成 绩：

江 西 科 技 师 范 大 学

课程设计（论文）

题目（中文）：基于 Web 客户端技术的个性化 UI 的设计和编程

（外文）：Customized UI design and Programming based onWeb 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：俞森洲

学　　号：20213663

指导教师：李健宏

2024 年 6 月 18 日

目录

摘要

随着互联网技术的快速发展，用户对网页界面的个性化需求日益增长。本文旨在探究基于 Web 客户端技术的个性化用户界面（UI）设计与编程方法，以提升用户体验和满足多样化的用户需求。本文首先界定了个性化 UI 的概念与特点，并分析了其在现代 Web 应用中的重要性。随后，本文详细阐述了 Web 客户端技术的基本框架，包括前端开发环境、浏览器渲染机制以及网络协议等关键元素。在此基础上，本文提出了个性化 UI 设计的原则，探讨了设计流程与实施策略，并通过案例分析展示了这些原则与策略的应用。在编程实践部分，本文介绍了 HTML5、CSS3、JavaScript 及其框架和库的使用，并讨论了响应式布局、数据驱动的动态 UI 生成、交互性增强及性能优化等关键技术点。最后，本文通过实例分析，验证了所提出的设计理念和编程方法的有效性，并针对未来的研究方向提出了展望。本文的研究不仅为 Web 开发者提供了一套完整的个性化 UI 设计与实现方案，同时也为相关领域的研究提供了理论支持与实践指导。

关键词：Web 客户端技术；个性化 UI 设计；编程方法；用户体验；前端开发

Abstract: With the rapid development of Internet technology, users' demand for personalized web interfaces is increasing. This paper aims to explore the design and programming methods of personalized user interface (UI) based on Web client technology, in order to improve user experience and meet diverse needs. This paper first defines the concept and characteristics of personalized UI, and analyzes its importance in modern Web applications. Subsequently, this paper elaborates on the basic framework of Web client technology, including key elements such as front-end development environment, browser rendering mechanism, and network protocols. On this basis, this paper proposes the principles of personalized UI design, discusses the design process and implementation strategies, and demonstrates the application of these principles and strategies through case analysis. In the part of programming practice, this paper introduces the use of HTML5, CSS3, JavaScript and its frameworks and libraries, and discusses key technical points such as responsive layout, data-driven dynamic UI generation, interaction enhancement and performance optimization. Finally, through the analysis of examples, this paper verifies the effectiveness of the proposed design concepts and programming methods, and puts forward prospects for future research directions. The research in this paper not only provides a complete set of personalized UI design and implementation schemes for Web developers, but also provides theoretical support and practical guidance for research in related fields.

Keywords: Web Client Technology; Personalized UI Design; Programming Methods; User Experience; Front-End Development

1.前言

在传统的 Web 客户端应用中，用户界面设计和编程是至关重要的环节。随着 Web 技术的不断发展和进步，如今的 Web 界面设计已经不再局限于简单的静态网页，而是更注重用户体验和个性化定制。本论文将探讨基于 Web 客户端技术的个性化 UI 设计和编程，旨在为开发人员提供更加具体和实用的指导，帮助他们在开发过程中更好地满足用户需求并提升用户体验。

1.1 毕设任务分析

首先，我们将介绍个性化 UI 设计的基本概念和原则，探讨在 Web 客户端应用中如何设计出符合用户习惯和需求的个性化界面。我们将深入研究用户画像分析、用户偏好设置以及用户行为数据分析等方面，以便开发人员更好地了解用户，从而为用户提供更加个性化的界面设计。

其次，我们将探讨基于最新的 Web 前端技术，如 HTML5、CSS3 和 JavaScript 等，如何实现个性化 UI 的设计和编程。我们将介绍响应式设计、交互式元素、动画效果等技术和工具，帮助开发人员实现更加动态和个性化的 Web 界面。

最后，我们将通过实际案例和代码示例，演示如何在实际项目中应用个性化 UI 设计和编程。我们将分享一些最佳实践和经验，帮助开发人员更加高效地设计和开发具有个性化特色的 Web 客户端应用。

本论文旨在为 Web 开发人员和界面设计师提供一份全面且实用的指南，让他们能够更好地理解和应用个性化 UI 设计和编程的理念，从而为用户提供更加智能和个性化的 Web 客户端应用体验。

1.2 研学计划

本研学计划旨在通过为期六个月的系统学习与实践，深入探索基于 Web 客户端技术实现个性化 UI 设计和编程的方法。项目将从技术调研出发，覆盖设计原则、编程技能提升，最终落实到开发一个个性化 UI 的 Web 客户端原型。过程中将重点研究 HTML5、CSS3、JavaScript 框架等技术，并利用现代前端工具和组件库如 React 或 Vue. js 来高效实现界面个性化。预期成果包括掌握个性化 UI 设计的理论与实践，及开发一个经过用户体验测试优化的 UI 原型，以提升设计与开发能力，同时为后续的研究与应用奠定坚实基础。

1.3 研究方法

本研究将采用迭代式开发和用户中心设计方法，结合文献综述、案例分析和实证测试。

通过文献回顾和技术评估确立研究基础；设计和实现个性化 UI 原型，并结合用户反馈迭代优化；通过定量和定性数据分析评估原型的有效性和用户满意度，确保研究成果具有实用价值和科学性。

2. 技术总结和文件综述

2.1 Web 平台和客户端技术概述

Web 平台和客户端技术是互联网应用开发的两大支柱，它们共同构成了用户交互和数据处理的基础设施。Web 之父 Tim Berners-Lee 在发明 Web 的基本技术架构以后，就成立了 W3C 组织，该组织在 2010 年后推出的 HTML5 国际标准，结合欧洲 ECMA 组织维护的 ECMAScript 国际标准，几乎完美缔造了全球开发者实现开发平台统一的理想，直到今天，科学家与 Web 行业也还一直在致力于完善这个伟大而光荣的理想[1]。学习 Web 标准和 Web 技术，学习编写 Web 程序和应用有关工具，最终架构一套高质量代码的跨平台运行的应用，是我的毕设项目应用的技术路线。

