

成绩:

江西科技师范大学

课程设计（论文）

题目（中文）：____ 基于 Web 客户端技术的个性化 UI 设计和编程 ____

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

院（系）：____ 元宇宙产业学院 ____

专 业：____ 计算机科学与技术 ____

学生姓名：____ 曾俊菁 ____

学 号：____ 20213639 ____

指导教师：____ 李健宏 ____

2024 年 6 月 18 日

目录

1.引言	1
1.1 毕设任务分析	1
1.2 研学计划	2
1.3 研究方法	3
1.3.1 文献法	3
1.3.2 模型研究法	3
2.技术总结和文献综述	4
2.1 Web 平台和客户端技术概述	4
2.1.1 历史记录	4
2.1.2 万维网联盟	4
2.1.3 Web 平台和 Web 编程	4
2.2 项目的增量式迭代开发模式	5
2.2.1 瀑布模型	5
2.2.2 增量模型	6
3.内容设计概要	7
3.1 分析和设计	7
3.2 项目的实现和编程	8
3.3 项目的运行和测试	9
3.4 项目的代码提交和版本管理	9
4.移动互联时代的 UI 开发初步——窄屏终端的响应式设计	11
4.1 分析与设计	11
4.2 项目的实现和编程	12
4.3 项目的运行和测试	13
4.4 项目的代码提交和版本管理	14
5.应用响应式设计技术开发可适配窄屏和宽屏的 UI	15
5.1 分析与设计	15
5.2 项目的实现和编程	16
5.3 项目的运行和调试	18
5.4 项目的代码提交和版本管理	19
6.个性化 UI 设计中对鼠标交互的设计开发	21
6.1 分析与设计	21
6.2 项目的实现和编程	22
6.3 项目的运行和调试	23
6.4 项目的代码提交和版本管理	24
7.对触屏和鼠标的通用交互操作的设计开发	25
7.1 分析与设计	25
7.2 项目的实现和编程	26
7.3 项目的运行和调试	27
7.4 项目的代码提交和版本管理	27

8. UI 的个性化键盘交互控制的设计开发	29
8.1 分析与设计	29
8.2 项目的实现和编程	30
8.3 项目的运行和调试	31
8.4 项目的代码提交和版本管理	32
9. 用 gitBash 工具管理项目的代码仓库和 http 服务器	34
9.1 经典 Bash 工具介绍	34
9.2 通过 gitHub 平台实现本项目的全球域名	34
9.3 创建一个空的远程代码仓库	35
9.4 设置本地仓库和远程代码仓库的链接	35
参考文献	36

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 曾俊菁

摘要：在过去十年，HTML5 作为 Web 标准，在软件开发领域取得了显著进展，特别是在跨平台和开源特性方面。本研究聚焦于 HTML5 在现代软件开发中的应用，通过基于 HTML5 的 Web 客户端技术开发个性化用户界面（UI）应用程序。项目采用 HTML、CSS 和 JavaScript，实现内容结构、外观设计和交互功能。响应式设计理念使应用程序能适应不同用户屏幕，面向对象编程思想，如统一控制鼠标和触屏的 pointer 模型，提升了代码质量。工程管理上，项目采用增量式开发模式，通过六次重构，包括分析、设计、实现和测试，确保了开发过程的高效和愉快。使用 Git 进行版本管理，代码经过六次重构和多次提交，最终通过 gitbash 工具上传至 GitHub 平台。GitHub 的 HTTP 服务器功能使 UI 应用程序得以全球部署，用户可通过网址或二维码访问。项目展示了 HTML5 的应用潜力，增量式开发和响应式设计在提升开发效率和用户体验方面的重要性。开源和分享代码，推动了学术交流和技术进步，为未来相关领域的研究和实践提供了参考。

关键词：客户端技术；响应式设计；增量式开发；Git

1.引言

在当今数字化时代，随着互联网技术的飞速发展，HTML5 作为一种创新的 Web 标准，已成为软件开发领域的关键技术之一。它以其跨平台兼容性和开源特性，为构建现代 Web 应用程序提供了强大的支持。本研究旨在探讨 HTML5 在现代软件开发中的应用，特别是其在个性化用户界面（UI）开发中的关键作用。通过采用 HTML、CSS 和 JavaScript 等核心技术，本项目不仅实现了内容结构的优化、外观设计的个性化，还增强了交互功能的丰富性。

1.1 毕设任务分析

在计算机专业，毕业论文和毕业设计是两个紧密相连的环节，它们共同构成了我们学术成果的重要部分。然而，当前的“毕业论文（设计）”这一表述方式，可能无法完全准确地反映出软件开发方向毕设的实际流程。

在软件开发中，我会投入大量的时间和精力进行设计与实现，通过编写代码、测试与迭代，最终完成一个具有实际应用价值的软件作品。这个设计过程，不仅锻炼了我的编程技能，也让我对软件工程有了更深入的理解。

在毕业论文中，我会详细阐述系统功能模块、设计思路、实现方法以及遇到的挑战与解决方案，同时也会对设计成果进行客观的评价和展望。

1.2 研学计划

毕业论文（设计）进度计划			
序号	工作内容	起讫日期	备注
一	定选题	2024 年 10 月 7 日-10 月 10 日	完成指导教师下达的任务书
二	开题答辩	2024 年 10 月 14 日-10 月 15 日	进行 2025 届 2021 级毕业论文开题答辩，制作开题报告答辩 PPT。
三	文献综述	2024 年 10 月 16 日-10 月 31 日	完成文献综述的撰写，文献综述要求 3000 字。
四	论文一稿	2024 年 11 月 1 日-11 月 30 日	先撰写论文框架
五	论文二稿	2024 年 12 月 1 日-12 月 31 日	再撰写论文内容
六	中期检查	2025 年 2 月 1 日-2 月 7 日	项目作品完成，论文完成一半的进度。
七	论文三稿	2025 年 1 月 4 日-1 月 31 日	按照学术规范修改论文格式
八	论文定稿	2025 年 2 月 26 日-4 月 12 日	做好中国知网学术不端审核检测自查工作
九	毕业论文（设计）中国知网学术不端审核检测自查工作	2025 年 4 月 12 日-14 日	提交中国知网重复率检测报告，踩实本科毕业生毕业论文（设计）中国知网学术不端审核检测自查工作。
十	本科毕业生毕业论文（设计）中国知网学术不端审核检测	2025 年 4 月 15 日-4 月 19 日	进行本科毕业生毕业论文（设计）中国知网学术不端审核检测，提交中国知网重复率自查达标的论文 Word 文档，重复率 $\leq 15\%$ 。

1.3 研究方法

1.3.1 文献法

文献法主要是通过查阅、分析和总结相关领域的文献资料，来获取研究所需的理论基础、研究现状、研究方法以及可能的研究方向等信息。在计算机专业中，这种方法尤其重要，因为计算机技术发展迅速，新的理论、技术和方法层出不穷，只有通过深入研究和分析相关文献，才能确保研究的创新性和实用性。

在撰写毕业论文过程中，我们需要引用大量的文献资料来支持自己的观点和结论。通过查阅和分析相关领域的文献，我们可以了解该领域的学术规范和引用格式等要求，从而确保自己的论文和设计符合学术规范和要求。

1.3.2 模型研究法

作为一名计算机专业的学生，我深刻体会到在写毕设的过程中，模型研究法的重要性和便捷性。这种方法让我能够将复杂的软硬件系统从编程的视角进行抽象化理解。这些软硬件对象在我脑海中逐渐转化为一系列的模型，而这些模型正是我们编写代码时所要考虑和解决的问题。

在面向对象编程（OOP）的分析和开发过程中，我尝试将毕业设计中的实际问题转化为不同层级的模型。模型研究法不仅具有高度的抽象性，而且与具体的编程语言无关，它更多地关注于问题的本质和解决方案的结构。为了更清晰地表达这些模型，我尝试使用国际标准 UML（统一建模语言）来建立抽象模型。尽管 UML 语言相对抽象，设计准确有一定难度，但它与 OOP 程序设计方法的目标是一致的，即在不同层面分析和表述问题。

