

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：黄丹丹

学号：20213644

指导教师：李健宏

2024 年 6 月 18 日

# 目录

第 1 章 前言 .....	3
1.1 毕设的目的和意义 .....	3
1.2 毕设的路线规划 .....	3
1.3 毕设的研究方法 .....	4
第 2 章 Web 平台和客户端技术概述 .....	5
第 3 章 软件开发的过程管理——增量式开发模式（软件工程视角的项目开发和管理） .....	6
3.1 瀑布模型 .....	6
3.2 增量模型 .....	7
第 4 章 内容设计概要 .....	8
4.1 分析和设计 .....	8
4.2 项目的实现和编程 .....	9
4.3 项目的运行和测试 .....	9
4.4 项目代码提交和版本管理 .....	10
第 5 章 移动互联时代的响应式设计和窄屏代码实现 .....	11
5.1 设计和分析 .....	12
5.2 项目的实现和编程 .....	13
5.3 项目的运行和测试 .....	14
5.4 项目代码提交和版本管理 .....	15
第 6 章 应用响应式设计技术开发可适配窄屏和宽屏的 UI .....	16
6.1 分析和设计 .....	17
6.2 项目的实现和编程 .....	17
6.3 项目的运行和测试 .....	18
6.4 项目代码的提交和版本管理 .....	19
第 7 章 个性化 UI 设计中鼠标模型 .....	20
7.1 分析和设计 .....	20
7.2 项目的实现和编程 .....	20
7.3 项目的运行和测试 .....	22
7.4 项目代码的提交和版本管理 .....	24
第 8 章 通用的 UI 设计，用一套代码同时为触屏和鼠标建模 .....	24
8.1 分析和设计 .....	24
8.2 项目的实现和编程 .....	25
8.3 项目的运行和测试 .....	27
8.4 项目代码的提交和版本管理 .....	28
第 9 章 UI 的个性化键盘控制——应用 keydown 和 keyup 键盘底层事件 .....	28
9.1 分析和设计 .....	28
9.2 项目的实现和编程 .....	29
9.3 项目的运行和测试 .....	30
9.4 项目代码的提交和版本管理 .....	31
第 10 章 用 git 工具展开代码的版本管理和发布 .....	31
10.1 跨世纪的经典工具 Git bash .....	31

10.2 Git 分层操作 .....	32
10.3 Git 工作 .....	33
10.3.1 远程工作 .....	33
10.3.2 创建 <code>github</code> 远程仓库 .....	34
10.3.3 用 Git Bash 建立本地仓库 .....	34
10.3.4 项目提交 .....	36

# WebAppUI 设计开发论文

**摘要：**近十年来，以 html5 为核心的 web 标准软件开发技术以其跨平台、开源的优势广泛地运用在各领域的应用软件开发中。本次课程设计任务选择 html5 的 web 客户端技术为技术路线，展开对程序设计和软件开发的研究和实践。通过广泛查阅相关技术文献、开发者论坛和书籍，我设计开发了一个个性化的用户界面（UI）的应用程序。在开发中综合应用了 html 进行内容建模、css 进行 UI 的外观设计、以及 javascript 编程实现 UI 的交互功能。此外，本项目还采用了响应式设计，可以智能地适应移动互联网时代用户屏幕多样化的需要；另外运用了面向对象的程序设计思想，比如用代码构建了一个通用的 pointer 模型，该模型仅用一套代码就实现了对鼠标和触屏的控制，这也是本项目的亮点。从工程管理的角度看，本项目采用的增量式开发模式，以逐步求精的方式展开了六次代码的增量式重构，比较愉快地实现项目的设计开发和测试。从代码的开源和分享的角度看，本项目采用了 git 工具进行版本管理，对开发过程中重构代码正式做了六次提交，最后利用 gitbash 工具把本项目的代码仓库上传到 github 上，再利用 github 提供的 http 服务器，本项目实现了 UI 应用在全球互联网的部署，我们可以通过地址和二维码便捷地跨平台高效访问这个程序。

