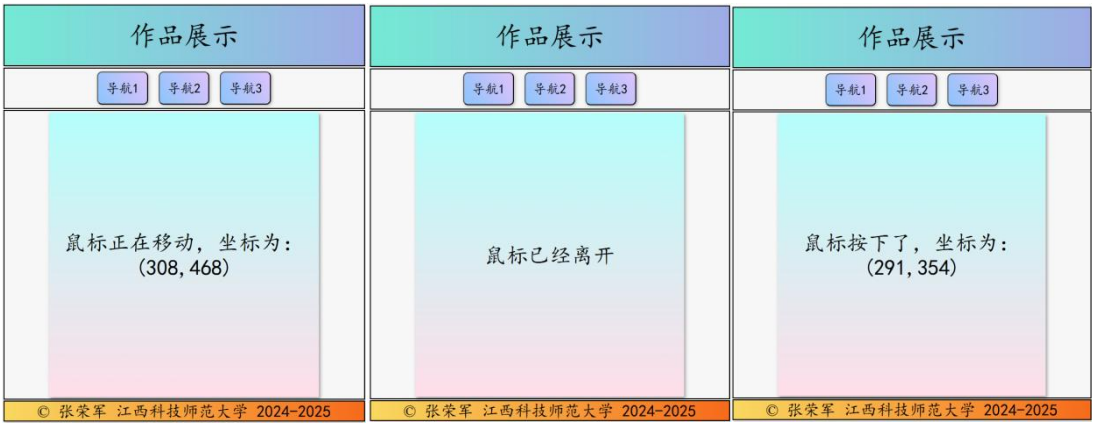


**图 5-3 页面效果图**

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。

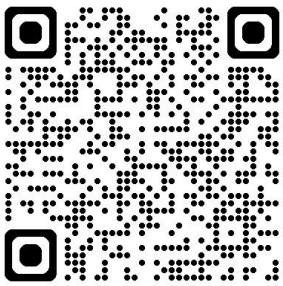

**图 5-4 阶段 3 增量迭代二维码**

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 a81dd27b672cc91246c24036c5136b080b42bd62 (HEAD -> master)
Author: 20213623张荣军@科师大 <314744262@qq.com>
Date:   Fri Jun 14 16:46:44 2024 +0800

    第三次增量迭代--本次提交了'个性化化交互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**

# 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。

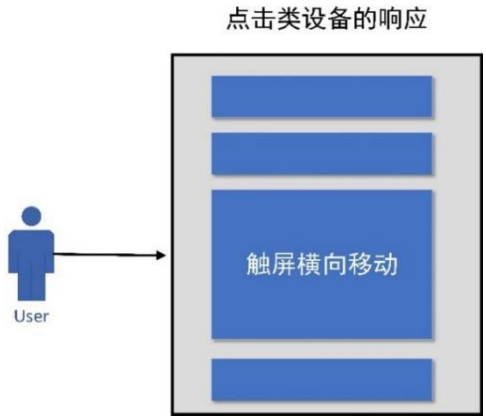


**图 6-1 第 4 次增量用例图**

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：

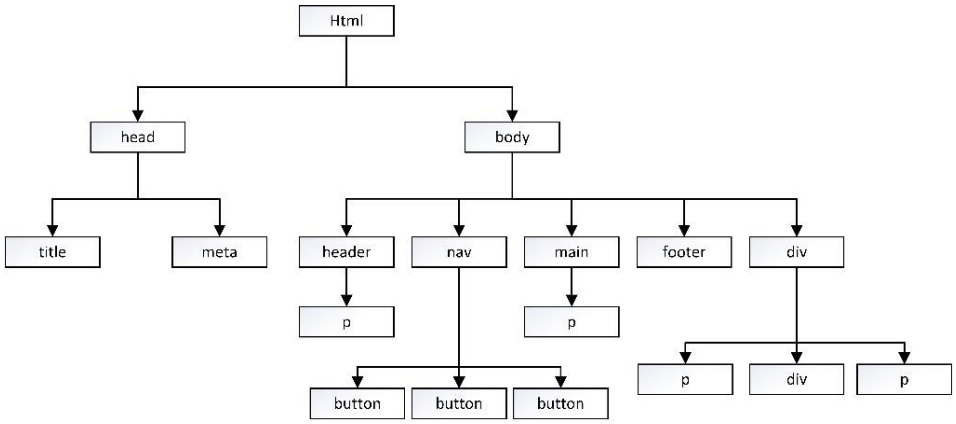


**图 6-2 项目 Dom 树**

## 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>
    © 张荣军 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">
           