在我看来，对于本科生而言，采用模型研究法时，一个更为实用的做法是先通过编写代码来建立模型并进行研究。一旦代码能够正常运行并达到预期效果，我们可以再利用 UML 语言绘制出这些模型，作为代码的文档资料。这样做的好处在于，我们能够通过具体的编程实践来验证和完善模型设计，而 UML 语言则为我们提供了一个更加直观和系统的表达方式。此外，通过 OOP 语言和代码运行环境对模型进行运行和调试，我们还可以倒推出模型设计中可能存在的问题和缺陷，从而进行针对性的改进。

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。

2.1.1 历史记录

1989 年,蒂姆·伯纳斯-李爵士发明了万维网(见最初的提案)。他在 1990 年 10 月创造了“万维网”一词,并写下了第一个万维网服务器“httpd”和第一个客户端程序(一个浏览器和编辑器)“世界万维网”。

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,后来成为了 Web 的主要发布格式。他对 URI、HTTP 和 HTML 的最初规范随着网络技术的传播,在更大的圈子中得到了改进和讨论。

2.1.2 万维网联盟

1994 年,在许多公司的敦促下,决定成立万维网联盟。蒂姆·伯纳斯-李爵士开始领导网络联盟团队的基本工作,以培养一个一致的架构,以适应网络标准的快速发展,以构建网站、浏览器和设备,以体验网络所提供的一切。

在创立万维网联盟的过程中,蒂姆·伯纳斯-李爵士创建了一个同行社区。Web 技术已经如此之快,以至于组装一个组织来协调 Web 标准至关重要。蒂姆接受了麻省理工学院的邀请,来主持 W3C。他从一开始就要求 W3C 拥有全球的足迹。

2.1.3 Web 平台和 Web 编程

让我们从对网络的简要描述开始,它是万维网的缩写。大多数人说“是网络”而不是“万维网”,我们会遵循这个惯例。Web 是一个文档的集合,被称为网页,它们由世界各地的计算机用户(在大部分时间内)共享。不同类型的网页可以做不同的事情,但至少,它们都能在电脑屏幕上显示内容。我们所说的“内容”是指文本、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。所有的工具都与核心语言 HTML 一起工作，所以几乎所有的 Web 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，所有的深入。这三种技术被认为是客户端网络编程的支柱。使用客户端 Web 编程，所有网页计算都在终端用户的计算机（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

软件工程视角中，软件生命周期中主要包括开发和维护两项工作，其中在我们的本科毕业设计中仅涉及开发过程。软件的开发过程包括四个阶段：分析 (analysis)、设计 (design)、实现 (implementation) 和测试 (testing)，下文简称为 ADIT。而开发过程，有经典的两种模型：瀑布模型和增量模型。

2.2.1 瀑布模型

1970 年，温斯顿·罗伊斯提出了著名的“瀑布模型”，直到 80 年代早期，它一直是唯一被广泛采用的软件开发模型。瀑布模型的核心思想是按工序将问题化简，采用结构化的分析与设计方法将逻辑实现与物理实现分开，将软件生命周期的各项活动规定为按固定顺序而连接的若干阶段工作，形成了自上而下、相互衔接的固定次序，形如瀑布流水，逐级下落，最终得到软件产品（如图 2-1 所示）。

瀑布模型的特点是只有当一个任务完成，才能开始下一阶段的工作，每个阶段都会产生循环反馈，如果有信息未被覆盖或者发现问题，则应返回上一阶段并进行适当的修改。瀑布模型是一种严格线性的、按阶段顺序的、逐步细化的过程模型，可以解决城市轨道交通项目价值分摊过程中普遍存在的分摊随意、时序混

乱问题，为按时序制定符合逻辑的价值分摊方法，确保待摊费用分摊不遗漏、分摊过程的连贯性、财务价值的准确性奠定了理论基础。

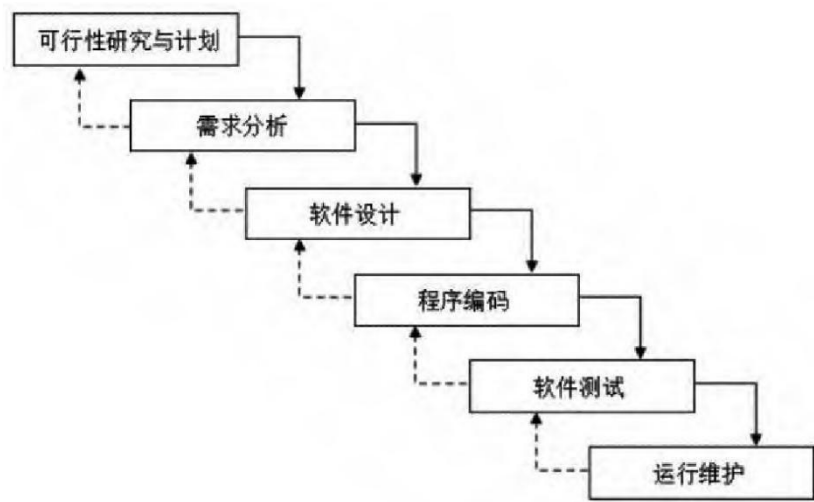


图 2-1 瀑布模型

2.2.2 增量模型

在增量式的软件开发模型中，开发者分一系列步骤进行开发。我们首先完成了整个系统的简化版本，这个版本表示整个系统，但不包括系统内部的详细设计。图 2-2 表达了增量式的软件开发模型的概念。在后续版本的开发中，通过分析和设计添加了更多的系统软件的细节，然后再次进行系统部署的开发和测试^[5]。