1989 年，蒂姆·伯纳斯-李爵士发明了万维网（见原始提案）。1990 年 10 月，他创造了“万维网”一词，编写了第一台万维网服务器“httpd”和第一个客户端程序（浏览器和编辑器）“万维网。

他编写了“超文本标记语言”（HTML）的第一个版本，这是一种具有超文本链接功能的文档格式化语言，成为网络的主要发布格式。随着 Web 技术的普及，他对 URI、HTTP 和 HTML 的最初规范得到了改进，并在更大的圈子里进行了讨论。

1994 年，在许多公司的敦促下，成立了万维网联盟，向网络投入了越来越多的资源。Tim Berners-Lee 爵士开始领导网络联盟团队的重要工作，以促进一致的架构，适应网络标准的快速发展，从而构建网站、浏览器和设备，体验网络所提供的一切。

蒂姆·伯纳斯-李爵士在创建万维网联盟时创建了一个同行社区。Web 技术已经发展得如此之快，因此组建一个单一的组织来协调 Web 标准至关重要。蒂姆接受了麻省理工学院的邀请，他在联盟方面有丰富的经验，主持 W3C。他从一开始就要求 W3C 具有全球影响力。

让我们先简单介绍一下 Web，它是万维网的缩写。大多数人说“网络”而不是“万维网”，我们将遵循这一惯例。网络是一组文档，称为网页，由世界各地的计算机用户共享（大部分）。不同类型的网页做不同的事情，但至少，它们都在电脑屏幕上显示内容。所谓“内容”，我们指的是文本、图片和用户输入机制，如文本框和按钮[2]。

Web 编程是一个很大的领域，通过不同的工具实现不同类型的 Web 编程。所有的工具都

使用核心语言 HTML，所以几乎所有的网络编程书籍都在某种程度上描述了 HTML。这本教科书涵盖了 HTML5、CSS 和 JavaScript，所有这些都是深入的。众所周知，这三种技术是客户端 web 编程的支柱。使用客户端 web 编程，所有网页计算都在最终用户的计算机（客户端计算机）上执行[3]。

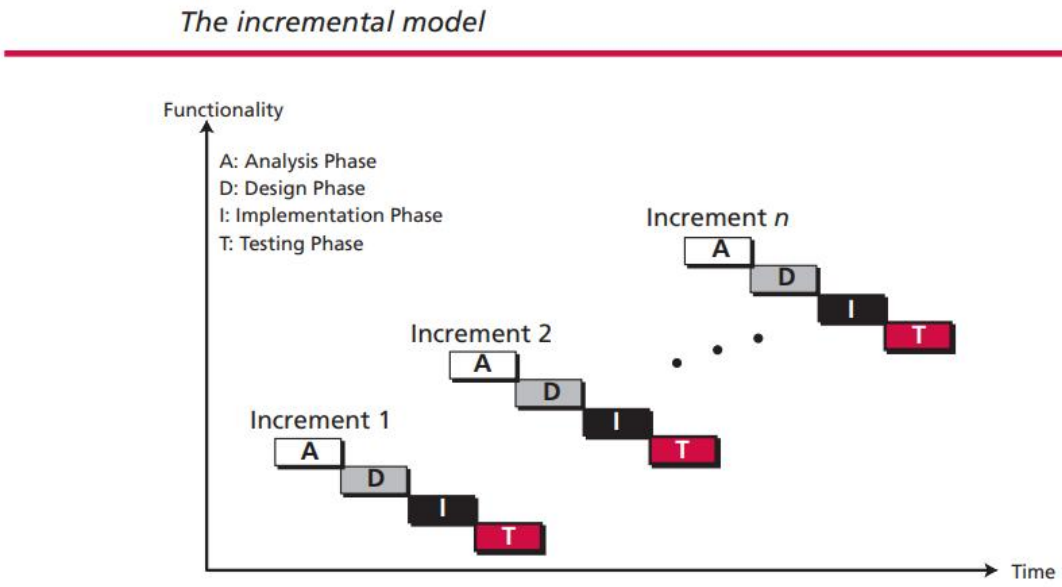
Web 应用的程序设计体系由三大语言有机组成：HTML，CSS， JavaScript。这三大语言的组合也体现了人类社会化大生产分工的智慧，可以看作用三套相对独立体系实现了对一个信息系统的描述和控制，可以总结为：HTML 用来描述结构（Structure）、CSS 用来描述外表（presentation）、Javascript 用来描述行为（Behavior）[4]；这也可以用经典的 MVC 设计模式来理解 Web 平台架构的三大基石，Model 可以理解为 HTML 标记语言建模，View 可以理解为用 CSS 语言来实现外观，Controller 则可理解为用 JavaScript 结合前面二个层次，实现了在微观和功能层面的代码控制。

2.2 项目的增量式迭代开发模式

本项目作为一个本科专业学生毕业设计的软件作品，与单一用途的程序相比较为复杂，本项目所涉及的手写代码量远超过简单一二个数量级以上，从分析问题的到初步尝试写代码也不是能在几天内能落实的，可以说本项目是一个系统工程，因此需要从软件工程的管理视角来看待和规范项目的编写过程。

而本项目考虑选择的软件工程开发过程管理模式有两种经典模型：瀑布模型（The waterfall model）和增量式迭代模型(The incremental model)。而任何开发模式则都必须同样经历四个阶段：分析（Analysis）、设计（Design）、实施（Implementation）、测试（test）。

瀑布模型需要专业团队完美的配合，从分析、设计到实施，最后到测试，任何阶段的开始必须基于上一阶段的完美结束。而这对于我们大多数普通开发者是不太现实的，作为小微开发者由于身兼数职，其实无法 1 次就能完美完成任何阶段的工作，比如在实施过程中，开发者会发现前面的设计存在问题，则必须在下一次迭代项目时改良设计。在当今开源的软件开发环境中，开发者在软件的开发中总是在不断地优化设计、重构代码，持续改进程序的功能和代码质量。因此在本项目的开发中，也采用了增量模型的开发模式[5]。本项目中我一共做了六次项目的开发迭代，如下图 2-1 所示：



