成 绩：

江 西 科 技 师 范 大 学

课程设计（论文）

题目（中文）：基于 Web 客户端的个性化 UI 的设计和编程

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

院 （系）：元宇宙产业学院

专 业：计算机科学与技术

学生姓名：梅松祥

学 号：20213592

指导教师：李健宏

2024 年 6 月 19 日

目录

基于 Web 客户端技术的个性化 UI 的设计和编程

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

科师大元宇宙产业学院 2021 级 梅松祥

摘要:针对移动互联网时代的软件开发需求，本项目聚焦于 HTML5 为核心的 Web 客户端技术，深入研究并实践了 HTML 内容建模、CSS 样式设计与 JavaScript 交互编程，旨在构建一个个性化且响应式的用户界面（UI）。项目亮点在于创新性地构建了一套通用的 pointer 模型，统一处理鼠标与触屏输入，体现了面向对象的编程思想。采用增量式开发模式，历经六轮迭代，每个周期均包含分析、设计、实施与测试四个阶段，确保了代码质量的逐步提升。项目全程手工编码，未依赖任何第三方框架或库，充分展现了自主开发能力。同时，利用 Git 进行版本控制，记录了 12 次代码提交，最终将项目部署至 GitHub，借助其 HTTP 服务器功能，实现了全球范围内的便捷访问，展现了项目成果的开放性和可共享性。这一过程不仅验证了 Web 技术在跨平台应用中的优势，也为开发者提供了宝贵的实践经验与代码示例。

关键字：web；pointer 模型；git 管理

1. 前言

学习计划：首先如果需要完成自己的毕设项目，需要扎实的基础功底，这就需要理解什么是 web 毕设，自然需要进行一番规划，规划如下：

第一阶段：项目规划与需求分析（第 1 周）

- **目标：**明确项目主题，进行需求分析，制定初步的项目计划。
- **学习内容：**
  - 学习如何进行项目管理和需求分析的基本概念。
  - 研究类似项目，收集灵感，明确你的项目将解决什么问题。
  - 制定项目时间表，包括里程碑和截止日期。
- **实践：**撰写项目提案，列出功能需求和技术要求。

第二阶段：技术选型与基础知识学习（第 2-3 周）

- **目标：**选择合适的技术栈，深入学习 Web 开发基础知识。
- **学习内容：**
  - 了解不同的 Web 开发框架和工具（如 React, Vue, Angular, Flask, Django）。
  - 学习 HTML, CSS, JavaScript 基础，理解 DOM 操作。
  - 学习后端语言如 Python 或 Node.js。

- 掌握数据库知识，如 SQL 或 NoSQL 数据库（MySQL, MongoDB）。

- **实践**：创建几个小型的 Web 页面，练习 HTML/CSS 布局，尝试使用所选框架创建简单的组件。

第三阶段：设计与原型制作（第 4-5 周）

- **目标**：设计用户界面，制作项目原型。
- **学习内容**：
  - 学习 UI/UX 设计原则。
  - 使用工具如 Figma 或 Sketch 进行界面设计。
  - 学习响应式设计，确保设计在不同设备上良好显示。
- **实践**：制作项目的设计草图和交互原型。

第四阶段：前端开发（第 6-8 周）

- **目标**：实现前端功能，构建用户界面。
- **学习内容**：
  - 学习前端框架的高级用法。
  - 掌握状态管理和路由（如 Redux, React Router）。
  - 学习如何使用 API 与后端进行通信。
- **实践**：逐步实现前端功能，集成设计原型，进行单元测试。

第五阶段：后端开发与数据库集成（第 9-10 周）

- **目标**：开发后端逻辑，集成数据库。
- **学习内容**：
  - 学习后端框架的使用。
  - 数据库设计与实现。
  - 学习 RESTful API 设计。
- **实践**：开发后端服务，连接数据库，实现数据增删改查功能。

第六阶段：整合与测试（第 11-12 周）

- **目标**：整合前后端，进行全面测试。
- **学习内容**：
  - 学习部署和服务器配置。
  - 单元测试和集成测试技巧。
  - 学习如何使用 Git 进行版本控制。
- **实践**：整合前后端代码，部署到服务器，进行全面测试，修复发现的问题。

第七阶段：优化与文档编写（第 13 周）

- **目标**：优化性能，编写项目文档。
- **学习内容**：
  - 性能优化技巧（如代码压缩，图片优化）。
  - 文档编写，包括项目介绍、技术架构、使用说明。
- **实践**：进行最后的优化，编写详细的项目文档。

第八阶段：演示与提交（第 14 周）

- **目标**：准备演示，提交项目。
- **学习内容**：
  - 学习如何进行有效的项目演示。
  - 最终审查项目，确保所有功能正常工作。
- **实践**：准备演示文稿，向导师或小组演示项目，提交所有文档和源代码。

项目技术路线：

应用软件开发的方向内涵非常丰富，也是非常庞杂的，仅仅是表达代码的现代高级计算机语言常见的就不下十几种，而且每种都是博大精深独具自己特色的，他们在擅长领域都有一席之地，有的专攻 Android 手机平台、有的专门支持苹果的 IOS 平台、还有的仅仅适用于 window 桌面系统，当然还有用于一些大型机构使用 UNIX/Linux 平台的软件开发。现代计算机的种类多样，现代移动端设备的多样化，运行的操作系统也是同样是如此复杂，因此我们提出的第一个问题就是：软件无论是开发还是运行能否做到跨硬件跨系统 ？下面简称跨平台 ，人人都承认人的时间是最珍贵的，也就说：开发者能否写一套通用的代码，就可以直接在所有的硬件和操作系统上运行？就像圣经故事里的巴别塔，代码就是计算机世界里的语言通用的语言。当然肯定不是我第一个提出这个问题的，这个理想其实就是是改变互联网世界的 Web 标准，Web 之父（Tim）在发明 Web 的基本技术架构以后，成立了 W3C 组织，[2]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，这帮科学家与 Web 行业也还一直在致力于完善这个伟大而光荣的理想。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，这就是我所谓的 Web 应用开发的技术路线。

参考资料：the fundation of computer science

研究方法：模型法，文献研究法。

1.1 毕设任务分析

毕设任务：

我们作为计算机科学技术专业的本科生，在即将完成学业之际，的确很必要

设计开发一个本专业的作品，来回顾总结本学科专业学习的知识系统，梳理课程体系最核心的东西，体现我们的真实能力。

在我的毕业设计中，涉及的有关核心课程的理论包括：面向对象的程序设计语言、数据结构和算法、操作系统、软件工程等。以前这些核心课程供理论指导感觉非常抽象，加之基本上以理论知识为主，因此学完后我们感觉一直有所缺憾，本人与导师沟通后也一致认为，若能在实践层面应用这些核心课程的关键知识，则必然会在理解和技术二个维度提升自己的专业性。

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

深刻理解计算机系统（computing system）对我们专业开发者而言，是非常重要的，这也是我们即将成为建设国家现代化的工程师不同于与其他专业的人的特色，从其他专业眼中看来，我们是计算机专业的，我们对计算机系统的理解一定不是浮于表面的，而是尽量要更加接近计算机的本质，对任何技术的理解则是能接近技术的底层和基本原理。