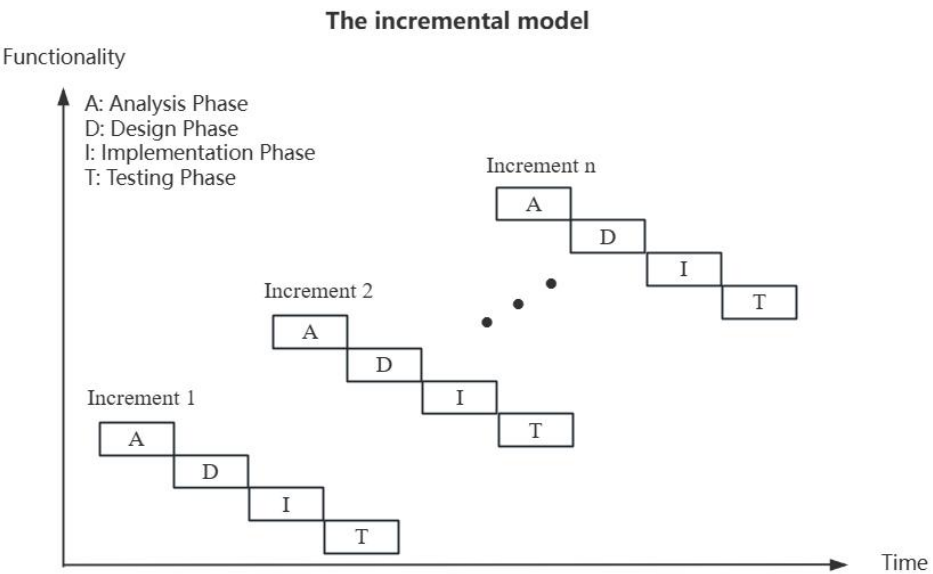


图 2-2 增量模型

3.内容设计概要

3.1 分析和设计

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

项目的用例图如图 3-1 所示：

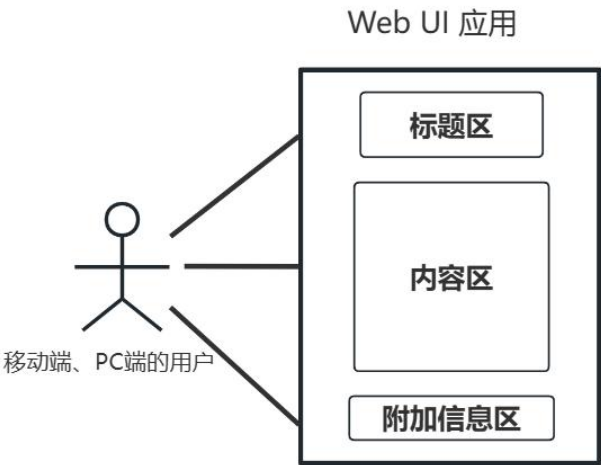


图 3-1 “三段式”设计用例图

项目的 DOM 树如图 3-2 所示：

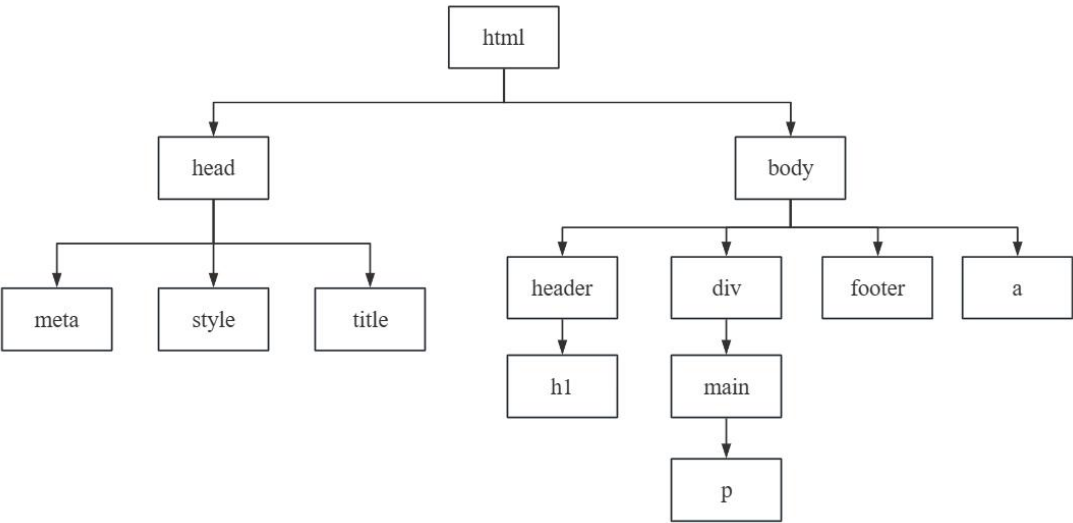


图 3-2 “三段式”设计 DOM 树

3.2 项目的实现和编程

项目的 HTML 代码如代码块 3-1 所示：

```
<header>
    《基于 SSM 的家庭财务管理系统设计与实现》
</header>
<main>
    本研究旨在设计并实现一个基于 SSM 框架的家庭财务管理系统，以提高
    家庭理财的效率和准确性。系统将提供用户友好的界面，使家庭成员能够轻松记
    录和管理日常收支、制定财务预算、进行财务分析，并跟踪心愿单等财务目标。
    通过采用 Spring、Spring MVC 和 MyBatis 等技术，本系统将实现数据的高效管
    理和操作的安全性，同时提供灵活的报表功能，帮助用户做出更明智的财务决策，
    优化家庭财务状况。
</main>
<footer>
    Copyright 曾俊菁 江西科技师范大学 2025-2025
</footer>
```

代码块 3-1

项目的 CSS 核心代码如代码块 3-2 所示：

```
* {
    margin: 10px;
    text-align: center;
}
main {
    height: 70%;
    position: relative;
    border: 3px solid gray;
}
#aid {
    position: absolute;
    border: 3px solid black;
    top: 0;
}
#tvface {
    width: 80%;
    height: 80%;
    margin: auto;
}
```

代码块 3-2

3.3 项目的运行和测试

项目的运行和测试至少要通过二类终端，本文此处仅给出 PC 端用 Edge 浏览器打开项目的结果，如下图 3-2 所示。



图 3-2 PC 端运行效果图

由于本项目的阶段性文件已经上传 github 网站，移动端用户可以通过扫描图 3-3 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 3-3 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。

一、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 曾俊菁  
$ git config user.email 1606681131@qq.com  
$ touch phase1.html phase1.css
```

二、编写好 phase1.html 和 phase1.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
```

三、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”式的内容设计概要开发  
[master c9dfea7] 项目第一版：“三段论”式的内容设计概要开发  
3 files changed, 3 insertions(+), 3 deletions(-)
```

四、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
```

五、gitbash 反馈代码的仓库日志如下所示：

```
$ git log  
commit c9dfea7649fccd1071555f4c4fb1ae5bbcdebe4f (HEAD -> master)  
Author: 曾俊菁 <1606681131@qq.com>  
Date:   wed Jun 5 20:52:34 2024 +0800
```

项目第一版：“三段论”式的内容设计概要开发

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 分析与设计

在本项目的第二次迭代中，继续沿用了“三段论”式的设计理念，这一理念通过将网页内容划分为清晰的结构化部分，增强了用户的浏览体验。此外，为了应对移动设备使用的日益增长，特别引入了响应式设计，确保项目能够在窄屏终端上提供优化的显示效果。响应式设计的核心在于使用流体布局、灵活的网格和可伸缩的元素，这些技术使得网页能够智能地根据屏幕大小和分辨率调整布局，从而在各种设备上都能保持良好的可用性和可读性。

项目的用例图如图 4-1 所示：

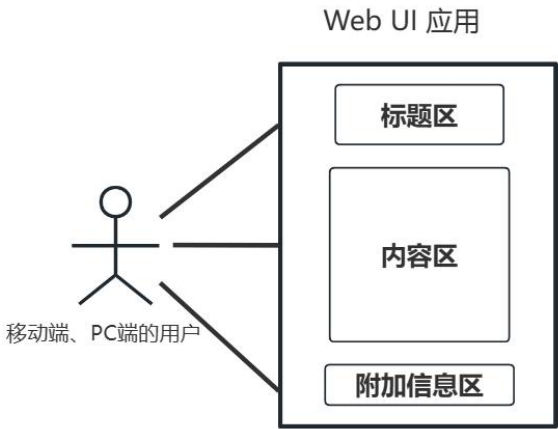


图 4-1 窄屏终端的响应式设计用例图

项目的 DOM 树如图 4-2 所示：

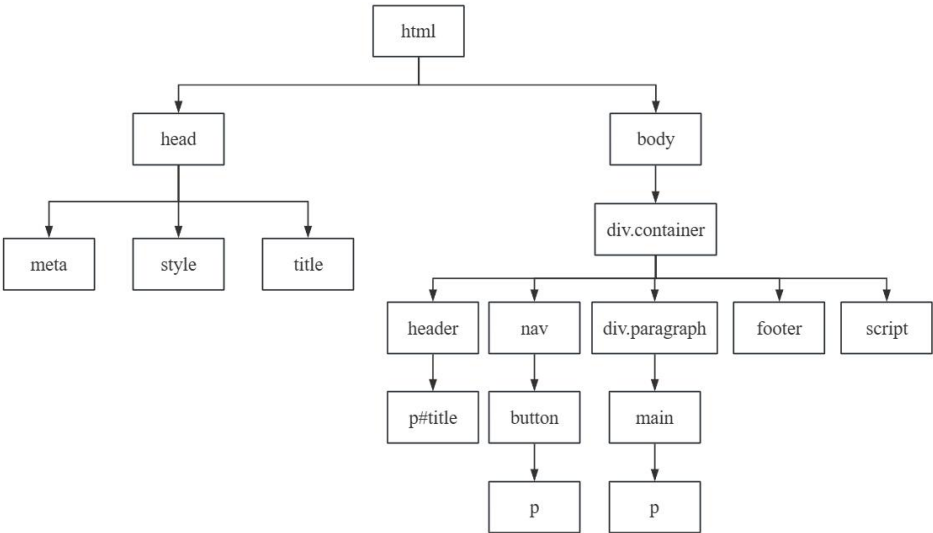


图 4-2 窄屏终端的响应式设计 DOM 树

4.2 项目的实现和编程

项目的核心 CSS 代码如代码块 4-1 所示：

```
body {  
  min-height: 100vh;  
  font: 16px/1.5 Arial, sans-serif;  
  margin: 0;  
  padding: 0;  
}  
header, nav, main, footer {  
  text-align: center;  
  margin: 10px 0;  
  padding: 20px;  
  box-sizing: border-box;  
}  
nav {  
  display: flex;  
  justify-content: space-around;  
}  
main {  
  height: auto;  
}  
button {  
  padding: 8px 15px;  
  margin: 5px;  
}  
p {  
  font-size: 1rem;  
  text-indent: 2em;  
  margin-bottom: 1em;  
  text-align: left;  
}
```