**关键词：**软件开发；用户界面；响应式；web；github

## 第 1 章 前言

### 1.1 毕设的目的和意义

毕业设计是高等教育中的一个重要环节。毕设项目通常需要我们从零开始规划和实施，对我们计算机学子而言，毕设要求我们学生综合运用本科阶段学习的计算机科学技术邻域知识，如编程、算法、系统设计等，尤其是程序设计和软件工程领域学习的方法、训练的代码能力，架构自己感兴趣的技术路线，结合自己探求的问题形成软件需求然后有条理第系统落实分析问题、建立模型、软件设计、系统实施、测试调试的等传统软件工程的全部的流程。完成毕业设计，总结开发文档撰写论文，践行二者的有机结合，从而在实践和理论二个维度训练我们专业的计算思维和工程思维。

毕设也是我们对所学过的基础理论和专业知识进行一次全面、系统地回顾和总结，我们通过计算机毕设论文的写作，培养我们综合运用计算机专业知识去分析并解决实际问题的能力，学有所用，不仅实践操作、动笔能力得到很好的锻炼，还极大地增强了我们今后走向社会拼搏、奋斗的勇气和自信。

### 1.2 毕设的路线规划

本项目论文只是毕设的雏形，内容并不完善，项目花费时间也较短，从 5

月开始进行初步的内容架构，到 6 月完成论文写作。毕设的路线规划是一个系统的过程，涉及项目开始到结束的各个阶段，有效的规划可以确保毕设项目的顺利进行、项目完成的质量和进度，合理安排时间，提高工作效率。

我的毕设路线规划分三个阶段完成。第一阶段选择 html5 的 web 客户端技术为技术路线，学习 HTML5 的语义标签、表单、基础 API，CSS 的布局技术、响应式设计，JavaScript 的基础语法、DOM 操作、事件处理和异步编程，理解三者在 Web 开发中的相互作用和关系；学习版本控制系统 Git Bash 的使用，以便在后期开发过程中进行代码管理和协作。。以导师的案例项目为参考，主要是理解各个技术之间的关系，在项目中的作用和分工，更重要的是在项目实施中提升自己写高质量的代码能力。

第二个阶段开始并完成毕设项目。第二阶段是毕设项目的核心开发阶段，按照软件工程的标准规范开发项目：1.分析设计自己的项目内容，明确项目要解决的问题；2.根据项目的问题，设计一套合适的技术解决方案；3.按解决方案设计流程和编写相关代码，实现技术部署；4.调试代码、测试软件、性能调优；5.使用 Git Bash 定期提交代码，保持代码的版本控制。

第三阶段完成毕设论文。在论文中完成项目的分析、设计、测试结果等文字描述，并插入和项目有关的用例图、DOM 树、运行结果图。

### 1.3 毕设的研究方法

在完成毕业设计的过程中，毕设采用了什么研究方法很关键，一个好的且合适研究方法可以加快推进我们毕设完成进度。毕业设计的研究方法取决于我们项目的研究领域和具体的研究问题，本项目以 html5 的 web 客户端技术为技术路线，展开对程序设计和软件开发的研究和实践，采用的是文献法和模型研究法。

1.文献法：此方法是最常见的毕业设计方法，通过阅读和分析现有文献可以了解我们的研究领域和发展趋势。文献法本质上就是利用前人的文字深入学习总结和研究本领域的知识和技术。对于我们 web 应用方向的学子而言，在自己代码能力成长的任何阶段，我们都无法离开文献法，我们可以合理利用书籍、期刊、调查报告、互联网等公开文献及调研结果来获取有用信息完成毕设。

2.模型研究法：在编写代码过程中，模型研究法是一种重要的方法，它通过

构建模型来模拟和研究现实世界中的问题。承载我们 web 应用的台式机、笔记本、手机、平板，传递在线信息要用到的互联网、服务器，沟通硬件和我们的代码之间的操作系统、浏览器、代码编辑器、编译器，这些软硬件对象，对我们而言，都值得从写代码的角度去研究，我们笼统地称它们为对象，这些对象最终会在我们大脑中就会被理解为抽象的模型，我们再通过分析把这些模型程序化、数据化，最后写出代码来，这种行为本质上就是先在思维上“建模”。