1.2 研学计划

研学计划：研学计划，也称为研究性学习计划或探究学习计划，是一种教育策略，旨在通过引导学生进行自主探索、实验和研究，促进深度学习和批判性思维能力的发展。研学计划通常围绕一个具体的主题或问题展开，鼓励学生运用跨学科的知识和技能，通过实践、调查、数据分析等方式，深入理解和解决问题。通过研学计划，学生不仅能够获得丰富的知识和技能，还能学会如何面对挑战，如何与他人合作，以及如何成为一个独立思考和负责任的学习者。以下为本人的研学计划表格：

书名	阅读时间
The fundation of computer science	2024.3.1
Web techonology	2024.4.1
The art of javascript	2024.5.1
The beauty of web	2024.6.1

1.3 研究方法

研究方法：1.文献查找法　2.模型研究法

1. 文献查找法（文献研究法）

文献查找法，也称文献研究法，是一种通过收集、分析和综合已有的文献资料来进行研究的方法。这种方法主要用于获取关于研究主题的历史背景、理论框架、前人研究结果以及相关理论和实践的信息。文献查找法在学术研究和项目前期准备中扮演着重要角色，其目的是为了：

- **系统地了解和掌握研究领域的现有知识**：通过阅读书籍、学术期刊文章、会议论文、技术报告、官方文档、电子资源等，了解研究主题的全貌。
- **识别研究空白**：在大量文献的基础上，找出尚未被充分探讨的问题，为自己的研究确定方向和范围。
- **理论构建与假设形成**：基于文献中的理论和观点，形成自己的研究理论框架或假设。
- **方法论借鉴**：学习前人的研究方法，为自己的研究设计提供参考。
- **论证支撑**：引用文献作为自己研究的证据和支撑，增强研究的说服力。

文献查找法的过程通常包括：确定研究问题、设计文献检索策略、执行检索、筛选文献、阅读和分析文献、文献综述写作等步骤。

2. 模型研究法

模型研究法是一种通过构建和分析模型来研究复杂系统行为和预测未来趋势的研究方法。模型可以是物理模型、数学模型、计算机模拟模型等，具体取决于研究的领域和目的。模型研究法的关键步骤包括：

- **定义研究问题**：明确研究的目标和需要解决的问题。
- **模型构建**：根据研究问题，选择或创建适合的模型。这可能涉及理论假设、方程设定、参数确定等。
- **数据收集与校准**：收集必要的数据，用于模型的校准和验证。
- **模型分析**：通过计算、模拟或实验，分析模型的行为和输出。
- **结果解释与应用**：解释模型分析的结果，将其应用于实际问题的解决，或者对未来趋势进行预测。

模型研究法在许多领域都有应用，如经济学中的宏观经济模型、生态学中的种群动态模型、工程学中的结构力学模型等。这种方法的优势在于能够处理复杂系统

中的不确定性，通过量化分析帮助决策者做出更合理的判断。然而，模型的有效性依赖于模型假设的合理性以及模型参数的准确性，因此在应用时需要谨慎。

2. 技术总结和文献综述

2.1 Web 平台和客户端技术概述

Web 之父 Tim Berners-Lee 在发明 Web 的基本技术架构以后，就成立了 W3C 组织，该组织在 2010 年后推出的 HTML5 国际标准，结合欧洲 ECMA 组织维护的 ECMAScript 国际标准，几乎完美缔造了全球开发者实现开发平台统一的理想，直到今天，科学家与 Web 行业也还一直在致力于完善这个伟大而光荣的理想[1]。学习 Web 标准和 Web 技术，学习编写 Web 程序和应用有关工具，最终架构一套高质量代码的跨平台运行的应用，是我的毕设项目应用的技术路线。

Web 平台：Web 平台指的是构建和提供网络应用的一系列技术、协议和标准。

客户端技术：Web 平台的客户端技术是指在用户的设备用于展示、控制和交互网页内容。这些技术使用户能够通过 Web 浏览器访问和使用网络上的信息和服务。使用 html、css、js 作为编写 web 页面的编程语言。

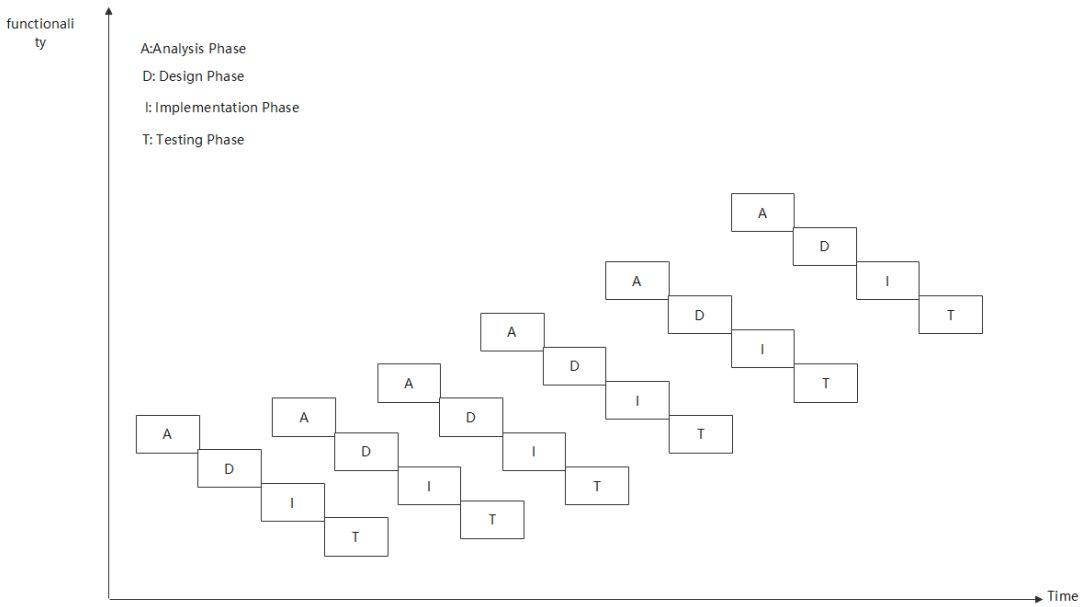
Web 应用的程序设计体系由三大语言有机组成：HTML, CSS, JavaScript。这三大语言的组合也体现了人类社会化大生产分工的智慧，可以看作用三套相对独立体系实现了对一个信息系统的描述和控制，可以总结为：HTML 用来描述结构（Structure）、CSS 用来描述外表（presentation）、Javascript 用来描述行为（Behavior）[3]；这也可以用经典的 MVC 设计模式来理解 Web 平台架构的三大基石，Model 可以理解为 HTML 标记语言建模，View 可以理解为用 CSS 语言来实现外观，Controller 则可理解为用 JavaScript 结合前面二个层次，实现了在微观和功能层面的代码控制。

2.2 项目的增量式迭代开发模式

将本科专业学生的毕业设计软件作品视为一个系统工程，而非简单的单一用途程序，意味着需要采取更加严谨和系统化的开发方法。这样的项目往往涉及大量的手写代码，以及多个相互关联的功能模块，其复杂度和工作量远远超过了简单的编程任务。因此，从软件工程的视角出发，规范项目的编写过程，不仅有助于确保项目的顺利进行，还能提升代码质量，降低维护成本，提高团队协作效率。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。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：