代码块 4-1

项目的 JS 代码实现了一个响应式网页布局的功能。首先，它定义了一个名为 UI 的对象，用来存储应用的宽度和高度。接下来，脚本通过修改 width 和 height 属性来设置 body 元素的尺寸。body 元素的宽度是应用宽度减去 baseFont 的 1 倍，高度是应用高度减去 baseFont 的 4 倍，这样做可能是为了在布局中留出一些边距或空间。通过这种方式，body 元素的尺寸会随着窗口大小的变化而动态调整，从而实现一个响应式的网页布局，使得网页内容能够适应不同大小的屏幕。

项目的核心 JS 代码如代码块 4-2 所示：

```
var UI = {};  
UI.appWidth = window.innerWidth > 800 ? 800 : window.innerWidth ;  
UI.appHeight = window.innerHeight;  
const LETTERS = 22 ;  
const baseFont = UI.appWidth / LETTERS;  
//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 - 1*baseFont + "px";  
document.body.style.height = UI.appHeight - 4*baseFont + "px";
```

代码块 4-2

4.3 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 Edge 浏览器打开项目的结果，如下图 4-3 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 4-4 的二维码，运行测试本项目的第一次开发的阶段性效果。

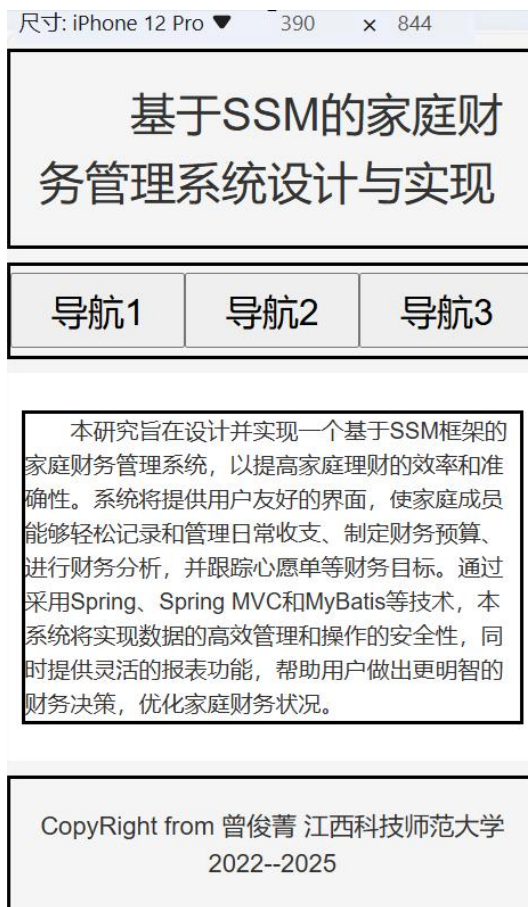


图 4-3 PC 端运行效果图



图 4-4 移动端二维码

4.4 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。

一、进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /  
$ mkdir webUI  
$ cd webUI  
$ git config user.name 曾俊菁  
$ git config user.email 1606681131@qq.com  
$ touch phase2.html phase2.css
```

二、编写好 phase2.html 和 phases2.css 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add phase2.html phase2.css  
$ git commit -m ""
```

三、成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m项目第二版：窄屏终端的响应式设计  
[master 75c92f3] 项目第二版：窄屏终端的响应式设计  
4 files changed, 165 insertions(+), 171 deletions(-)  
rewrite phase/phase1.html (67%)  
rewrite phase/phase2.html (67%)  
rewrite phase2.css (100%)
```

四、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
```

五、gitbash 反馈代码的仓库日志如下所示：

```
$ git log  
commit 75c92f3ee318840efe70b57d42e6c8cc85534ba7 (HEAD -> master)  
Author: 曾俊菁 <1606681131@qq.com>  
Date: Wed Jun 5 22:12:57 2024 +0800  
  
项目第二版：窄屏终端的响应式设计
```

5.应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.1 分析与设计

在本项目的第三次开发迭代中，在"三段论"式设计的基础上进行了扩展，引入了导航栏设计和用户键盘响应区，同时对用户界面进行了精心的重新设计。这次迭代不仅优化了用户交互体验，还增强了界面在不同屏幕尺寸下的自适应能力，确保了无论是在窄屏还是宽屏模式下，用户都能享受到美观且实用的界面布局。通过采用先进的响应式设计技术，实现了跨设备兼容性，使得项目能够在各种操作系统和浏览器上稳定运行，同时，也注重了用户体验的持续改进，根据用户反馈对界面设计和交互逻辑进行了细致的调整。

项目的用例图如图 5-1 所示：

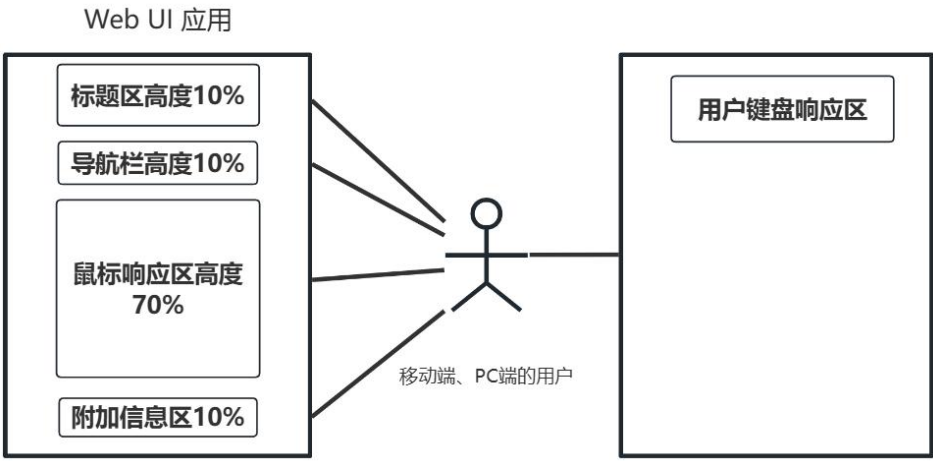


图 5-1 适配窄屏和宽屏的 UI 用例图

项目的 DOM 树如图 5-2 所示：

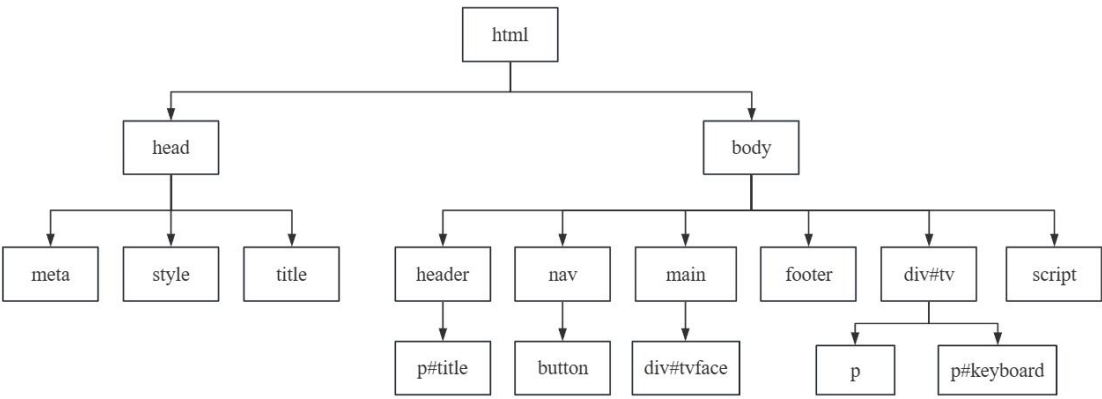


图 5-2 适配窄屏和宽屏设计的 DOM 树

5.2 项目的实现和编程

项目的核心 CSS 代码如代码块 5-1 所示：

```
*{
    margin: 10px;
    text-align: center;
}
header{
    border: 3px solid gray;
    height: 10%;
    font-size: 1em;
}
nav{
    border: 3px solid gray;
    height: 10%;
}
main{
    border: 3px solid gray;
    height: 70%;
    font-size: 0.8em;
    position: relative;
}
footer{
    border: 3px solid gray;
    height: 10%;
    font-size: 0.7em;
}
body{
    position: relative;
}
```