## 第 2 章 Web 平台和客户端技术概述

让我们先简单介绍一下 Web，Web 开发是一个内容广阔的技术，Web (World Wide Web)即全球广域网，也就是万维网的缩写。Web 是一种基于超文本和 HTTP 的、全球性的、动态交互的、跨平台的分布式图形信息系统，是建立在 Internet 上的一种网络服务，为浏览者在 Internet 上查找和浏览信息提供了图形化的、易于访问的直观界面，其中的文档及超级链接将 Internet 上的信息节点组织成一个互为关联的网状结构。我们在手机或电脑浏览器上输入网址(学名 URL(Uniform Resource Locator 即统一资源定位符))，就能进入图形化的交互页面。大多数人说“Web”而不是“World Wide Web”，我们将遵循这一惯例。网络是文档的集合，称为网页，由世界各地的计算机用户共享。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。这里的“内容”是指文本、图片和用户输入机制(如文本框和按钮)。

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。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，这也是我的毕设项目应用的技术路线。

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）；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。

## 第 3 章 软件开发的过程管理——增量式开发模式（软件工程视角的项目开发和管理）

本项目作为一个本科计算机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。

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 (The waterfall model) 和增量模型(The incremental model)。

### 3.1 瀑布模型

瀑布模型是一个经典的软件生命周期模型，也叫预测型生命周期、完全计划驱动型生命周期。在这个模型里，在项目生命周期的尽早时间，要确定项目范围及交付此范围所需的时间和成本。

瀑布模型必须等前一阶段的工作完成之后，才能开始后一阶段的工作，传统的瀑布模型如图 3-1 所示。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。

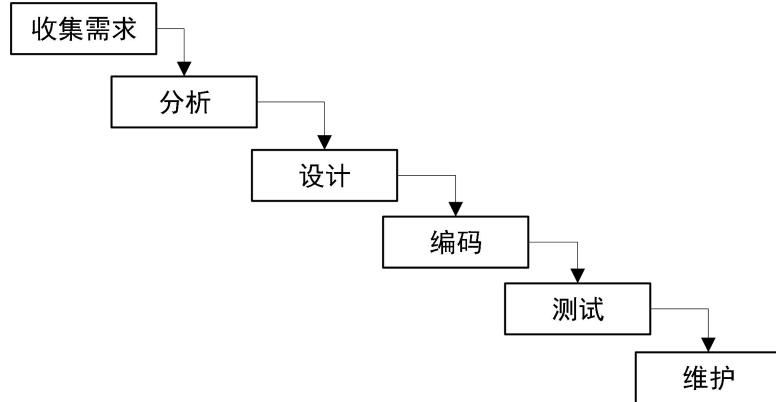


图 3-1 传统瀑布模型

## 3.2 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一个简化版本，这个版本表示整个系统，但不包括系统内部的详细设计。图 3-2 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。若在编程和测试中发现了设计甚至是分析中的问题，开发人员就能够及时反馈以及更改设计。