        </p>
      </div>
      <p id="displayCode">
         
      </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 = "鼠标里开!";
  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。下面分别展示在 iPhone SE 和 Samsung Galaxy S8+这两种设备上运行本项目的效果，如下图 6-3 所示：


**图 6-3 不同设备运行效果对比图**

探索 UI 拖动模拟触屏的操作模式在 PC 端现代浏览器（Edge）中的效果图如下图 6-4 所示：


**图 6-4 PC 端页面效果图**

分析图 6-4 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 6-5 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。

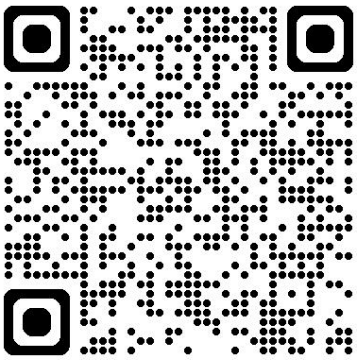

**图 6-5 阶段 4 增量迭代二维码**

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 6-6 提交代码图所示：

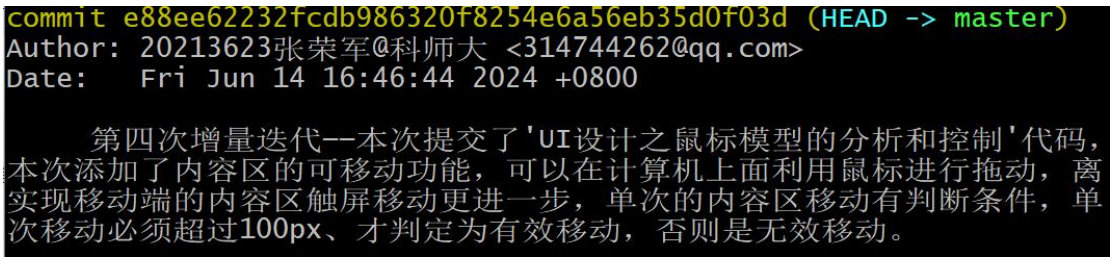


**图 6-6 提交代码**

# 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：

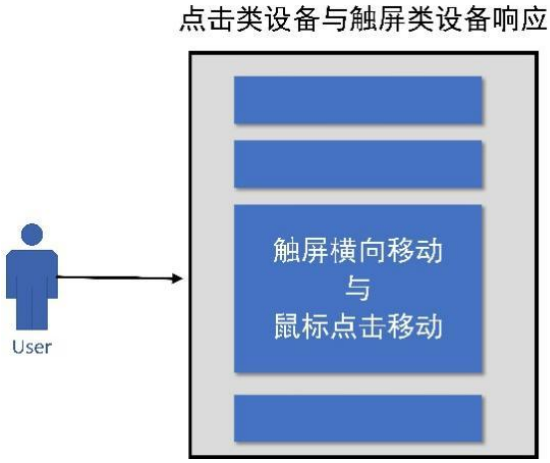


**图 7-1 第 5 次增量项目用例图**

### 7.1.2 功能设计

第 5 次迭代更新代码主要实现了了一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：

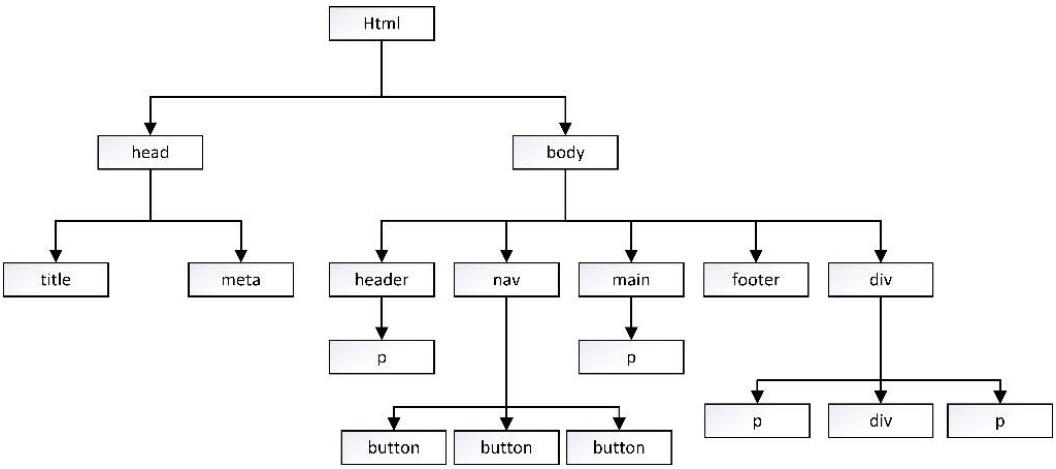


**图 7-2 项目 Dom 树**

## 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>
    © 张荣军 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">
           