代码块 5-1

项目的 JavaScript 代码实现了一个交互式的用户界面元素，用于响应鼠标和触摸事件。它定义了三个函数：**updateText**, **handleBegin**, 和 **handleEnd**，以及一个 **handleMoving** 函数来处理事件的移动阶段。

updateText 函数用于更新页面上某个元素的文本内容，这个元素的文本会根据是鼠标事件还是触摸事件来显示不同的信息，并显示事件触发时的坐标。相关代码如代码块 5-2 所示：

```
function updateText(x, y, isTouch) {
    var text = isTouch ? "触屏事件开始, 坐标: " : "鼠标按下, 坐标: ";
    bookface.textContent = text + "(" + x + "," + y + ")";
}
```

代码块 5-2

`handleBegin` 函数在用户开始触摸屏幕或按下鼠标时被调用, 它会设置一个全局变量 `Pointer.isDown` 为 `true`, 表示用户已经开始与界面交互。这个函数会检查事件对象 `ev` 是否包含触摸列表 `ev.touches`, 如果是触摸事件, 它会记录第一个触摸点的坐标, 并调用 `updateText` 函数显示触摸开始的信息; 如果是鼠标事件, 它会记录鼠标的坐标, 并显示鼠标按下的信息。相关代码如代码块 5-3 所示:

```
function handleBegin(ev) {
    Pointer.isDown = true;
    if (ev.touches) {
        Pointer.x = ev.touches[0].pageX;
        Pointer.y = ev.touches[0].pageY;
        updateText(Pointer.x, Pointer.y, true);
    } else {
        Pointer.x = ev.pageX;
        Pointer.y = ev.pageY;
        updateText(Pointer.x, Pointer.y, false);
    }
}
```

代码块 5-3

`handleEnd` 函数在用户停止触摸屏幕或释放鼠标时被调用。这个函数会将 `Pointer.isDown` 设置为 `false`, 表示用户已经停止与界面的交互。它还会检查用户移动的距离是否超过 100 像素, 如果是, 则认为这是一个有效的滑动或拖动动作, 并在文本内容中反映这一点。如果移动距离不足 100 像素, 则认为这是一个无效的滑动或拖动, 并重置界面元素的位置。相关代码如代码块 5-4 所示:

```
function handleEnd(ev) {
    Pointer.isDown = false;
    var isTouch = !!ev.touches;
    var isValidMove = Math.abs(Pointer.deltaX) > 100;
    bookface.textContent = "触屏/鼠标事件结束!" + (isValidMove ? "这是有效" +
(isTouch ? "触屏滑动" : "拖动") + "!" : "本次算无效" + (isTouch ? "触屏滑动" : "拖动") + "!");
    if (!isValidMove) bookface.style.left = '7%';
}
```

```
}
```

代码块 5-4

`handleMoving` 函数在用户在屏幕上滑动手指或拖动鼠标时被调用。如果 `Pointer.isDown` 为 `true`，表示用户正在与界面交互，这个函数会阻止默认的滚动行为，计算并更新用户滑动或拖动的距离，并实时更新界面元素的位置和文本内容，以反映当前的滑动状态。相关代码如代码块 5-5 所示：

```
function handleMoving(ev) {  
    if (Pointer.isDown) {  
        ev.preventDefault();  
        Pointer.deltaX = parseInt(ev.touches ? ev.touches[0].pageX - Pointer.x :  
ev.pageX - Pointer.x);  
        bookface.textContent = "正在" + (ev.touches ? "滑动触屏" : "拖动鼠标") + ",  
滑动距离: " + Pointer.deltaX + "px。";  
        bookface.style.left = Pointer.deltaX + 'px';  
    }  
}
```

代码块 5-5

5.3 项目的运行和调试

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 Edge 浏览器打开项目的结果，分别展示宽屏和窄屏模式下的项目界面，如下图 5-2、图 5-3 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 5-4 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 5-2 项目宽屏界面设计



图 5-3 项目窄屏界面设计



图 5-4 移动端二维码

5.4 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。

一、进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ mkdir webUI
$ cd webUI
$ git config user.name 曾俊菁
$ git config user.email 1606681131@qq.com
$ touch phase3.html phase3.css phase3.js
```

二、编写好 phase3.html、phases3.css 和 phase3.js 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add phase3.html phase3.css phase3.js
$ git commit -m 项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的 UI
```

三、成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的UI
[master ebf1282] 项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的UI
1 file changed, 5 insertions(+), 60 deletions(-)
```

四、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
```

五、gitbash 反馈代码的仓库日志如下所示：

```
$ git log
commit ebf1282db960ce8a9dd496362109dfb59627808f (HEAD -> master)
Author: 曾俊菁 <1606681131@qq.com>
Date: Thu Jun 6 10:05:05 2024 +0800
```

项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的UI

6.个性化 UI 设计中对鼠标交互的设计开发

6.1 分析与设计

在本次项目的第四次迭代中，我专注于个性化 UI 设计的一个关键方面——鼠标交互的设计开发。这一阶段的工作不仅增强了用户界面的互动性，而且提升了用户体验的直观性和动态性。我通过对鼠标交互的深入分析，开发了一套能够实时反映鼠标位置变化的系统，使用户能够在 PC 端通过视觉反馈直观地观察到鼠标的坐标变化。

项目用例图如图 6-1 所示：

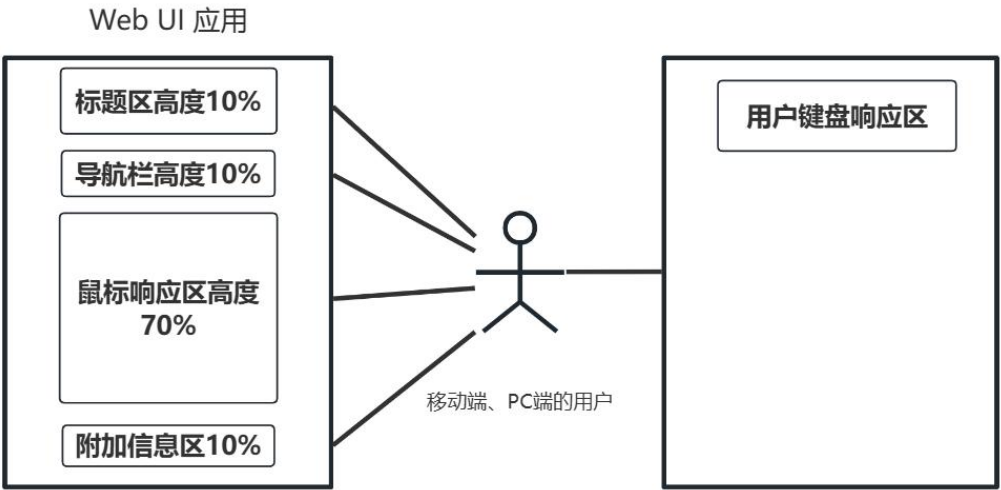


图 6-1 鼠标交互的设计用例图

项目的 DOM 树如图 6-2 所示：

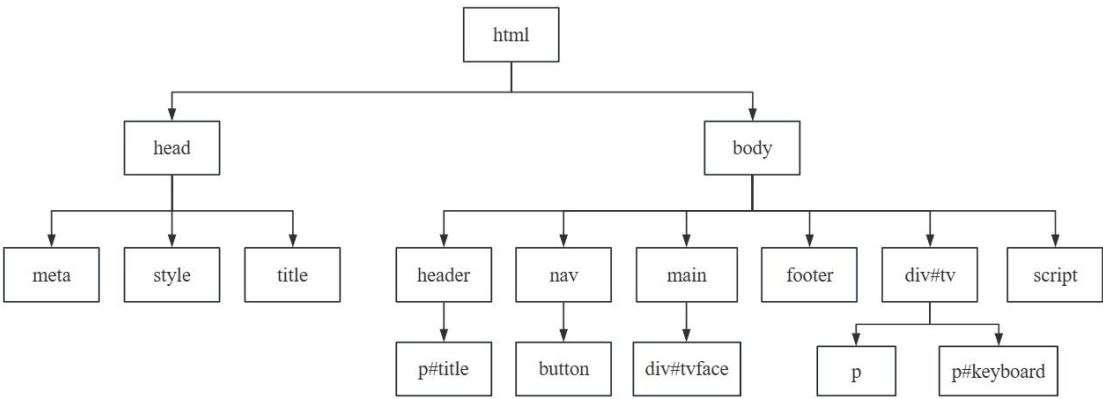


图 6-2 鼠标交互的设计 DOM 树

6.2 项目的实现和编程

此项目用于在网页上实现对鼠标和键盘事件的监听和响应，代码逻辑如下：定义一个名为 `mouse` 的对象，用来存储鼠标的状态和位置信息；使用 `addEventListener` 方法为页面上的元素添加了三个事件监听器；为 `body` 元素添加了一个 `"keypress"` 事件监听器，当有按键被按下时触发。触发时获取按键的值和编码，并在页面上显示按键信息。