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 4-1 用例图所示：

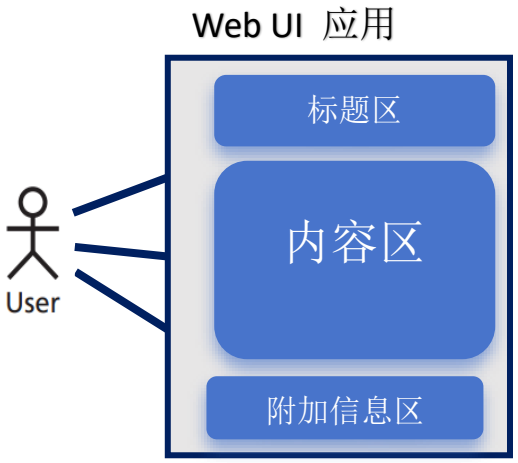


图 4-1 用例图

3.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
    《 我的毕设题目 》
</header>
 <main>
      我的主题内容： ‘读好书、练思维、勤编程’ @masterLijh 计算思维系列课程
 </main>
<footer>
  CopyRight XXX  江西科技师范大学 2024-2025
</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 4-2 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。

扫描二维码

图 4-2 PC 端运行效果图

扫描二维码

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。

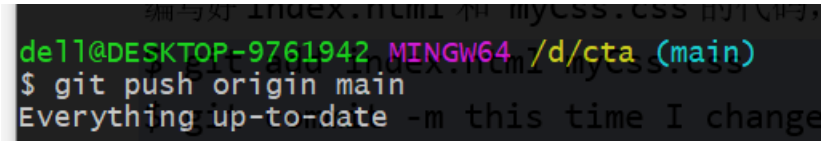
进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ mkdir cta
$ cd cta
$ git init
$ git config user.name moonandwindy
$ git config user.email 2152540548@qq.com
$ 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 this time I change the color of the border and font
```

成功提交代码后，gitbash 的反馈如下所示：

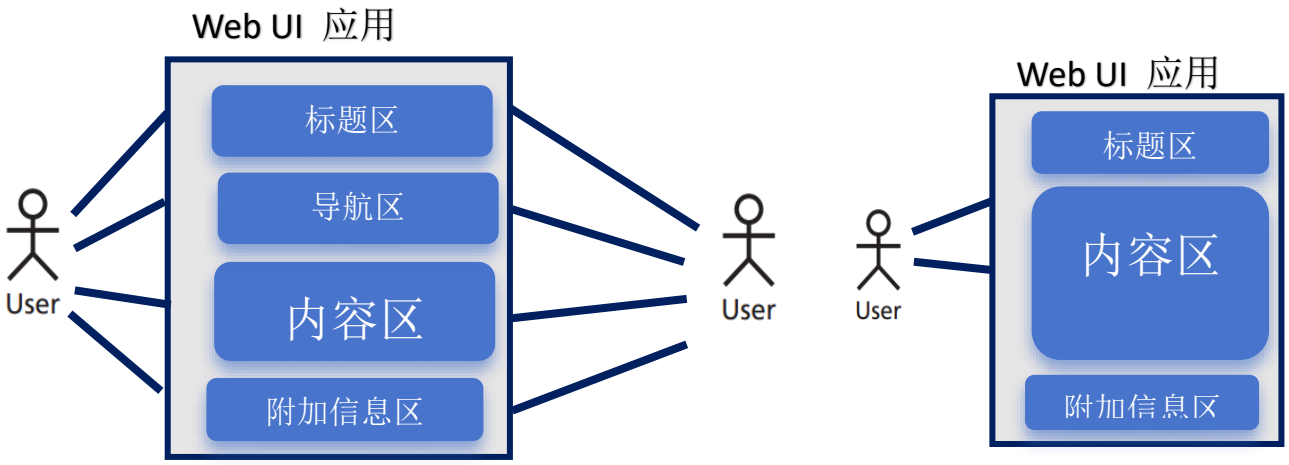


项目代码仓库自此也开启了严肃的历史记录，我们可以输入日志命令查看，

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：

4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计



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。

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

用汉语言来描述我们是如何实现的：与上一阶段比较，本阶段首次使用了 JavaScript ，首先创建了一个 UI 对象，然后把系统的宽度和高度记录在 UI 对象中，又计算了默认字体的大小，最后再利用动态 CSS，实现了软件界面的全屏设置。如代码块 4-2 所示：

```
  <script>
      var UI = {};
      UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;
      UI.appHeight = window.innerHeight;
      const LETTERS = 22 ;
      const baseFont = UI.appWidth / LETTERS;

      //通过更改 body 对象的字体大小，这个属性能够遗传其子子孙孙
      document.body.style.fontSize = baseFont + "px";
      //通过把 body 对象的宽度和高度设置为设备/屏幕的宽度和高度，实现全屏。
      //通过 CSS 对子对象百分比（纵向）的配合，从而实现响应式设计的目标。
      document.body.style.width = UI.appWidth - 2*baseFont + "px" ;
      document.body.style.height = UI.appHeight - 4*baseFont + "px";
   </script>
```

代码块 4-2

图表 1-1 手机端口

PC端口

《经济发展》

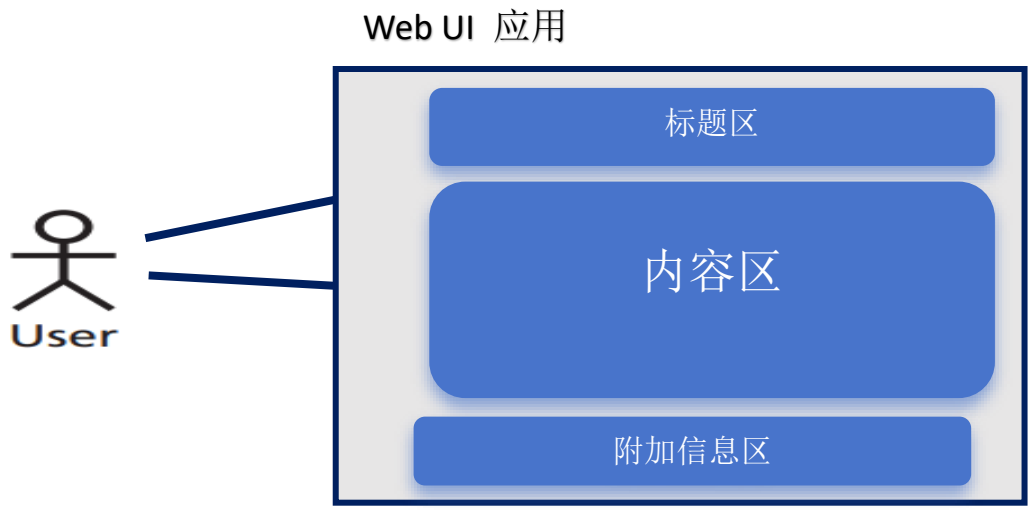
我的主题内容： ‘读好书、练思维、勤编程’ @moonandwindy 计算思维系列课程

CopyRight 梅松祥 江西科技师范大学 202-2025

其他章节： 1.2 1.3 1.4 1.5 1.6

图表 2-1PC 端口

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI



图表 6-1 响应用例图

对于进行窄屏和宽屏的 UI 设计采用 ADIT 的开发方式，以下是开发的过程解释：

A：需要分析目标用户群体的设备特性，确定 UI 需要支持的最小和最大屏幕尺寸范围。在这个案例中，我们关注于创建一个 UI，它能根据屏幕宽度自动调整字体大小，以确保在窄屏和宽屏设备上都能提供良好的阅读体验。此外，我们需要确保 UI 在不同屏幕尺寸下都能全屏显示，以最大化可用空间。

D: 设计阶段涉及规划 UI 的布局和响应机制。在这个场景中，我们的设计考虑如下：

使用变量 UI.appWidth 和 UI.appHeight 来存储屏幕的宽度和高度。

当屏幕宽度大于 600 像素时，UI 宽度固定为 600 像素；否则，UI 宽度等于屏幕宽度。

字体大小基于屏幕宽度动态计算，这里使用 UI.appWidth / 22 作为基数，以确保文本在不同设备上具有相似的可读性。

调整 document.body 的 fontSize、width 和 height 属性，以实现全屏显示并预留边距。

I：在实施阶段，我们将设计转化为具体的代码实现。代码如下：