在增量模型中，软件是按一系列步骤开发的。开发人员首先完成了整个系统的简化版本。此版本表示整个系统，但不包括详细信息。图显示了增量模型的概念。

在第二个版本中，添加了更多的细节，而有些细节尚未完成，系统将再次进行测试。如果出现问题，开发人员就会知道问题出在新功能上。在现有系统正常工作之前，它们不会添加更多功能。此过程一直持续到添加了所有必需的功能为止[5]。

3. 内容设计概要

3.1 分析和设计

这一步是项目的初次开发，本项目最初使用人们习惯的“三段论”式简洁方式开展内容设计，首先用一个标题性信息展示 logo 或文字标题，吸引用户的注意力，迅速表达主题；然后展现主要区域，也就是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的重点；最后则是足部的附加信息，用来显示一些用户可能关心的细节变化。如图 3-1 用例图所示：

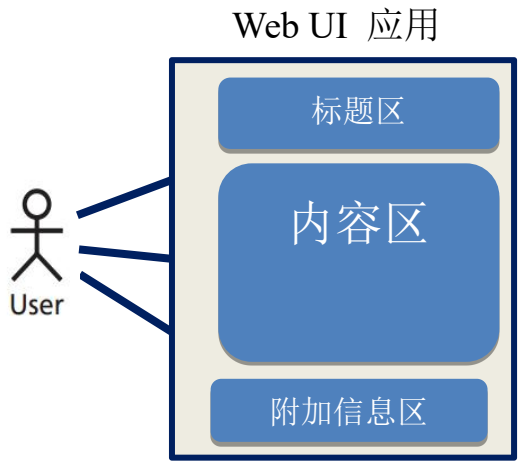


图 3-1 用例图

3.2 项目的实现和编程

一、HTML 代码编写如下：

```
</head>
<body>
    <header>
        <p id="book">
            《俞森洲的毕设题目》
        </p>
     </header>
      <nav>
      <button>Prev</button>
      <button>Other</button>
      <button>Next</button>
      </nav>

      <main id="main">
      <div id="bookface">
      这是书的封面图<br>
      在此对象范围拖动鼠标/滑动触屏<br>
          拖动/滑动超过 100 像素，视为有效 UI 互动！
        </div>
      </main>
      <footer>
       CopyRight 俞森洲 江西科技师范大学 2021--2025
      </footer>
```

二、CSS 代码编写如下：

```
  *{
    margin: 10px;
    text-align: center;
    font-size:30px ;
   }
    header{
      border: 2px solid blue;
      height: 200px;
       }
```

```
 main{
    border: 2px solid blue;
    height: 400px;
 }
 footer{
    border: 2px solid blue;
    height: 100px;
 }
a{
 display: inline-block ;
 padding:10px ;
 color: white;
 background-color: blue;
 text-decoration: none ;
}
```

3.3 项目的运行和测试

项目的运行和测试至少要通过二类终端，本文此处仅给出 PC 端用 Chrome 浏览器打开项目的结果，如下图 3-2 所示。由于本项目的阶段性文件已经上传 github 网站，移动端用户可以通过扫描图 4-3 的二维码，运行测试本项目的第一次开发的阶段性效果。

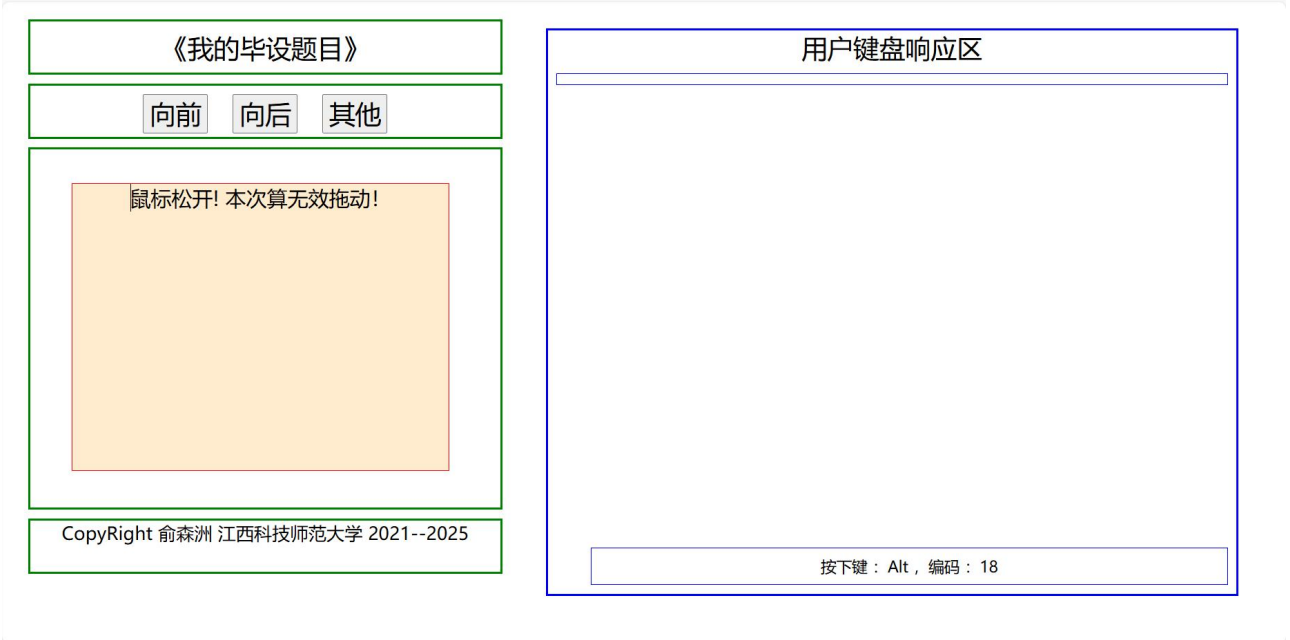


图 3-2 PC 端运行效果图

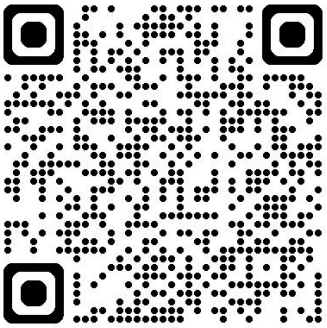


图 4-3 移动端二维码

3.4 项目的代码提交和版本管理

本项目的文件通过 gitBash 工具管理，作为项目的第一次迭代，在代码提交和版本管理环节，我们的目标是建立项目的基本文件结构，还有设置好代码仓库的基本信息：如开发者的名字和电子邮件。