代码中使用了 `$` 符号，通常表示使用了 `jQuery` 库来简化 `DOM` 操作。例如，`$("#bookface")` 和 `$("#keyboard")` 是选择页面上 `id` 分别为 `bookface` 和 `keyboard` 的元素。`ev` 是事件对象，包含了事件相关的信息，如 `ev.pageX` 和 `ev.pageY` 分别表示鼠标在页面上的 `x` 和 `y` 坐标，`ev.key` 和 `ev.keyCode` 分别表示按键的字符和编码。

项目的核心 JS 代码如代码块 6-1 所示：

```
//尝试对鼠标设计 UI 控制
var mouse={};
mouse.isDown= false;
mouse.x= 0;
mouse.deltaX=0;
$("#bookface").addEventListener("mousedown",function(ev){
    let x= ev.pageX;
    let y= ev.pageY;
    console.log("鼠标按下了，坐标为: "+"("+x+","+y+")");
    $("#bookface").textContent= "鼠标按下了，坐标为: "+"("+x+","+y+")";
});
$("#bookface").addEventListener("mousemove",function(ev){
    let x= ev.pageX;
    let y= ev.pageY;
    console.log("鼠标正在移动，坐标为: "+"("+x+","+y+")");
    $("#bookface").textContent= "鼠标正在移动，坐标为: "+"("+x+","+y+")";
});
$("#bookface").addEventListener("mouseout",function(ev){
    //console.log(ev);
    $("#bookface").textContent= "鼠标已经离开";
});
$("#body").addEventListener("keypress",function(ev){
    let k = ev.key;
    let c = ev.keyCode;
```

```
let s = "按键是: ";
let s1 = "编码是: ";
$("keyboard").textContent=s+k+" "+s1+c;
});
```

代码块 6-1

6.3 项目的运行和调试

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 Edge 浏览器打开项目的结果，如下图 6-3 所示。

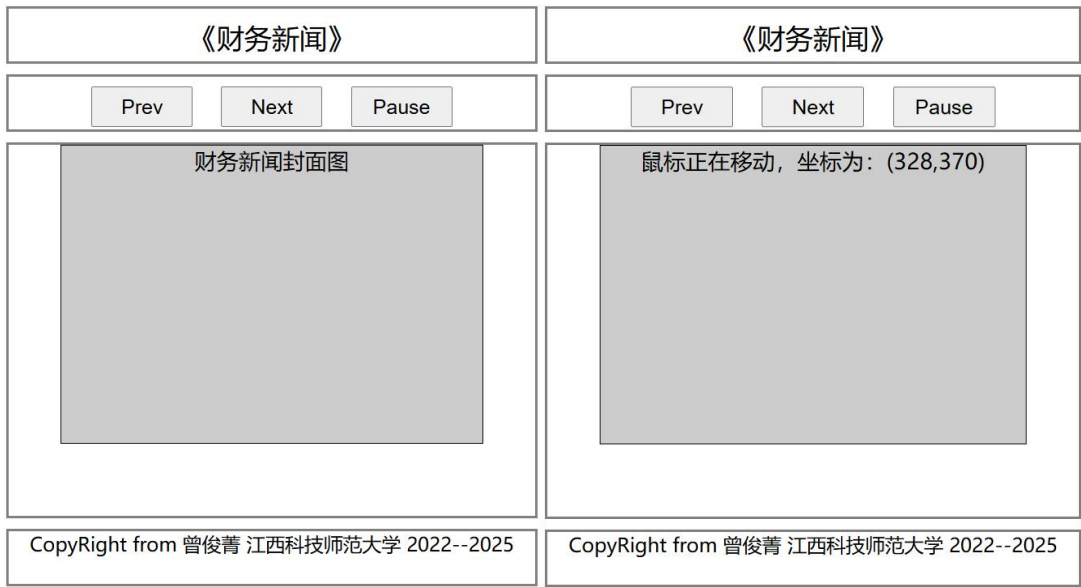


图 6-3 PC 端运行效果图

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 6-4 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 6-4 移动端二维码

6.4 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。

一、进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /  
$ mkdir webUI  
$ cd webUI  
$ git config user.name 曾俊菁  
$ git config user.email 1606681131@qq.com  
$ touch phase4.html phase4.css phase4.js
```

二、编写好 phase4.html、phases4.css 和 phase4.js 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add phase4.html phase4.css phase4.js  
$ git commit -m 项目第四版：个性化 UI 设计中对鼠标交互的设计开发
```

三、成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m项目第四版：个性化UI设计中对鼠标交互的设计开发  
[master b52f232] 项目第四版：个性化UI设计中对鼠标交互的设计开发  
3 files changed, 151 insertions(+), 35 deletions(-)
```

四、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
```

五、gitbash 反馈代码的仓库日志如下所示：

```
$ git log  
commit b52f23273075abe2cbf226c96544c72927ad6686 (HEAD -> master)  
Author: 曾俊菁 <1606681131@qq.com>  
Date: Thu Jun 6 11:07:13 2024 +0800
```

项目第四版：个性化UI设计中对鼠标交互的设计开发

7.对触屏和鼠标的通用交互操作的设计开发

7.1 分析与设计

在本项目的第五次迭代中，我致力于开发一套代码逻辑，用以实现对触屏和鼠标操作的通用交互支持。我采用了统一的事件处理方法，确保了无论是触屏还是鼠标操作，用户的操作都能得到准确的捕捉和响应。这种设计不仅提高了代码的可维护性，也使得用户界面在不同设备上都能保持一致性和直观性。

在实现过程中，我特别注重了界面的适应性，通过动态调整布局和行为，确保了用户在各种设备上都能获得优化的交互体验。这一设计策略的实施，不仅提升了系统的用户友好度，也为未来在多平台和多设备环境下的交互设计打下了坚实的基础。如图 7-1 用例图所示：