```
var UI = {};
    UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;
    UI.appHeight = window.innerHeight;
     const LETTERS = 22 ;
const baseFont = UI.appWidth / LETTERS;

//通过更改 body 对象的字体大小，这个属性能够遗传其子子孙孙
    document.body.style.fontSize = baseFont + "px";
    //通过把 body 对象的宽度和高度设置为设备/屏幕的宽度和高度，实现全屏。
    //通过 CSS 对子对象百分比（纵向）的配合，从而实现响应式设计的目标。
    document.body.style.width = UI.appWidth - 2*baseFont + "px" ;
document.body.style.height = UI.appHeight - 4*baseFont + "px";
```

T：测试阶段非常重要，确保 UI 在各种屏幕尺寸和设备上都能正常工作。测试应覆盖以

下方面：

在窄屏设备上检查 UI 是否能正确调整字体大小，保持文本清晰可读。

在宽屏设备上确认 UI 是否能充分利用额外的空间，同时字体大小适中。

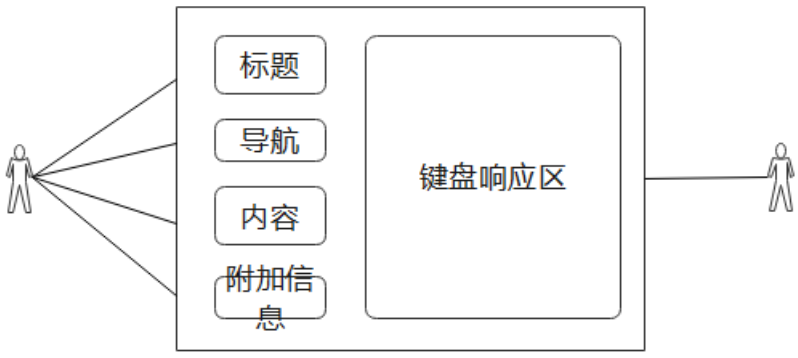
检查在不同分辨率和方向下的设备上，UI 是否能平滑过渡，没有布局错乱。

确认在调整窗口大小时，UI 能实时响应，动态调整布局。

以下为实现图：

图表 3-1 手机端口实现图

6. 个性化 UI 设计中对鼠标交互的设计开发



图表 4-1 响应用例图

阐述用一套代码逻辑同时为触屏和鼠标建立对象模型

对于个性化 UI 设计鼠标交互设计的开发中，本项目采取了 ADIT 方式：

A：旨在创建一个简单的交互式应用，能够响应用户的鼠标动作（按下、移动、移出）和键盘按键。为了确保代码的有效性和兼容性，我们需要考虑不同浏览器的事件模型、事件对象的属性，以及可能的性能影响。

D：设计阶段涉及规划如何实现需求。我们决定使用自定义的 $ 函数来简化 DOM 元素的选取，这将提高代码的可读性和维护性。同时，我们将利用原生 JavaScript 的事件监听器来处理鼠标和键盘事件，这样可以确保跨浏览器的兼容性。

对于鼠标事件，我们设计了以下功能：

当鼠标在指定元素上按下时，记录并显示鼠标位置。

当鼠标在元素上移动时，实时更新并显示鼠标位置。

当鼠标移出元素时，显示相应的消息。

对于键盘事件，我们设计了捕获按键并显示按键名称和键码的功能。

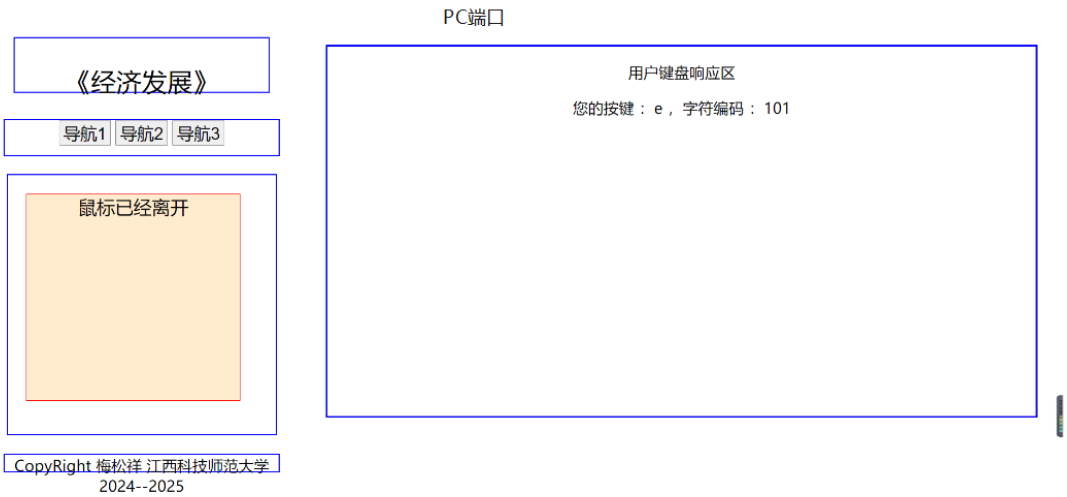
I：在实施阶段，我们编写代码来实现设计阶段规划的功能。代码片段已经包含了实现这些功能的具体细节。自定义 $ 函数用于选择 DOM 元素，而事件监听器则负责捕获和处理鼠标及键盘事件。

以下是核心代码的实现：

```
var mouse={};
mouse.isDown= false;
mouse.x= 0;
mouse.deltaX=0;
$("bookface").addEventListener("mousedown",function(ev){
    let x= ev.pageX;
    let y= ev.pageY;

    console.log("鼠标按下了，坐标为："+"("+x+","+y+")");
    $("bookface").textContent= "鼠标按下了，坐标为："+"("+x+","+y+")";
});
$("bookface").addEventListener("mousemove",function(ev){
    let x= ev.pageX;
    let y= ev.pageY;

    console.log("鼠标正在移动，坐标为："+"("+x+","+y+")");
    $("bookface").textContent= "鼠标正在移动，坐标为："+"("+x+","+y+")";
});
$("bookface").addEventListener("mouseout",function(ev){
    //console.log(ev);
    $("bookface").textContent="鼠标已经离开";