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ mkdir webUI
$ cd webUI
$ git init
$ git config user.name 江科师大俞森洲
$ git config user.email marsterlijh@jxstnu.edu.cn
$ touch index.html myCss.css
```

编写好 index. html 和 myCss. css 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add index.html myCss.css
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
```

成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m 项目第一版：“三段论”式内容设计概要开发
[master (root-commit) 3e9accf] 项目第一版：“三段论”式内容设计概要开发
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
 create mode 100644 myCss.css
```

项目代码仓库自此也开启了严肃的历史记录，我们可以输入日志命令查看，

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：

```
$ git log
commit 3e9accf1ac531691555268b47f16e8415436169f (HEAD -> master)
Author: 江科师大俞森洲 <marsterlijh@jxstnu.edu.cn>
Date:   Thu Jun 13 19:05:39 2024 +0800

    项目第一版：“三段论”式内容设计概要开发
```

4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计

4.1 响应式设计——适应显示硬件

与计算机一起使用的显示器硬件变化很大，显示器的大小和分辨率取决于成本。设计师们没有为每种类型的显示器提供每个网页的版本，而是选择让网页提供总体布局指南，并允许浏览器选择如何在给定的计算机上显示页面。因此，网页并不能提供很多细节。例如，网页的作者可以指定一组句子组成一个段落，但作者不能指定细节，如一行的确切长度或是否缩进段落的开头[1]。

允许浏览器选择显示详细信息有一个有趣的结果：当通过两个浏览器或在两台硬件不同的计算机上查看时，网页可能会出现不同的外观。如果一个屏幕比另一个屏幕宽，则可以显示的文本行的长度或图像的大小不同。重点是：网页提供了关于所需演示的一般指南；浏览器在显示页面时选择详细信息。因此，当在两台不同的计算机上或通过不同的浏览器显示时，同一个网页可能看起来略有不同[1]。

分析移动互联时代的多样化屏幕的需求。用 JavaScript 开动态读取显示设备的信息，然后按设计，使用 js+css 来部署适配当前设备的显示的代码。

4.2 实现代码

用汉语言来描述我们是如何实现的，与上一阶段比较，本阶段初次引入了 em 和 % ，这是 CSS 语言中比较高阶的语法，可以有效地实现我们的响应式设计。如代码块 4-1 所示：

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

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

  }
```

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

    }
    nav{
      border: 2px solid blue;
      height: 10%;
          }
    nav button{
     font-size: 1.1em;
    }
    footer{
      border: 2px solid blue;
      height: 5%;
    }
  </style>
```

代码块 4-1

5 应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.1 移动互联时代的用户终端多样性

在移动互联时代，用户访问 Web 应用的终端设备种类极其丰富，从传统的桌面电脑到平板电脑、智能手机乃至手表和家用电器，屏幕尺寸和分辨率千差万别。这种设备的多样性带来了独特的设计挑战—如何确保不同设备上的用户都能获得一致且优质的用户体验。

为了实现这一目标，响应式设计技术成为了一种重要工具。它允许 UI 根据设备的特性而自动调整布局、内容大小等，从而提升用户的浏览体验。

5.2 CSS 语言在响应式设计中的应用

CSS 媒体查询是实现响应式设计的核心工具之一。通过使用媒体查询，可以指定在某些屏幕尺寸条件下应用一组 CSS 规则。例如，当屏幕尺寸小于特定宽度时，可以调整字体大小或更改布局流向。

CSS

```
@media screen and (max-width: 600px) {
    .container {
        width: auto;
    }
```

```
    .column {
        float: none;
        width: 100%;
    }
}
```

这段代码表示，当屏幕尺寸小于 600 像素时，将取消列的浮动并设置宽度为 100%，从而实现垂直堆叠的布局。

5.3 JavaScript 语言在响应式设计中的作用

虽然 CSS 能够处理大部分的响应式设计问题，但某些更复杂的适应性功能可能需要借助 JavaScript 来实现。例如，可以根据屏幕尺寸动态加载不同的脚本文件，或者改变网页中元素的行为。

javascript

```
if (window.innerWidth < 800) {
    document.getElementById('myElement').src = 'smallVersion.jpg'; // 载入小尺寸图片
} else {
    document.getElementById('myElement').src = 'largeVersion.jpg'; // 载入大尺寸图片
}
```

此段 JavaScript 代码检测屏幕宽度，并根据检测结果动态更改图片源地址，以适应不同分辨率的需求。

5.4 结合 CSS 与 JavaScript 的响应式设计

在实际开发中，CSS 和 JavaScript 通常结合使用，以达到最佳的响应式设计效果。CSS 负责基础的样式调整，而 JavaScript 则用于处理交互逻辑和动态内容。

5.4.1 断点选择

合理选择媒体查询断点是响应式设计的关键。断点的选择应基于内容的需要以及常见设备分辨率的分布。

5.4.2 设计原则

遵循设计原则如简洁性、一致性和可访问性，以确保无论在何种设备上，用户都能获得良好的体验。

5.4.3 测试与优化

不断测试在不同设备上的表现，并对发现的问题进行优化，是实现高效响应式设计的必要步骤。

在移动互联时代，终端设备的多样性要求我们必须采用响应式设计技术来满足跨屏幕的用户体验。通过灵活运用 CSS 和 JavaScript，我们可以创建出能够适应各种屏幕尺寸和分辨

率的个性化 UI，从而提供更加丰富和便捷的用户体验。

6 个性化 UI 设计中对鼠标交互的设计开发

在 HTML 中，本身并不直接提供为触屏和鼠标建立对象模型的语言特性，因为 HTML 主要负责定义网页的结构和内容。然而，我们可以通过 JavaScript（通常结合 CSS 和 HTML）来实现这种交互，从而使得我们的代码逻辑能够同时处理触屏和鼠标事件。

以下是一个简单的示例，说明如何使用 JavaScript 来同时为触屏和鼠标建立交互模型：

html 结构：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Touch and Mouse Interaction</title>
    <style>
        #interactionArea {
            width: 300px;
            height: 300px;
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <div id="interactionArea">
        Click or Tap Here
    </div>
    <script src="interaction.js"></script>
</body>
</html>
```