        </p>
      </div>
      <p id="displayCode">
         
      </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。下面分别展示在 iPhone SE 和 Samsung Galaxy S8+这两种设备上运行本项目的效果，如下图 7-3 所示：


**图 7-3 不同设备运行效果对比图**

通用 UI 设计为触屏和鼠标统一建模在 PC 端现代浏览器（Edge）中的效果图如下图 7-4 所示：

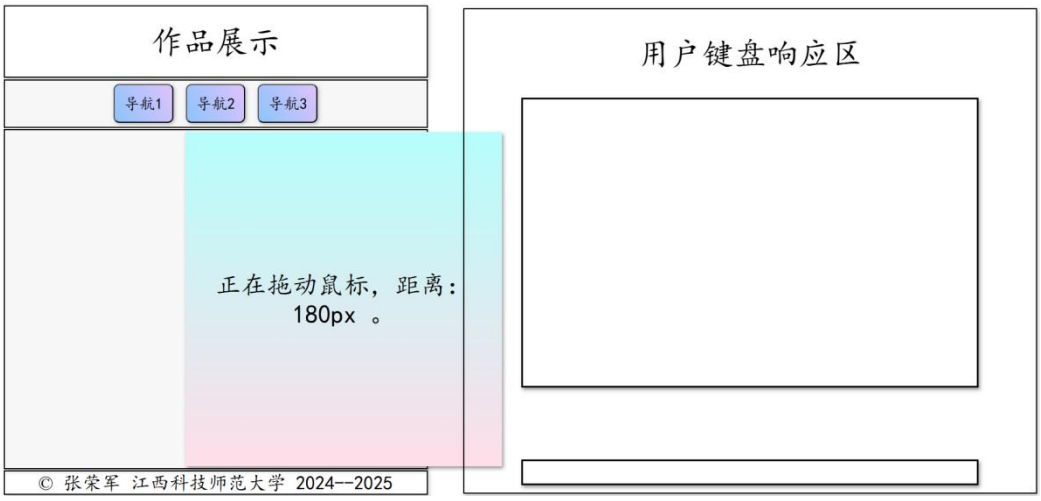

**图 7-4 PC 端页面效果图**

分析图 7-4 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 7-5 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 7-5 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 7-6 提交代码图所示：

```
commit e14440e90258c79d3f4bcff6212d6f25b269177e (HEAD -> master)
Author: 20213623张荣军@科师大 <314744262@qq.com>
Date:   Fri Jun 14 16:46:44 2024 +0800

    第五次增量迭代--本次提交了'通用的UI设计，用一套代码同时为触屏和鼠标建模'代码，本次将JS代码整合在一个全局对象Pointer中，这个对象中存储了本程序的所有异步代码，这个对象在本程序的JS里面可以任意调用，简化了代码的复杂度，优化了代码的重复使用性。这样可以减小浏览器的响应时间，提供用户更好的使用体验。
```