});
$("body").addEventListener("keypress",function(ev){
    let k = ev.key;
    let c = ev.keyCode;
    $("keyboard").textContent = "您的按键 ：" + k + " ，"+ "字符编码 ：" + c;
```

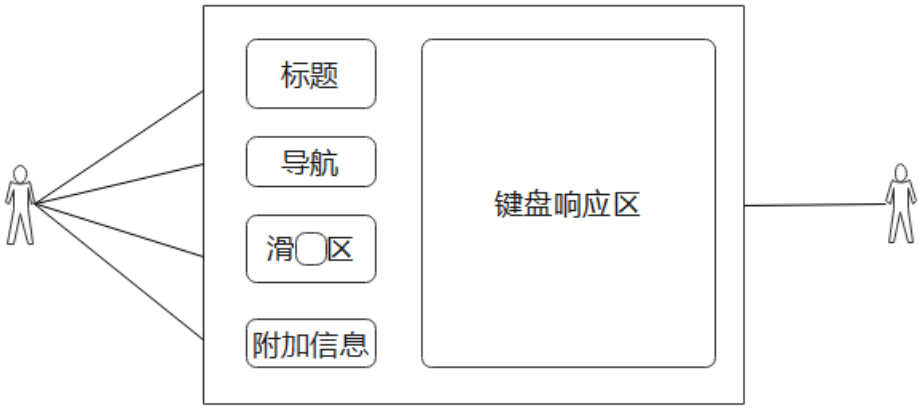
```
});

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

T：测试阶段至关重要，用于验证功能是否按预期工作。测试应该包括：
检查鼠标按下、移动、移出事件是否被正确捕获和处理。
测试键盘按键事件是否准确识别按键和键码。
检查自定义 $ 函数在不同元素 ID 和选择器下的表现。
确保所有事件处理逻辑在各种浏览器环境下均能稳定运行。
通过全面测试，我们可以确保代码的质量和稳定性，满足设计阶段设定的需求。
以下为实现图：



图表 5-1 响应实现图

7. 对触屏和鼠标的通用交互操作的设计开发



图表 6-6 鼠标触屏交互用例图

A：首先，代码分析了设备的屏幕尺寸，确定了应用的宽度和高度，并且基于屏幕宽度计算了一个基准字体大小，用于调整页面元素的大小，确保页面在不同设备上都能良好显示。它还检测窗口宽度，如果小于 1000 像素，则隐藏一个 ID 为 aid 的元素，并相应地调整其宽度和高度。

D：设计部分主要体现在代码对用户输入的处理上，特别是鼠标和触摸屏输入。通过监听 mousedown, mouseup, mousemove, touchstart, touchmove, 和 touchend 事件，代码能够识别用户的交互动作，并根据这些动作更新页面上的元素状态。例如，当用户按下鼠标或触摸屏幕时，记录下开始位置；当移动鼠标或手指时，计算并显示移动的距离；当释放鼠标或手指时，判断是否进行了有效的拖动或滑动。

I：在实施部分，代码使用了一个自定义的$函数来查找 DOM 元素，这与 jQuery 库中的$函数相似。然后，通过添加事件监听器，实现了对 bookface 元素的交互控制。具体来说，handleBegin 函数在接收到鼠标或触摸事件时开始记录坐标，handleEnd 函数在事件结束时判断是否有足够的位移来认定为一次有效的拖动或滑动，而 handleMoving 函数则在用户移动鼠标或手指时持续更新元素的位置。

此外，代码还监听了 keypress 事件，将按键值追加到 aid 元素的内容中，虽然这部分功能与主交互逻辑无关，但它展示了如何处理键盘输入。以下为代码实例：

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



  UI.appHeight = window.innerHeight;
```

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

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

      if(ev.touches){console.log("touches1"+ev.touches);
          Pointer.x = ev.touches[0].pageX ;
          Pointer.y = ev.touches[0].pageY ;
          console.log("Touch begin : "+"("+Pointer.x +"," +Pointer.y +")" ) ;
          $("bookface").textContent=  " 触 屏 事 件 开 始 ， 坐 标 ：
"+"("+Pointer.x+","+Pointer.y+")";
       }else{
         Pointer.x= ev.pageX;
         Pointer.y= ev.pageY;
         console.log("PointerDown at x: "+"("+Pointer.x +"," +Pointer.y +")" ) ;
         $("bookface").textContent=  " 鼠 标 按 下 ， 坐 标 ：
"+"("+Pointer.x+","+Pointer.y+")";
       }
      };
     let handleEnd = function(ev){
      Pointer.isDown=false;
      ev.preventDefault()
      //console.log(ev.touches)
      if(ev.touches){
        $("bookface").textContent= "触屏事件结束!";
```

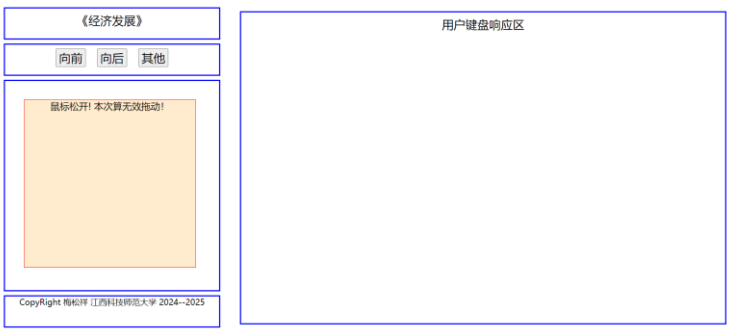
```
        if(Math.abs(Pointer.deltaX) > 100){
            $("bookface").textContent += ", 这是有效触屏滑动！"  ;
        }else{
           $("bookface").textContent += " 本次算无效触屏滑动！"  ;
           $("bookface").style.left = '7%' ;
        }
      }else{

        $("bookface").textContent= "鼠标松开!";
        if(Math.abs(Pointer.deltaX) > 100){
            $("bookface").textContent += ", 这是有效拖动！"  ;
        }else{
           $("bookface").textContent += " 本次算无效拖动！"  ;
           $("bookface").style.left = '7%' ;
           }
      }
     };
    let handleMoving = function(ev){
       ev.preventDefault();
      if (ev.touches){
        if (Pointer.isDown){
          console.log("Touch is moving");
          Pointer.deltaX = parseInt( ev.touches[0].pageX - Pointer.x );
          $("bookface").textContent="正在滑动触屏，滑动距离：" +Pointer.deltaX
+"px 。";
          $('bookface').style.left =  Pointer.deltaX + 'px' ;
        }
      }else{
        if (Pointer.isDown){
          console.log("Pointer isDown and moving");
          Pointer.deltaX = parseInt( ev.pageX - Pointer.x );
         $("bookface").textContent="正在拖动鼠标,距离:" +Pointer.deltaX +"px 。
";
          $('bookface').style.left =  Pointer.deltaX + 'px' ;
        }
      }
     };

     $("bookface").addEventListener("mousedown",handleBegin );
     $("bookface").addEventListener("touchstart",handleBegin );
     $("bookface").addEventListener("mouseup", handleEnd );
     $("bookface").addEventListener("touchend",handleEnd );
     $("bookface").addEventListener("mouseout", handleEnd );
```

```
    $("bookface").addEventListener("mousemove", handleMoving);
    $("bookface").addEventListener("touchmove", handleMoving);
    $("body").addEventListener("keypress", function(ev){
      $("aid").textContent += ev.key ;
    });
  } //Code Block  end
function $(ele){
        if (typeof ele !== 'string'){
           throw("自定义的$函数参数的数据类型错误，实参必须是字符串！");
           return
        }
        let dom = document.getElementById(ele) ;
          if(dom){
            return dom ;
          }else{
            dom = document.querySelector(ele) ;
            if (dom) {
                return dom ;
            }else{
                throw("执行$函数未能在页面上获取任何元素，请自查问题！");
                return ;
            }
          }
       } //end of $
```