JavaScript 代码（interaction. js）：

```
javascript
 document.getElementById('interactionArea').addEventListener('click', handleInteraction);
 document.getElementById('interactionArea').addEventListener('touchstart', handleInteraction);
   function handleInteraction(event) {
 // 判断是触屏还是鼠标点击事件
 const isTouch = event.type === 'touchstart';
 const message = isTouch ? 'You tapped the screen' : 'You clicked the mouse';
   console.log(message);
 // 这里可以添加更多的交互逻辑，比如更新 DOM、发送请求等
 }
```

在上面的示例中，我们为#interactionArea 元素添加了两个事件监听器：一个是 click 事件（用于鼠标点击），另一个是 touchstart 事件（用于触屏上的触摸开始）。这两个事件都调用了同一个 handleInteraction 函数。在 handleInteraction 函数中，我们根据事件的类型（event. type）来判断是触屏还是鼠标点击，并打印出相应的消息。

这样，无论是使用鼠标点击还是触屏触摸，都会触发相同的交互逻辑，实现了用一套代码逻辑同时为触屏和鼠标建立对象模型的目的。当然，根据具体需求，你可以在这个基础上添加更多的交互逻辑。

7. 对触屏和鼠标的通用交互操作的设计开发

在设计开发一个同时支持触屏和鼠标交互的网页时，通常会使用事件监听器来检测这两种设备的输入。为了使用一套代码逻辑同时为触屏和鼠标建立对象模型，你可以采用事件委托（Event Delegation）和抽象事件处理逻辑的策略。

以下是一个简单的示例，展示了如何使用 JavaScript 为触屏和鼠标的交互创建一套通用的对象模型：

HTML 结构：

```
html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>触屏和鼠标交互示例</title>
<style>
#interactable {
width: 200px;
height: 200px;
background-color: lightblue;
display: flex;
justify-content: center;
align-items: center;
color: white;
font-size: 24px;
}
</style>
</head>
<body>
<div id="interactable">点击或触摸我</div>
  <script src="interaction.js"></script>
```

```
</body>
</html>
```

JavaScript 逻辑（interaction. js）：

```
javascript
document.addEventListener('DOMContentLoaded', function() {
var interactable = document.getElementById('interactable');
  // 抽象的事件处理函数
function handleInteraction(event) {
// 根据事件类型，你可以在这里添加更具体的逻辑
// 例如，检查是否是左键点击或触屏的特定行为
  // 更新交互区域的文本
interactable.textContent = '你' + (event.type === 'click' ? '点击' : '触摸') + '了我';
  // 阻止默认行为和冒泡（如果需要）
// event.preventDefault();
// event.stopPropagation();
}
  // 为鼠标点击添加事件监听器
interactable.addEventListener('click', handleInteraction);
  // 为触屏事件添加事件监听器
// 注意：在触屏设备上，你可能需要监听 'touchstart', 'touchmove', 'touchend' 等事件
// 但为了简单起见，这里只监听 'touchstart' 作为触屏的“点击”事件
interactable.addEventListener('touchstart', handleInteraction);
});
```

在这个示例中，我们创建了一个可交互的 元素，并为其添加了鼠标点击 (click) 和触屏触摸 (touchstart) 的事件监听器。当这些事件被触发时，它们都会调用同一个 handleInteraction 函数。这个函数根据触发的事件类型来更新 的文本内容。

当然，还有其他方法可以设计一套同时支持触屏和鼠标交互的代码逻辑。

使用现代前端框架，现代前端框架（如 React、Vue、Angular 等）通常提供了抽象层来处理不同设备的输入。这些框架允许你编写组件，并通过事件处理函数来响应不同的交互事件，而无需关心事件的具体来源（触屏或鼠标）。使用指针事件（Pointer Events），

指针事件是一个 W3C 规范，旨在统一鼠标、触摸笔、触摸等不同类型的输入设备。通过监听 pointerdown、pointermove、pointerup 等事件，你可以编写一套代码来处理所有类型的指针输入。

javascript 代码：

```
interactable.addEventListener('pointerdown', handlePointerDown);
function handlePointerDown(event) {
// 处理指针按下事件
console.log('Pointer down:', event.pointerType); // 可以是 'mouse', 'pen', 'touch' 等
}
```

8. UI 的个性化键盘交互控制的设计开发

阐述探索和利用 keydown 和 keyup 键盘底层事件，为未来 UI 的键盘功能提供底层强大的潜力。

因为系统中只有一个键盘，所以我们在部署代码时，把键盘事件的监听设置在 DOM 文档最大的可视对象——body 上，通过测试，不宜把键盘事件注册在 body 内部的子对象中。代码如下所示：

```
    $("body").addEventListener("keydown",function(ev){
    ev.preventDefault() ; //增加“阻止事件对象的默认事件后”，不仅 keypress 事件将不再响应，而且系统的热键，如“F5 刷新页面/Ctrl+R ”、“F12 打开开发者面板”等也不再被响应
    let k = ev.key;
    let c = ev.keyCode;
    $("keyStatus").textContent = "按下键 ： " + k + " ，"+ "编码 ： " + c;
   });
 $("body").addEventListener("keyup",function(ev){
    ev.preventDefault() ;
    let key = ev.key;
    $("keyStatus").textContent =  key + " 键已弹起" ;
    if (printLetter(key)){
        $("typeText").textContent += key ;
     }
     function printLetter(k){
     if (k.length > 1){ //学生须研究这个逻辑的作用
          return false ;
     }
     let puncs =
['~','`','!','@','#','$','%','^','&','*','(',')','-','_','+','=',',',',','.',';',';',';','<
','>','?','/',' ','\'','\"'] ;
     if ( (k >= 'a' && k <= 'z') || (k >= 'A' && k <= 'Z')
|| (k >= '0' && k <= '9'))  {
          console.log("letters") ;
          return true ;
     }
     for (let p of puncs ){
      if (p === k) {
            console.log("puncs") ;
           return true ;
          }
     }
     return false ;
      //提出更高阶的问题，如何处理连续空格和制表键 tab？
    }   //function printLetter(k)
 });
```