**图 7-6 提交代码**

# 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。

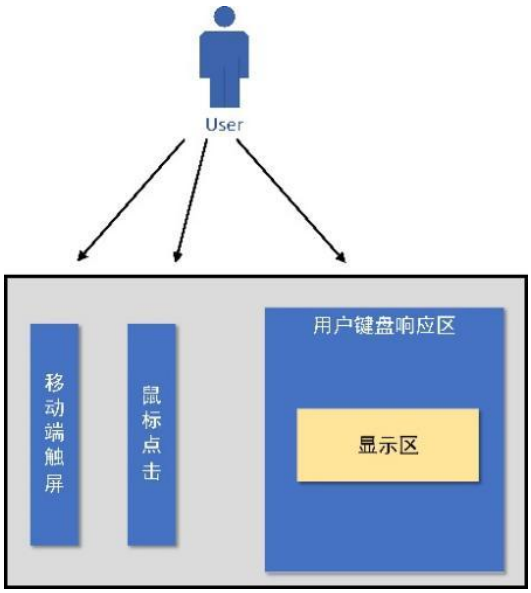


**图 8.1 键盘响应用例图**

### 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：

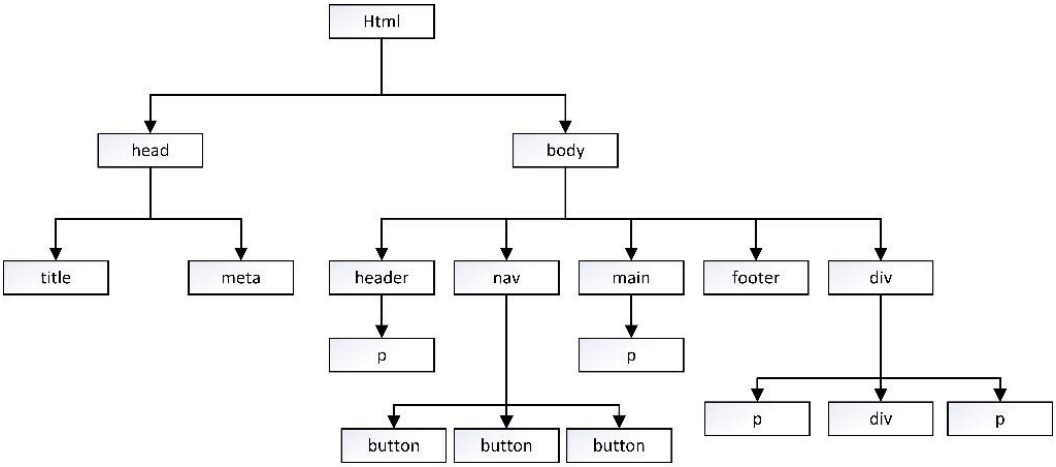


**图 8-2 项目 Dom 树**

## 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>
    © 张荣军 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">
           
        </p>
```

```
      </div>
      <p id="displayCode">
         
      </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 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。个性化 UI 监控键盘实现在 PC 端现代浏览器（Edge）中的效果图如下图 8-3 所示：

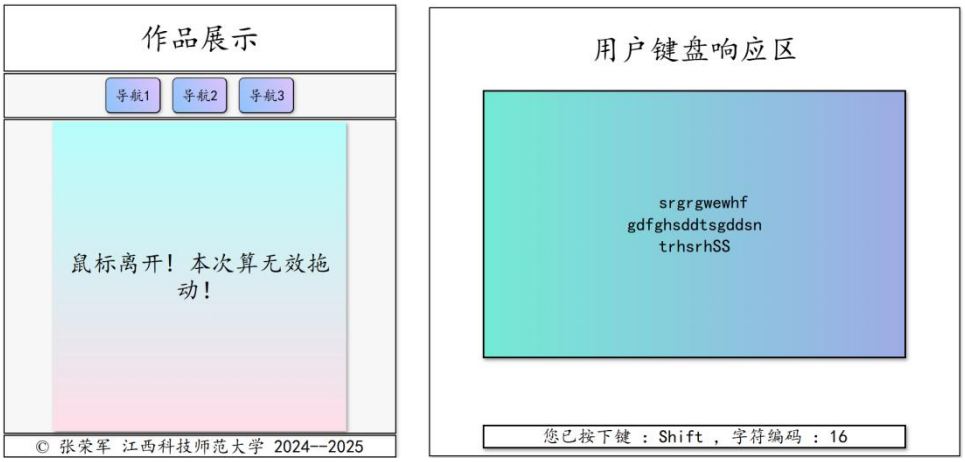


图 8-3 PC 页面效果图

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 9475d190f6df2decab6aab907d3662ccb1097409 (HEAD -> master)
Author: 20213623张荣军@科师大 <314744262@qq.com>
Date:   Fri Jun 14 16:46:44 2024 +0800