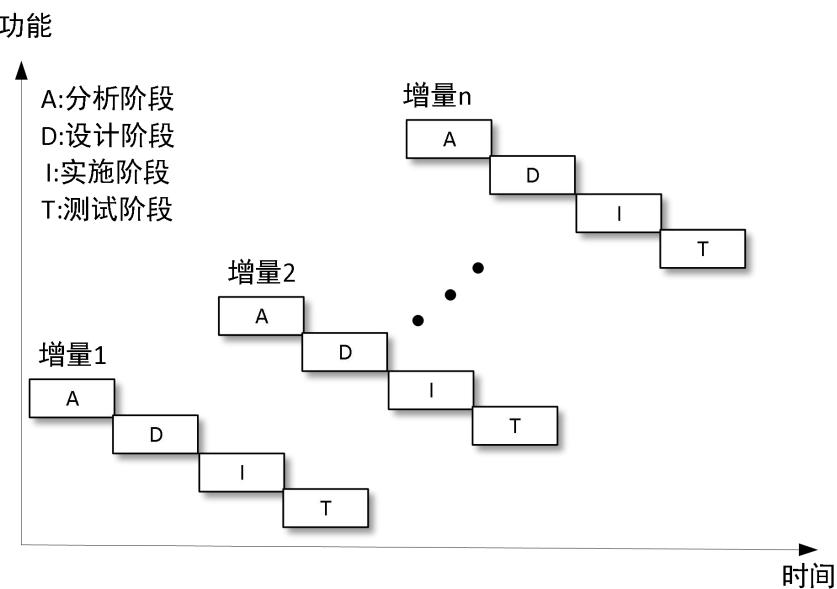


图 3-2 增量模型

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。本项目中我们一共做了六次项目的开发迭代。

## 第 4 章 内容设计概要

### 4.1 分析和设计

本项目使用人们习惯的简洁的“三段论”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，然后展现的主要区域则是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的主题，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。内容分析如图 4-1，设计如图 4-2。

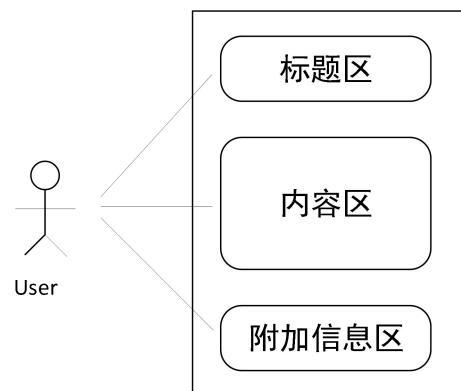


图 4-1 三段论用例图

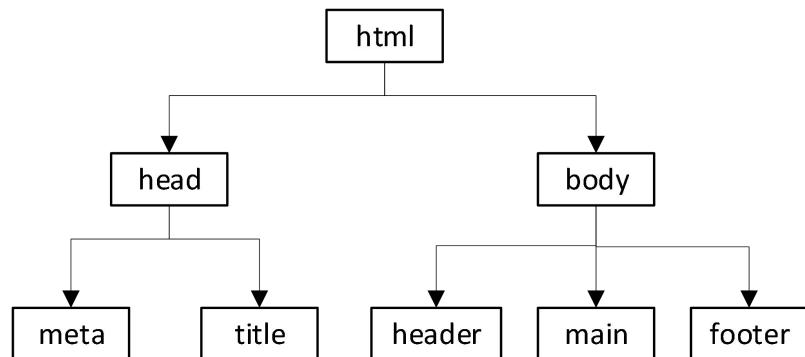


图 4-2 三段论 DOM 树

## 4.2 项目的实现和编程

### 1.HTML 代码

第一阶段内容设计代码比较简单，html 代码如代码块 4-3 所示。

```
<header>
    《书籍推荐》
</header>
<main>
    读书是在别人思想的帮助下，建立起自己的思想。
</main>
<footer>
    CopyRight from 黄丹丹 江西科技师范大学 2024-2025
</footer>
```

代码块 4-3 第一次 html 代码

### 2.CSS 代码

内容设计概要部分 CSS 代码如代码块 4-4 所示，定义了 header、main、footer 的边框和高度信息。

```
* {
    margin: 10px;
    text-align: center;
}
header {
    border: 2px solid blue;
    height: 200px;
}
main {
    border: 2px solid blue;
    height: 400px;
}
footer {
    border: 2px solid blue;
    height: 100px;
}
```

代码块 4-4 第一次 CSS 设计代码

## 4.3 项目的运行和测试

项目的运行和测试至少要通过两类终端，本文此处仅给出 PC 端用 Chrome 浏览器打开项目的结果，如下图 4-5 所示。本项目的阶段性文件已经上传 github