图 8-1 PC 端运行效果图

图 8-2 移动端二维码

9.谈谈本项目中的高质量代码

这是一本关于指导计算机的书。今天，计算机和螺丝刀一样常见，但它们要复杂得多，让它们做你想让它们做的事情并不总是那么容易。

如果你的电脑任务是一个常见的、众所周知的任务，比如给你看电子邮件或像计算器一样工作，你可以打开相应的应用程序开始工作。但对于独特或开放式的任务，可能没有应用程序。

这就是编程的用武之地。编程是构建程序的行为——一组精确的指令告诉计算机该做什么。因为计算机是愚蠢、迂腐的野兽，编程从根本上来说是乏味和令人沮丧的。幸运的是，如果你能克服这一事实，甚至可以享受愚蠢机器所能处理的思维的严谨性，那么编程是有回报的。它可以让你在几秒钟内完成那些需要你用手才能完成的事情。这是一种让你的计算机

工具做以前做不到的事情的方法。它提供了一个很好的抽象思维练习[6]。

首先，我们需要创建一个 Pointer 类，用于封装鼠标和触屏的操作。然后，我们将实现一个控制器类，用于处理用户输入并调用相应的操作。最后，我们将实现一个视图类，用于显示当前的状态。

1.创建 Pointer 类：

html：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MVC 模式示例</title>
    <style>
        #container {
            width: 600px;
            height: 400px;
            border: 1px solid black;
            position: relative;
        }
    </style>
</head>
<body>
    <div id="container"></div>
    <script>
        class Pointer {
            constructor() {
                this.x = 0;
                this.y = 0;
            }

            move(x, y) {
                this.x = x;
                this.y = y;
            }

            click() {
                console.log(`点击坐标：(${this.x}, ${this.y})`);
            }
        }
    </script>
</body>
</html>
```

2.创建控制器类：

```
<script>
    class Controller {
        constructor(pointer) {
            this.pointer = pointer;
        }

        onMouseMove(x, y) {
            this.pointer.move(x, y);
        }

        onMouseClick() {
            this.pointer.click();
        }

        onTouchMove(x, y) {
            this.pointer.move(x, y);
        }

        onTouchClick() {
            this.pointer.click();
        }
    }
</script>
```

3.创建视图类：

```
<script>
    class View {
        constructor(controller) {
            this.controller = controller;
        }

        display() {
            console.log("显示当前状态");
        }
    }
</script>
```

4.使用 MVC 模式组织代码：

```
<script>
    function main() {
        const pointer = new Pointer();
        const controller = new Controller(pointer);
        const view = new View(controller);

        // 模拟鼠标移动和点击事件
        controller.onMouseMove(100, 200);
        controller.onMouseClick();
```

```
        // 模拟触屏移动和点击事件
        controller.onTouchMove(300, 400);
        controller.onTouchClick();