    第六次增量迭代一本次提交了'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

# 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[9]。

## 9.2 通过 Git Hub 平台实现本项目的全球域名。

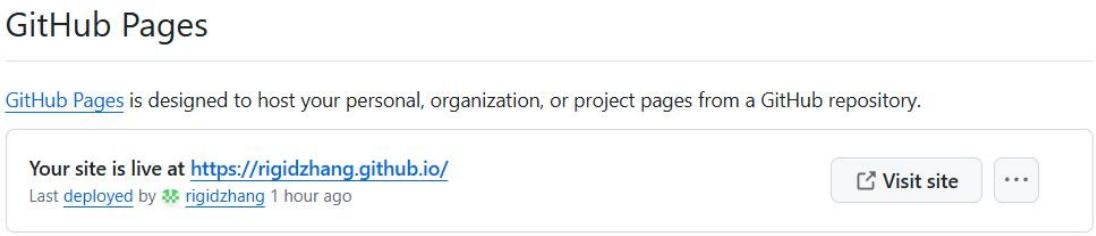


**图 9-1 本项目的全球域名**

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

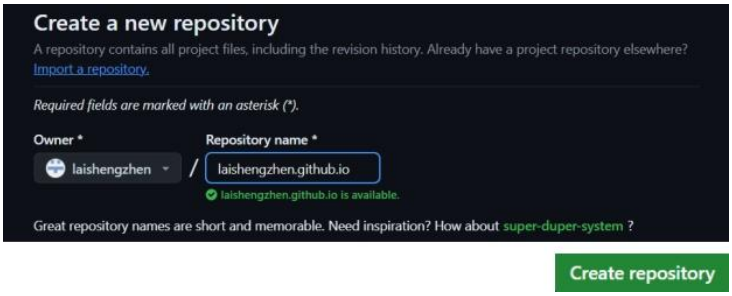


**图 9.2 创建空的远程代码仓库**

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/laishengzhen/laishengzhen.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：

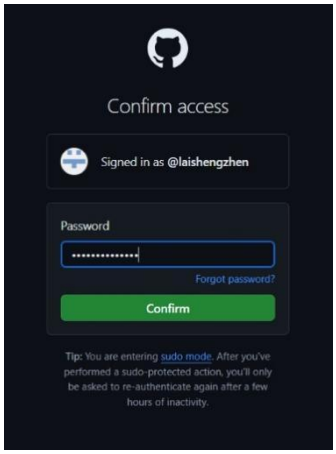


**图 9.3 开发者授权**

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：

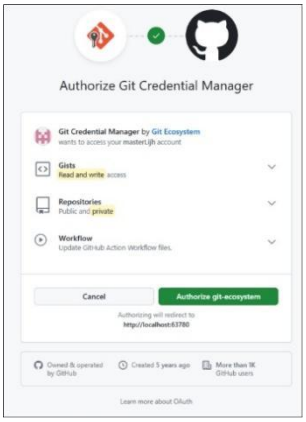


**图 9.4 再次确认授权**

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：

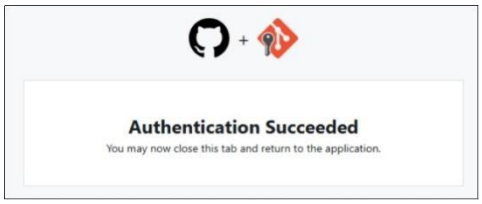


**图 9.5 成功实现远程连接**

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：

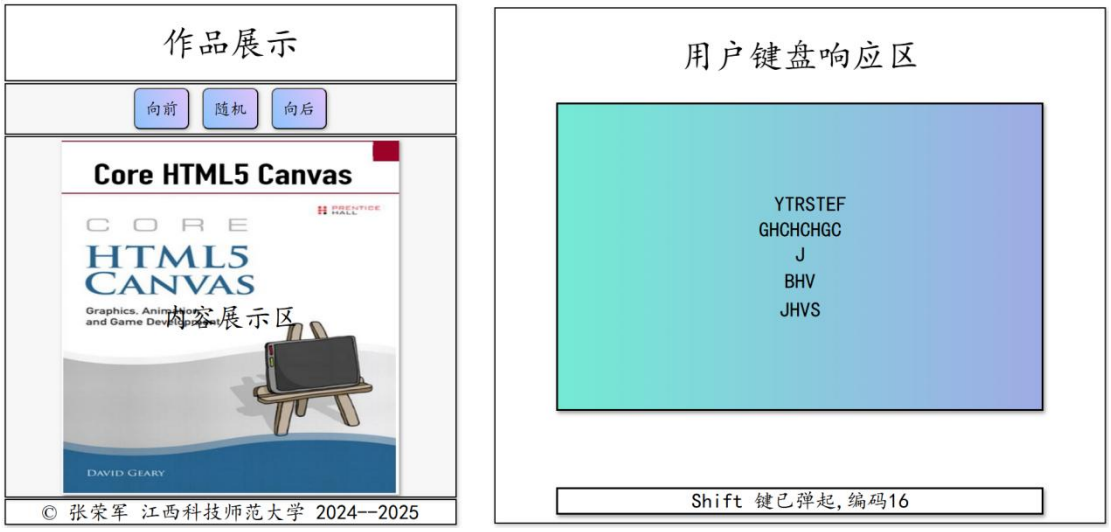


**图 9.6 项目在互联网的部署**

# 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.