T：虽然代码本身没有明确的测试部分，但在实际开发中，测试是非常重要的。测试应该包括检查响应式布局在不同屏幕尺寸下的表现，验证鼠标和触屏事件的正确处理，以及确保键盘输入被准确记录。可以通过手动测试和自动化测试框架来完成这些任务。

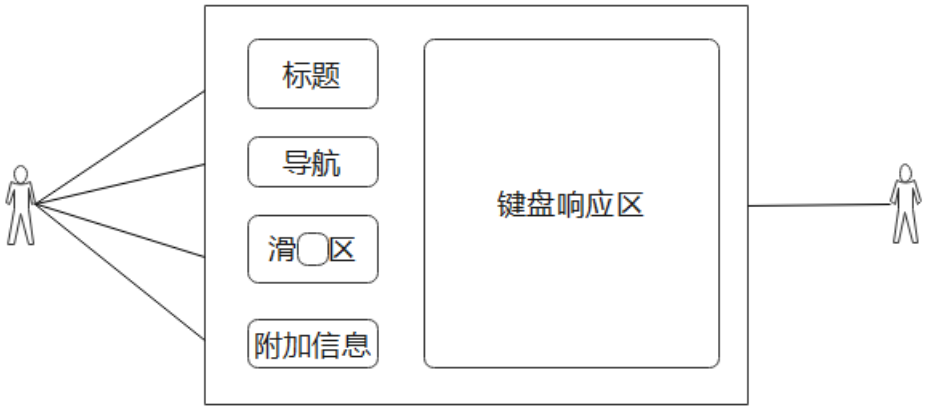
综上所述，这段代码示例展示了如何使用 JavaScript 和 DOM 操作来创建一个具有响应式设计和多点触控/鼠标输入处理能力的用户界面。通过遵循 ADIT 开发模式，开发者可以更好地组织和优化他们的代码，确保最终产品的质量和用户体验。

以下为实现图：



图表 7-2 触屏鼠标交互开发

8. UI 的个性化键盘交互控制的设计开发



图表 8-1 键盘交互控制用例图

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：

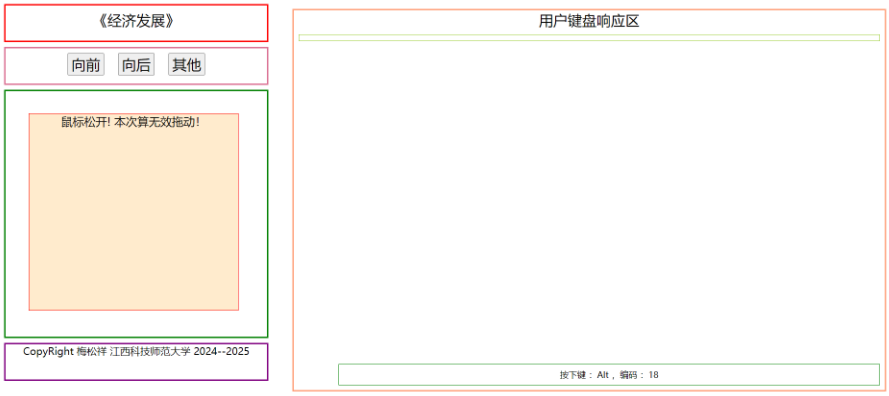
```
        $("body").addEventListener("keydown",function(ev){
      ev.preventDefault() ; //增加“阻止事件对象的默认事件后”，不仅 keypress 事件将不再响应，而且系统的热键，如“F5 刷新页面/Ctrl+R ”、“F12 打开开发者面板”等也不再被响应
       let k = ev.key;
       let c = ev.keyCode;
       $("keyStatus").textContent = "按下键 ： " + k + " ，"+ "编码 ： " + c;
      });
```

```
    $("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)
    });
```

以下为在 PC 端的实现图：



图表 9-1PC 端实现图

9. 谈谈本项目中的高质量代码

创建一个Pointer对象,践行MVC设计模式,设计一套代码同时对鼠标和触屏实现控制。

面向对象思想，封装，抽象，局部变量，函数式编程，逻辑。（围绕着抽象定义函数、代码块、模型设计以及降低全局变量的使用来写）

代码如下：

```
    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 = ev.touches[0].pageX ;
            Pointer.y = ev.touches[0].pageY ;
            console.log("Touch begin : "+"("+Pointer.x +"," +Pointer.y +")" ) ;
            $("bookface").textContent=  " 触 屏 事 件 开 始 ， 坐 标 ："+"("+Pointer.x+","+Pointer.y+")";
        }else{
          Pointer.x= ev.pageX;
          Pointer.y= ev.pageY;
          console.log("PointerDown at x: "+"("+Pointer.x +"," +Pointer.y +")" ) ;
          $("bookface").textContent=   " 鼠 标 按 下 ， 坐 标 ："+"("+Pointer.x+","+Pointer.y+")";
        }
       };
      let handleEnd = function(ev){
       Pointer.isDown=false;
       ev.preventDefault()
       //console.log(ev.touches)
       if(ev.touches){
         $("bookface").textContent= "触屏事件结束!";
         if(Math.abs(Pointer.deltaX) > 100){
             $("bookface").textContent += "，这是有效触屏滑动！"  ;
         }else{
            $("bookface").textContent += " 本次算无效触屏滑动！"  ;
            $("bookface").style.left = '7%' ;
         }
       }else{

         $("bookface").textContent= "鼠标松开!";
```

```
        if(Math.abs(Pointer.deltaX) > 100){
            $("bookface").textContent += "，这是有效拖动！"  ;
        }else{
           $("bookface").textContent += " 本次算无效拖动！"  ;
           $("bookface").style.left = '7%' ;
           }
      }
     };
    let handleMoving = function(ev){
       ev.preventDefault();
      if (ev.touches){
        if (Pointer.isDown){
          console.log("Touch is moving");
          Pointer.deltaX = parseInt( ev.touches[0].pageX - Pointer.x );
          $("bookface").textContent="正在滑动触屏，滑动距离：" +Pointer.deltaX
+"px 。";
          $('bookface').style.left =  Pointer.deltaX + 'px' ;
        }
      }else{
        if (Pointer.isDown){
          console.log("Pointer isDown and moving");
          Pointer.deltaX = parseInt( ev.pageX - Pointer.x );
         $("bookface").textContent="正在拖动鼠标,距离:" +Pointer.deltaX +"px 。
";
          $('bookface').style.left =  Pointer.deltaX + 'px' ;
        }
      }
     };
```