        view.display();
    }

    main();
</script>
```

这样，我们就实现了一个简单的 MVC 设计模式，可以同时对鼠标和触屏进行控制。

10.用 gitBash 工具管理项目的代码仓库和 http 服务器

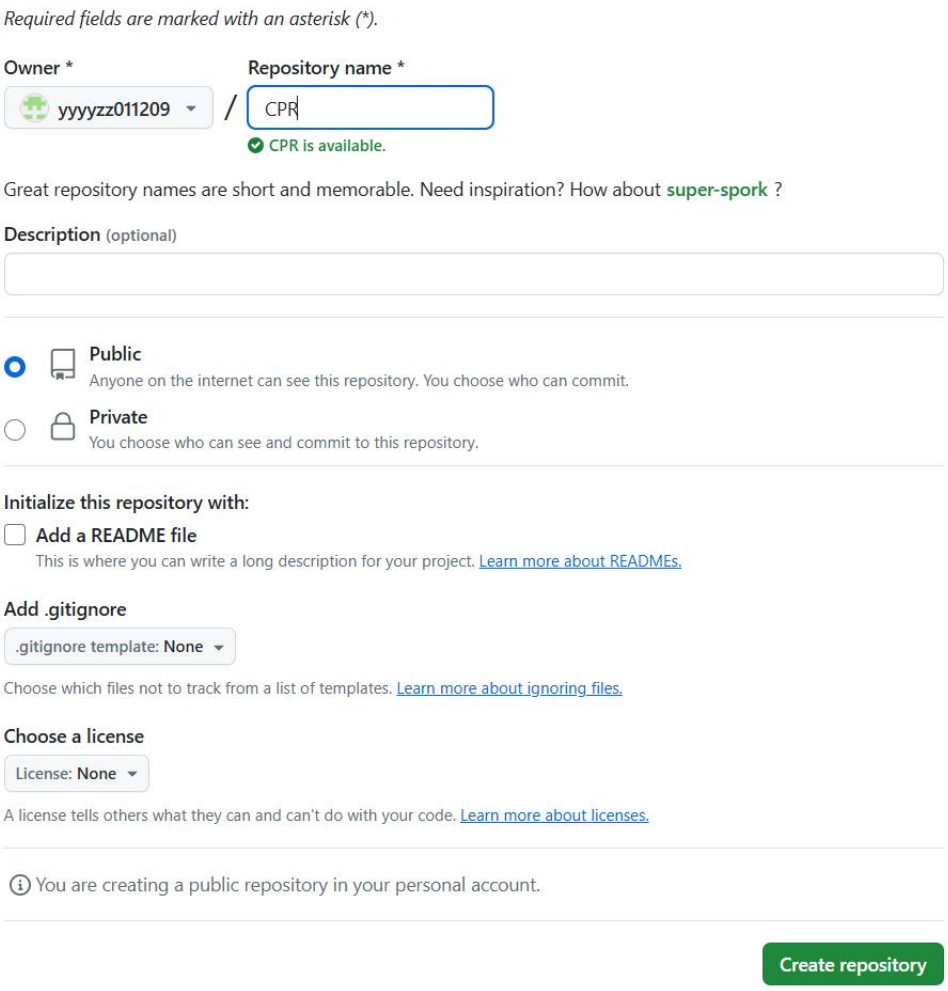
10.1 经典 Bash 工具介绍

当我们谈到命令行时，我们实际上指的是 shell。shell 是一个接受键盘命令并将其传递给操作系统执行的程序。几乎所有的 Linux 发行版都提供了一个名为 bash 的 GNU 项目的 shell 程序。这个名字是 bourne-reagain shell 的首字母缩写，指的是 bash 是由 Steve bourne 编写的原始 Unix shell 程序 sh 的增强替代品[7]。

像 Windows 一样，像 Linux 这样的类 Unix 操作系统将其文件组织在所谓的分层目录结构中。这意味着它们以树状目录模式组织（有时在其他系统中称为文件夹），其中可能包含文件和其他目录。文件系统中的第一个目录称为根目录。根目录包含文件和子目录，其中包含更多的文件和子文件夹，依此类推[7]。

10.2 通过 gitHub 平台实现本项目的全球域名

10.3 创建一个空的远程代码仓库



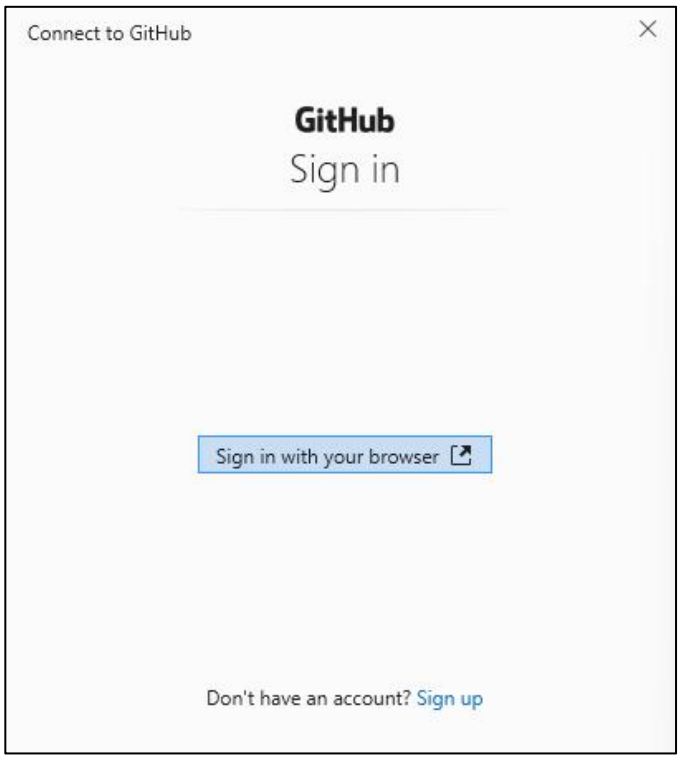
点击窗口右下角的绿色“Create repository”，则可创建一个空的远程代码仓库。

10.4 设置本地仓库和远程代码仓库的链接

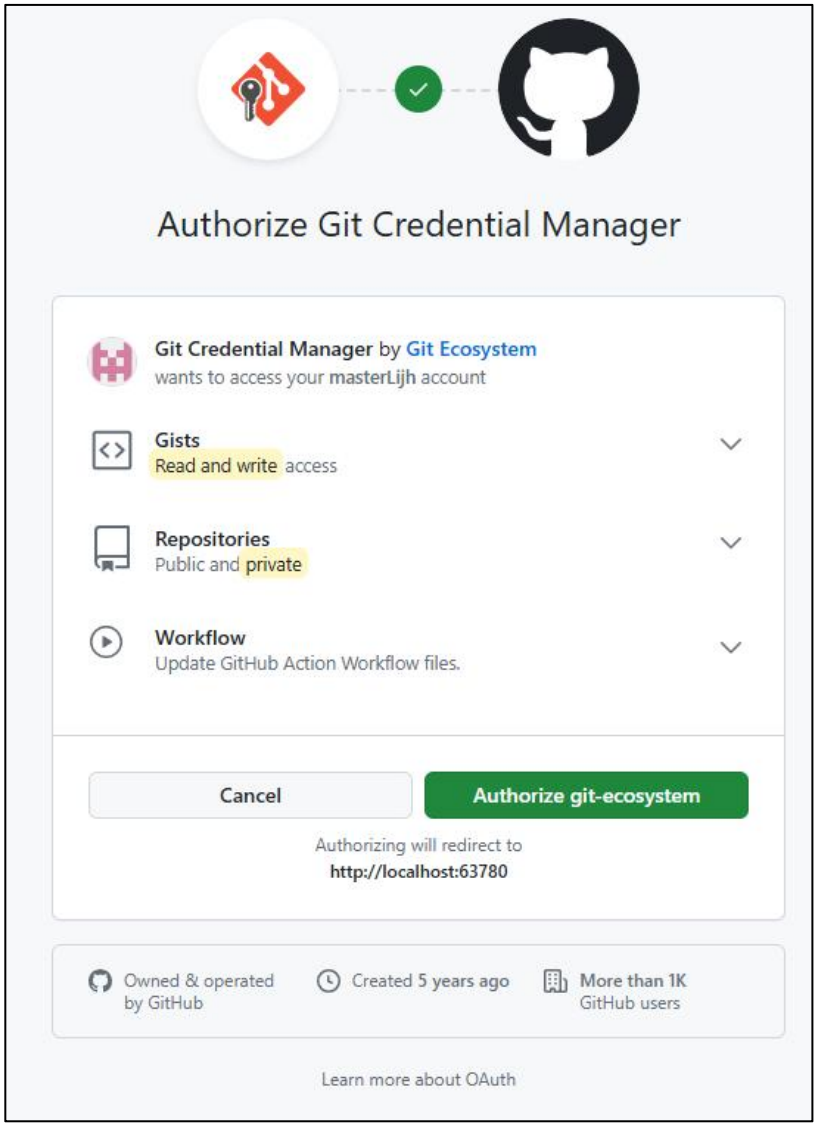
进入本地 webUI 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接