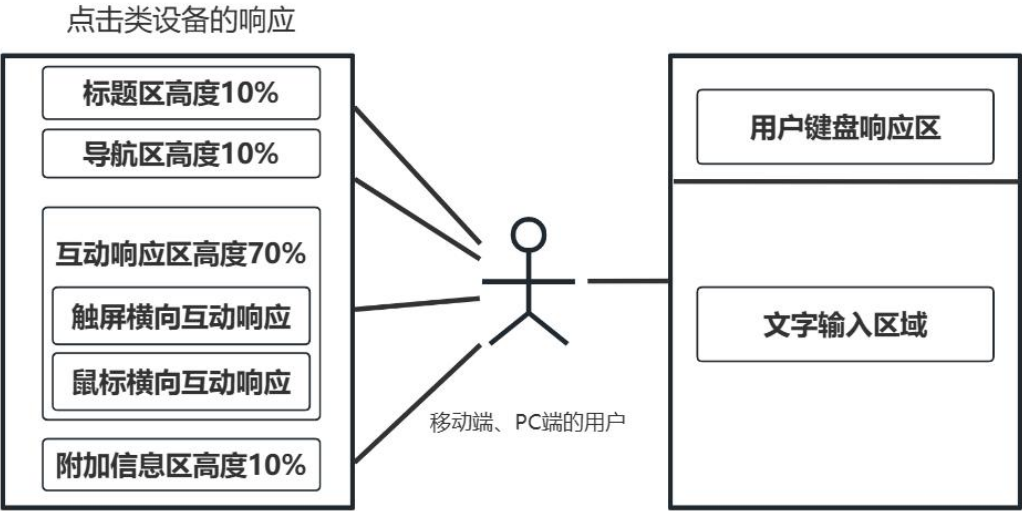


图 7-1 触屏和鼠标的通用交互设计用例图

此次项目的 DOM 树如图 7-2 所示：

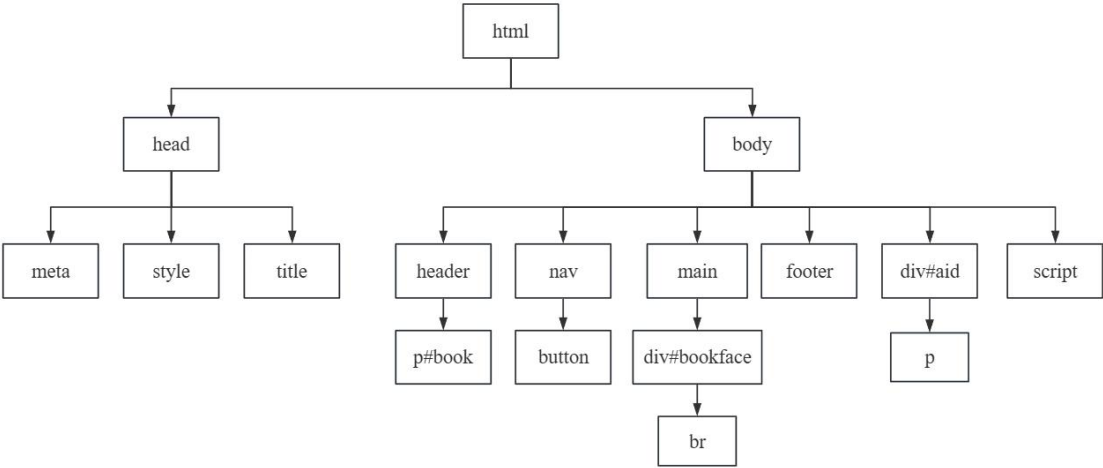


图 7-2 触屏和鼠标的通用交互设计 DOM 树

7.2 项目的实现和编程

这段代码的 `handlePointerEvent` 函数是一个事件处理器，它能够同时处理鼠标和触屏的交互。当检测到按下动作时，函数记录初始位置并显示坐标；在移动过程中，它计算并显示移动距离，同时动态更新元素位置；事件结束时，判断移动是否超过一定阈值来决定操作是否有效，并根据结果更新元素文本和位置。

项目的核心 JS 代码如代码块 7-1 所示：

```
function handlePointerEvent(start, move, end) {
  if (start) {
    Pointer.isDown = true;
    Pointer.x = (ev.touches && ev.touches.length ? ev.touches[0].pageX :
ev.pageX);
    bookface.textContent = `触屏/鼠标按下，坐标：${Pointer.x},${ev.touches ?
ev.touches[0].pageY : ev.pageY}`;
  }
  if (move && Pointer.isDown) {
    Pointer.deltaX = (ev.touches && ev.touches.length ? ev.touches[0].pageX :
ev.pageX) - Pointer.x;
    bookface.style.left = `${Pointer.deltaX}px`;
    bookface.textContent = `正在${ev.type === "touchmove" ? "滑动触屏" : "拖
动鼠标"}，距离：${Pointer.deltaX}px。`;
  }
  if (end) {
    Pointer.isDown = false;
    var isValidMove = Math.abs(Pointer.deltaX) > 100;
    bookface.textContent = `触屏/鼠标事件结束！${isValidMove ? "有效操作" :
"无效操作"}`;
    if (!isValidMove) bookface.style.left = '7%';
  }
}
```

代码块 7-1

7.3 项目的运行和调试

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 Edge 浏览器打开项目的结果，如下图 7-3 所示。



图 7-3 PC 端运行效果图

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 7-4 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 7-4 移动端二维码

7.4 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。

一、进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /  
$ mkdir webUI  
$ cd webUI  
$ git config user.name 曾俊菁  
$ git config user.email 1606681131@qq.com  
$ touch phase5.html phase5.css phase5.js
```

二、编写好 phase5.html、phase5.css 和 phase5.js 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add phase5.html phase5.css phase5.js  
$ git commit -m 项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的 UI
```

成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m项目第五版：实现对触屏和鼠标操作的通用交互支持  
[master 34bda9b] 项目第五版：实现对触屏和鼠标操作的通用交互支持  
3 files changed, 117 insertions(+), 2 deletions(-)
```

三、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
```

四、gitbash 反馈代码的仓库日志如下所示：

```
$ git log  
commit 34bda9b2b69dab101a27de7094c5d311e4435b93 (HEAD -> master)  
Author: 曾俊菁 <1606681131@qq.com>  
Date: Thu Jun 6 13:15:49 2024 +0800  
  
项目第五版：实现对触屏和鼠标操作的通用交互支持
```


8. UI 的个性化键盘交互控制的设计开发

8.1 分析与设计

在项目的第六次迭代中，我为项目 UI 增添了个性化的键盘交互控制功能。使用户能够通过键盘与界面进行更丰富的交互。在这个过程中，我深入探索了键盘事件“keydown”和“keyup”的使用，这两个事件是键盘交互中的基础，为实现复杂的用户界面键盘功能提供了底层支持。

通过对这些底层事件的监听和处理，项目不仅能够捕捉用户的键盘输入，还能实时显示按键动作和对应的字符编码。这样的设计使得用户在进行键盘操作时，能够获得即时的反馈，增强了用户对键盘输入的感知。

项目的用例图如图 8-1 所示：

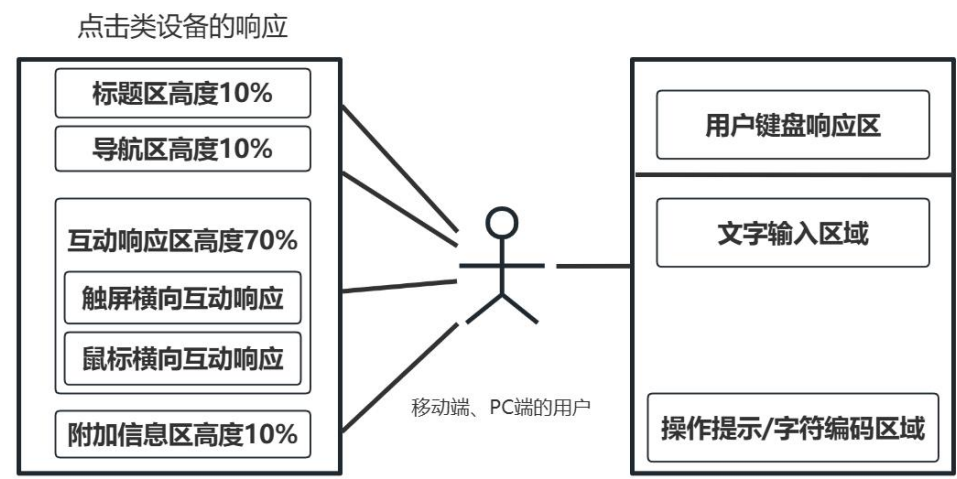


图 8-1 UI 个性化键盘交互设计用例图

项目的 DOM 树如图 8-2 所示：

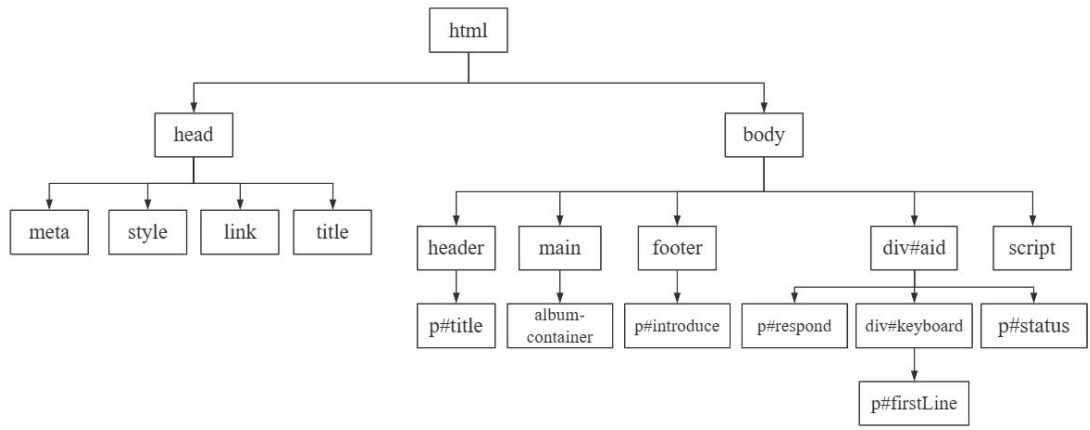


图 8-2 UI 的个性化键盘交互控制设计 DOM 树

8.2 项目的实现和编程

项目的 JS 代码实现了一个基于鼠标滚轮事件的翻页功能。它定义了一个名为`currentIndex`的变量来跟踪当前显示的相册索引，以及一个`albumContainer`变量来引用页面上的相册容器元素。`showPoster`函数用于显示指定索引的海报，更新海报的名称和描述。初始时，它显示第一张海报。通过监听`wheel`事件，当用户向上滚动鼠标滚轮时，页面会切换到前一张海报；向下滚动时，则切换到下一张海报。翻页逻辑确保了在相册的第一张和最后一张海报之间循环切换。