10. 用 gitBash 工具管理项目的代码仓库和 http 服务器

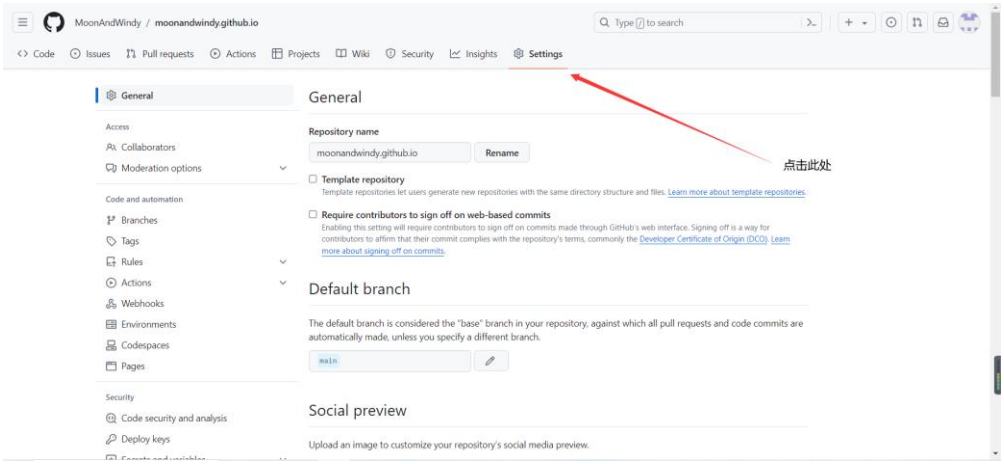
10.1 经典 Bash 工具介绍

当我们提到命令行时，实际上指的是 shell。Shell 是一个程序，它接收键盘输入的命令，并将它们传递给操作系统来执行。几乎所有的 Linux 发行版都提供了一个来自 GNU 项目的 shell 程序，名为 bash。这个名字是 bourne-again shell 的缩写，意指 bash 是对 sh 的增强替代，sh 是最初的 Unix shell 程序，由 Steve Bourne 编写。

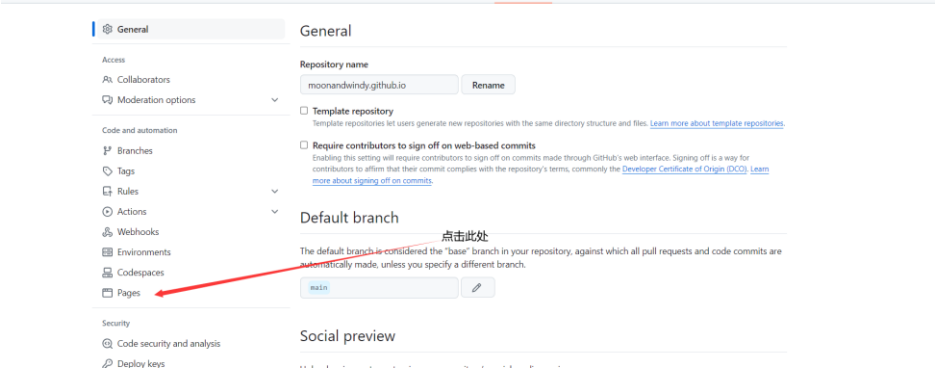
类似于 Windows，类 Unix 操作系统如 Linux 以一种被称为层次目录结构的方式来组织其文件。这意味着文件被组织成树状的目录结构（在其他系统中有时被称为文件夹），其中可能包含文件和其他子目录。文件系统的第一个目录被称