```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md
$ git init
$ git add README.md
$ git commit -m "这是我第一次把代码仓库上传至 gitHub 平台"
$ git branch -M main
$ git remote add origin
        https://github.com/masterLijh/userName.github.io.git
$ git push -u origin main
```

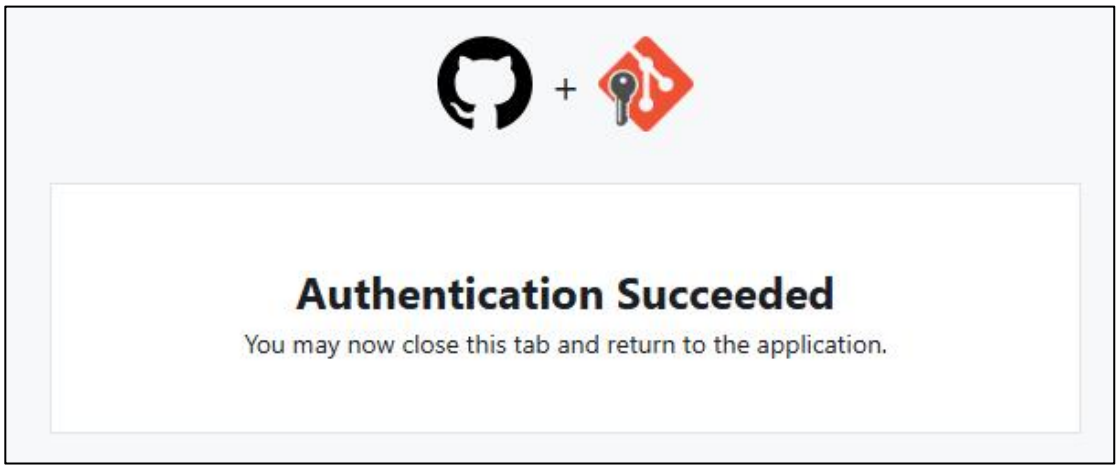
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



再次确认授权 gitBash 拥有访问改动远程代码的权限，如下图所示：

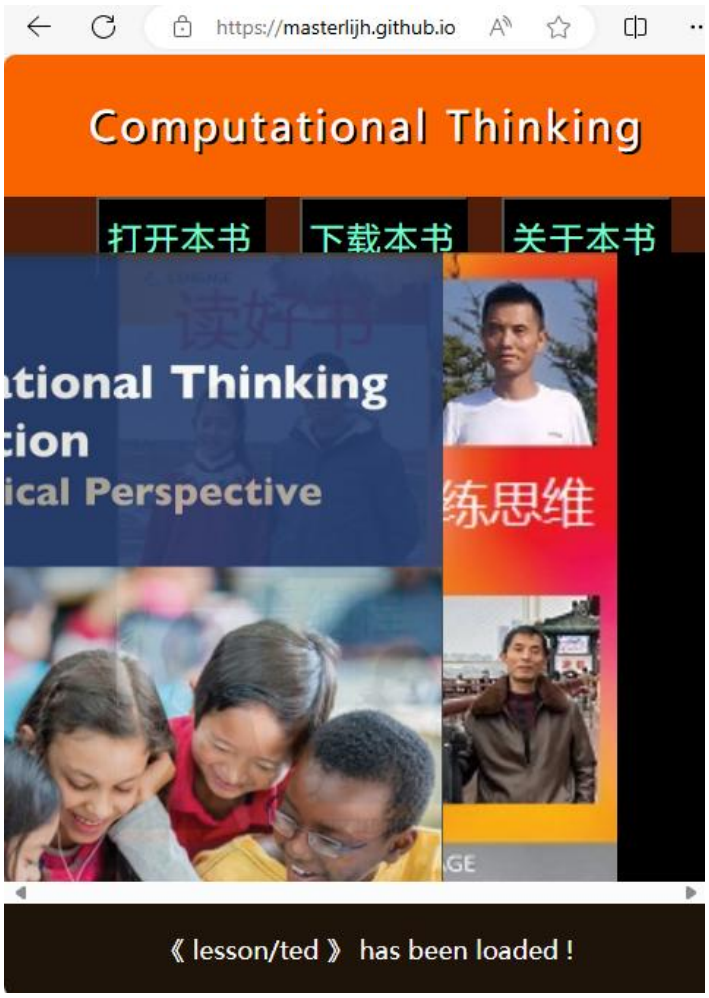


最后，GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。



从此，我们无论在本地做了任何多次代码修改，也无论提交了多少次，上传远程时都会把这些代码和修改的历史记录全部上传 github 平台，而远程上传命令则可简化为一条：git push ，极大地方便了本 Web 应用的互联网发布。

远程代码上传后，项目可以说免费便捷地实现了在互联网的部署，用户可以通过域名或二维码打开，本次使用 PC 的微软 Edge 浏览器打开，本文截取操作中间的效果图，如下所示：



参考文献

[1] W3C. W3C 'shistory.W3C Community. [EB/OL]. https://www.w3.org/about/. https://www.w3.org/about/history/. 2023.12.20

[2] Douglas E. Comer. The Internet Book [M] (Fifth Edition). CRC Press Taylor & Francis Group, 2019: 217-218

[3] John Dean,PhD. Web programming with HTML5,CSS,and JavaScript[M]. Jones & Bartlett Learning,LLC. 2019: 2

[4] John Dean,PhD. Web programming with HTML5,CSS,and JavaScript[M]. Jones & Bartlett Learning,LLC. 2019: xi

[5] Behrouz Forouzan. Foundations of Computer Science[M](4th Edition). Cengage Learning EMEA,2018: 274--275

[6] Marijn Haverbeke. Eloquent JavaScript 3rd edition. No Starch Press,Inc, 2019.

[7] William Shotts. The Linux Command Line, 2nd Edition [M]. No Starch Press, Inc, 245 8th Street, San Francisco, CA 94103, 2019：3-7