项目的核心 JS 代码如代码块 8-1 所示：

```
/*实现翻页功能*/
var currentIndex = 0;
var albumContainer = $("album-container");
function showPoster(index) {
    var currentPoster = album[index];
    albumContainer.innerHTML = albumHTML;
    $("title").textContent = currentPoster.name;
    $("introduce").textContent = currentPoster.description;
}
showPoster(currentIndex);
// 监听鼠标滚轮滑动事件
document.addEventListener('wheel', function (event) {
    // 鼠标滚轮向上滑动，显示前一张
    if (event.deltaY < 0) {
        currentIndex = (currentIndex === 0) ? album.length - 1 : currentIndex - 1;
    }
    // 鼠标滚轮向下滑动，显示后一张
    else {
        currentIndex = (currentIndex === album.length - 1) ? 0 : currentIndex + 1;
    }
    showPoster(currentIndex);
});
```

代码块 8-1

8.3 项目的运行和调试

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 Edge 浏览器打开项目的结果，如下图 8-3 所示。

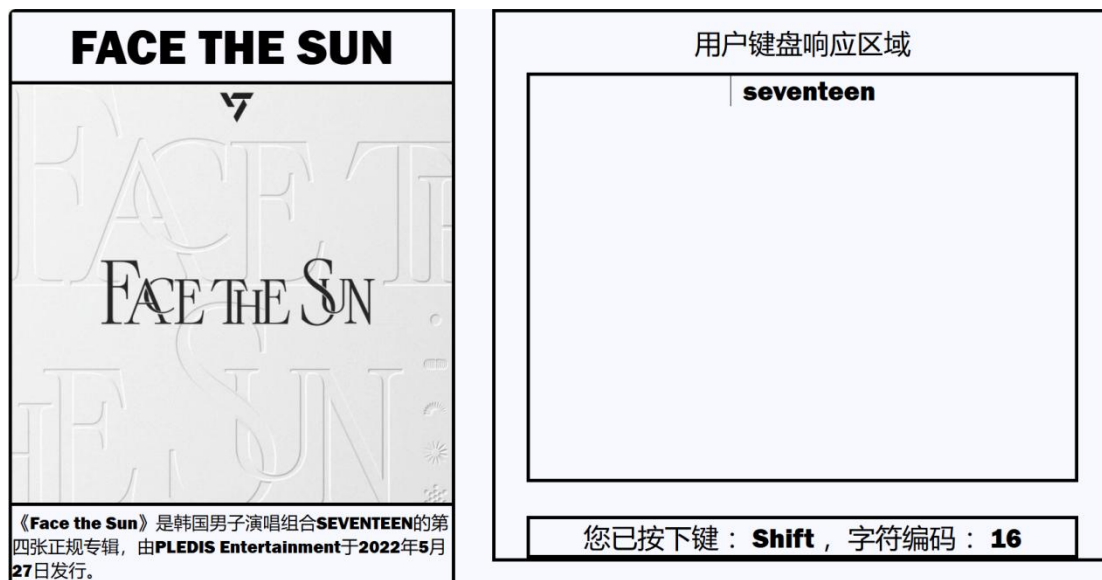


图 8-3 PC 端运行效果图

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 8-4 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 8-4 移动端二维码

8.4 项目的代码提交和版本管理

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();
    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){
        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 ;
} //function printLetter(k)
});
```

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。

一、进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /  
$ mkdir webUI  
$ cd webUI  
$ git config user.name 曾俊菁  
$ git config user.email 1606681131@qq.com  
$ touch phase6.html phase6.css phase6.js
```

二、编写好 phase6.html、phase6.css 和 phase6.js 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add phase6.html phase6.css phase6.js  
$ git commit -m 项目第六版：为项目 UI 增添了个性化的键盘交互控制功能
```

三、成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m项目第六版：为项目UI增添了个性化的键盘交互控制功能  
[master 9c3b003] 项目第六版：为项目UI增添了个性化的键盘交互控制功能  
1 file changed, 147 insertions(+)
```

四、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
```

五、gitbash 反馈代码的仓库日志如下所示：

```
$ git log  
commit 9c3b0033c899902d4b8bbca31504e1d6397283b8 (HEAD -> master)  
Author: 曾俊菁 <1606681131@qq.com>  
Date: Thu Jun 6 13:45:49 2024 +0800  
  
项目第六版：为项目UI增添了个性化的键盘交互控制功能
```

9.用 gitBash 工具管理项目的代码仓库和 http 服务器

9.1 经典 Bash 工具介绍

Bash(Bourne Again Shell)是一个为 GNU 项目编写的 Unix shell 和命令语言。它是 Bourne Shell (sh) 的自由软件替代品,但它提供了比 sh 更多的功能。以下是一些在 Bash 中常用的工具和概念:

一、文件操作命令:

```
$ cd /  
ls: 列出目录内容。  
cat: 查看文件或连接文件。  
cp: 复制文件或目录。  
mv: 移动或重命名文件或目录。  
rm: 删除文件或目录。  
touch: 创建新文件或更新现有文件的时间戳。
```

二、文本处理命令:

```
$ cd /  
echo: 输出一行文本。  
grep: 搜索文本并打印匹配行。
```

三、目录操作命令:

```
$ cd /  
cd: 更改当前目录。  
pwd: 打印当前工作目录。  
mkdir: 创建新目录。  
rmdir: 删除空目录。
```

9.2 通过 gitHub 平台实现本项目的全球域名


通过 GitHub 平台实现本项目的全球域名,主要是利用 GitHub Pages 服务,它允许用户将其 GitHub 仓库作为静态网站托管,并可通过自定义域名提供访问。首先,用户需要在 GitHub 上创建一个新的仓库或者使用现有的项目仓库。然后,在仓库的设置中找到 GitHub Pages 选项,并配置一个分支作为静态网站的源。接着,用户可以将自己的域名与 GitHub 仓库关联,通过设置 DNS 记录中的 CNAME 或 A 记录指向 GitHub Pages 提供的默认域名或 IP 地址来实现域名的自定义。一旦设置完成,用户就可以通过自己的域名来访问托管在 GitHub 上的项目,实现全球范围内的访问。

9.3 创建一个空的远程代码仓库

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *	Repository name *
 1606681131	/ ZengJJ.github.io

✔ ZengJJ.github.io is available.

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

Create repository

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

一、进入本地 webUI 项目的文件夹后，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/1606681131/ZengJJ.github.io.git
$ git push -u origin main
```

二、成功提交代码后，gitbash 的反馈如下所示：

```
$ git push -u origin main
warning: redirecting to https://github.com/1606681131/ZengJJ.github.io.git/
Enumerating objects: 65, done.
Counting objects: 100% (65/65), done.
Delta compression using up to 12 threads
Compressing objects: 100% (63/63), done.
Writing objects: 100% (65/65), 17.91 KiB | 2.56 MiB/s, done.
Total 65 (delta 37), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (37/37), done.
To http://github.com/1606681131/zengjj.github.io.git
* [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

从此，我们无论在本地做了多少次代码修改，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。

参考文献

- [1] W3C. W3C's history. W3C Community. [EB/OL]. <https://www.w3.org/about/>.
<https://www.w3.org/about/history/>. 2024.12.20
- [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