为根目录。根目录包含文件和子目录，而这些子目录又包含更多的文件和子目录，以此类推。

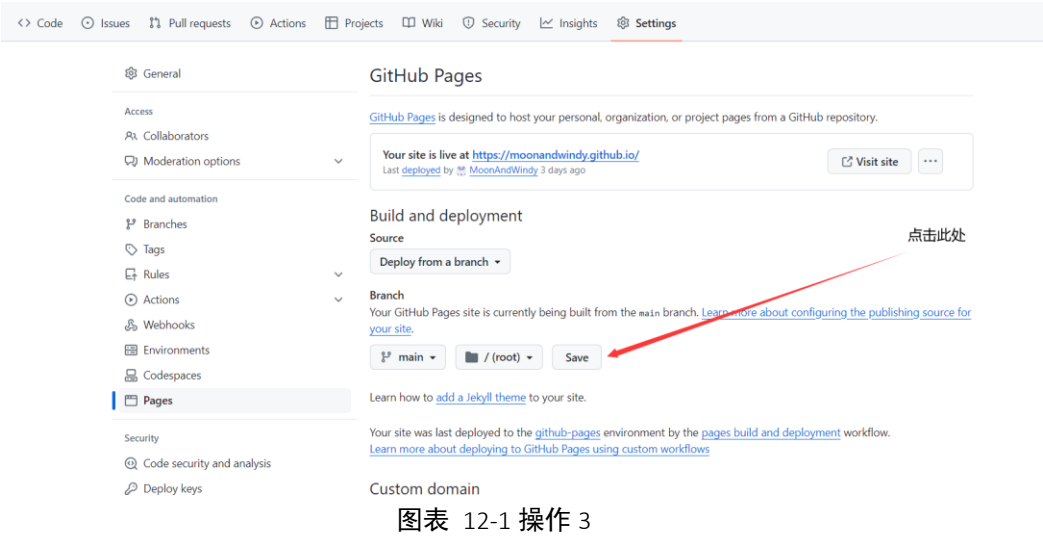
10.2 通过 gitHub 平台实现本项目的全球域名



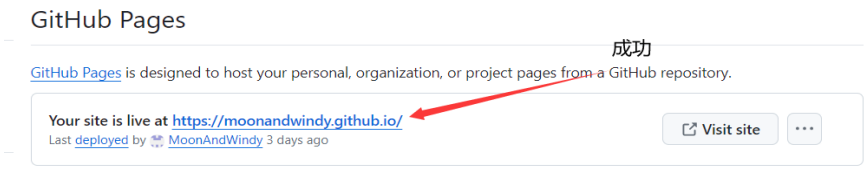
图表 10-1 操作 1



图表 11-1 操作 2

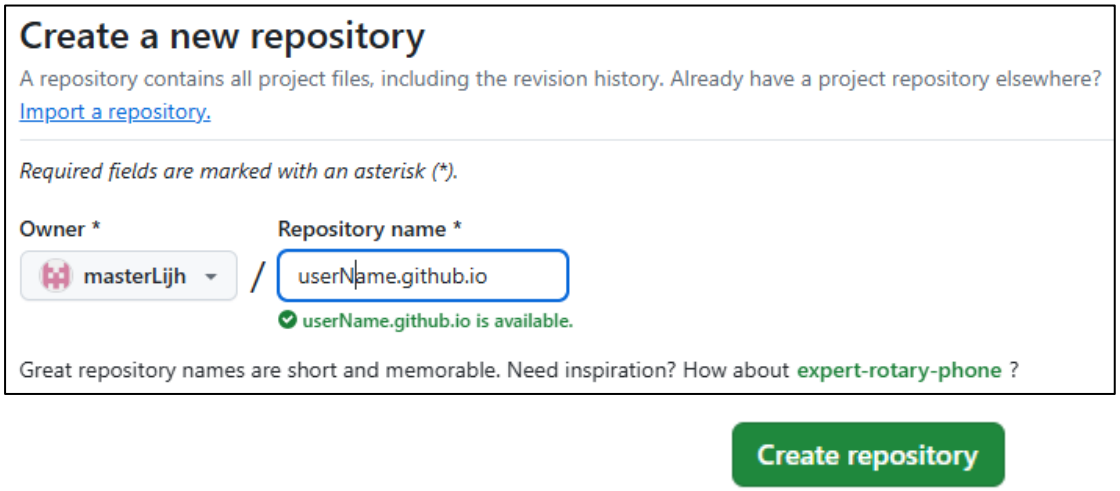


图表 12-1 操作 3



图表 13-1 操作 4

10.3 创建一个空的远程代码仓库



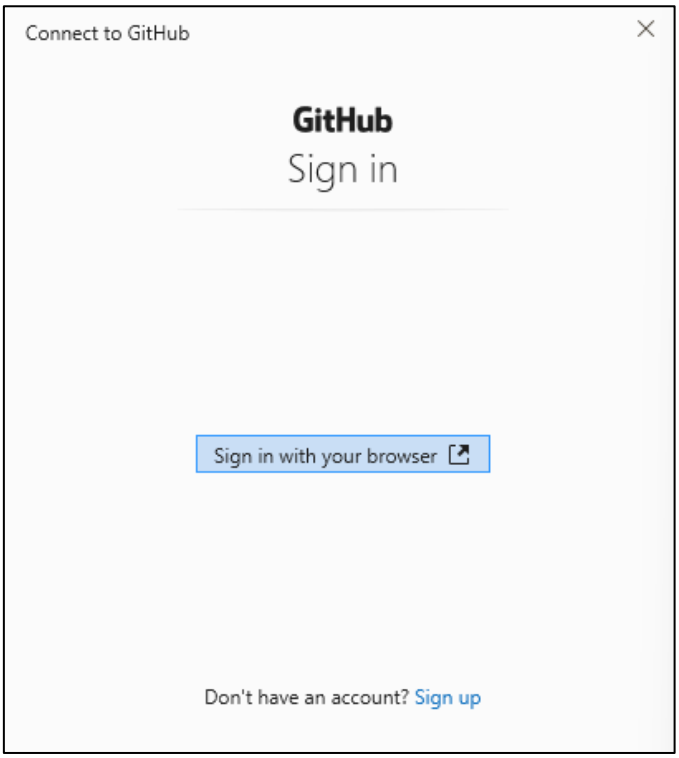
点击窗口右下角的绿色“Create repository”，则可创建一个空的远程代码仓库。

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

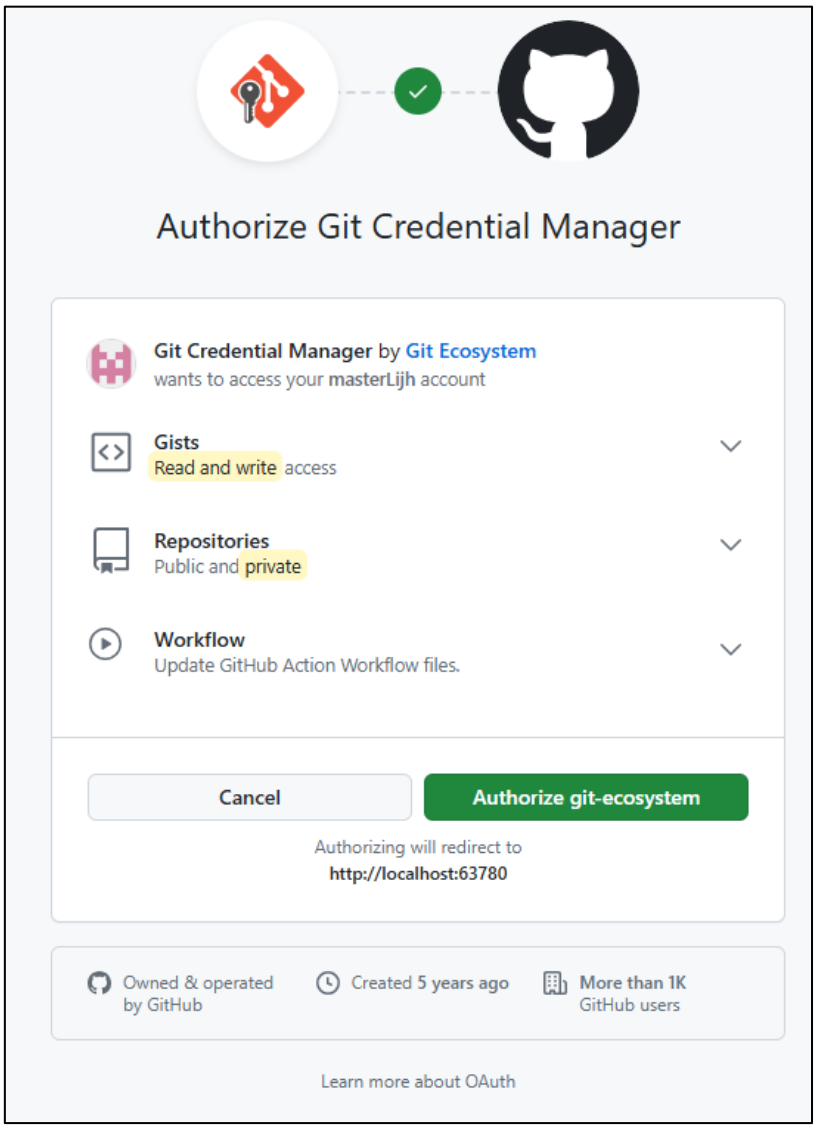
```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md
$ git init
$ git add README.md
$ git commit -m "这是我第一次把代码仓库上传至 gitHub 平台"
$ git branch -M main
$ git remote add origin
   https://github.com/MoonAndWindy/moonandwindy.github.io.git
$ git push -u origin main
```

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：



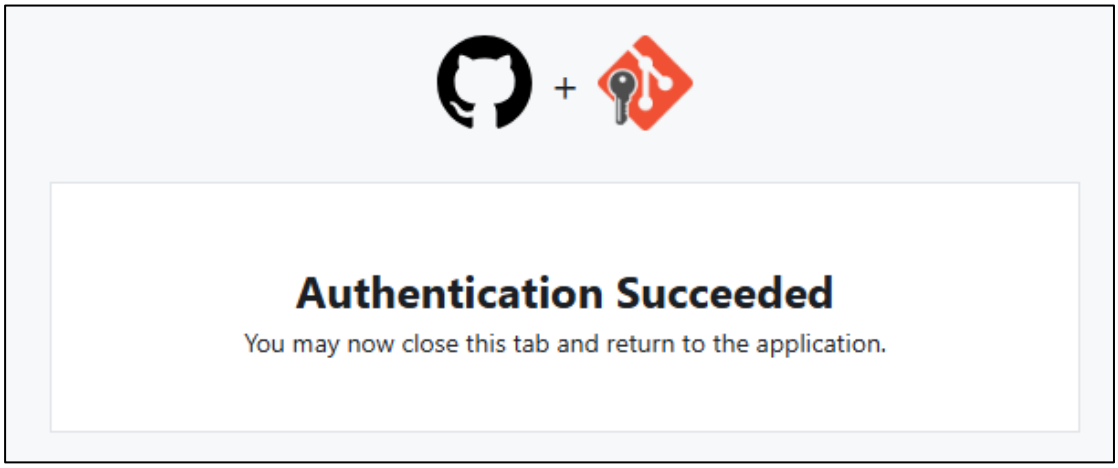
图表 14-1 示例图

再次确认授权 gitBash 拥有访问改动远程代码的权限，如下图所示：



图表 15-1 示例图

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



图表 16-1 示例图

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

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

图表 17-项目首页图

全文完成，谢谢！

参考文献：

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

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

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

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

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

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

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