网站（本章代码 URL 地址：<https://hdd1654231824.github.io/exp/1.1.html>），我们在 PC 端可以复制上述地址，在现代浏览器（含微信）中运行。移动端用户可以通过扫描图 4-6 的二维码，在移动端的现代浏览器（含微信）中运行测试本项目的第一阶段效果。

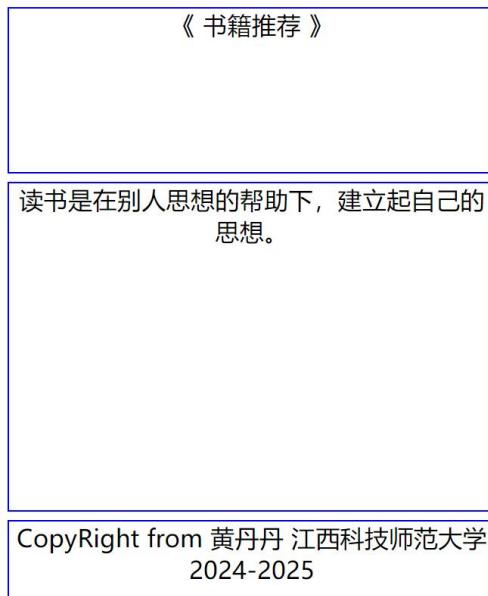


图 4-5 第一次运行效果

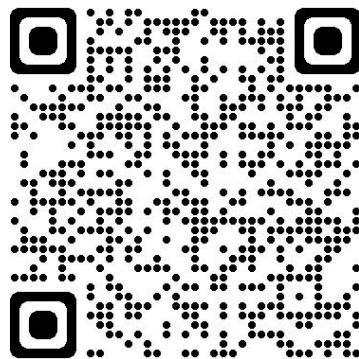


图 4-6 第一次二维码

#### 4.4 项目代码提交和版本管理

本阶段完成的代码存放在文件中，通过 Git Bash 根据作正式的代码提交。本项目的文件通过 Git Bash 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。通过 Git Bash 进入本地项目仓库后，先建立项目需要的文件夹，如图 4-7 所示。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ touch index.html myCSS.css myJS.js

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ mkdir images txt exp
```

图 4-7 建立基本文件

在代码编辑器中完成代码编写整合到 index.html，测试运行成功后，打开 gitbash，先使用 git add 命令将修改内容从本地仓库提交到暂存区，再使用 git commit 命令将内容从暂存区提交到仓库。成功提交代码后，gitbash 的反馈如图 4-8 所示：

```

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git add .

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:  images/myface1.jpg
    new file:  images/myface2.jpg
    new file:  index.html
    new file:  myCSS.css
    new file:  myJS.js

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git commit -m "建立应用的文件夹结构并做第一次代码提交"
[master (root-commit) 786cce9] 建立应用的文件夹结构并做第一次代码提交
  5 files changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 images/myface1.jpg
  create mode 100644 images/myface2.jpg
  create mode 100644 index.html
  create mode 100644 myCSS.css
  create mode 100644 myJS.js

```

图 4-8 git 第一次管理代码

成功提交后，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 git log 日志命令查看，gitbash 反馈代码的仓库日志如下图 4-9 所示：

```

$ git log
commit 85bf4cf139d355621d2a302d9c656116bb50ab30 (HEAD -> master)
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date:   Tue Jun 4 10:44:01 2024 +0800

    把本阶段的代码进行了文件合并，形成了文件1.1.html用于项目的阶段性效果展示。

commit 0380a1d5a45d34724fa9b723c91d6a4ce781db07
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date:   Tue Jun 4 10:33:07 2024 +0800

    “三段论”式的内容设计概要

commit 786cce991989e4d9407a99b74071b19cfaf4dd3d
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date:   Tue Jun 4 10:04:51 2024 +0800

    建立应用的文件夹结构并做第一次代码提交

```

图 4-9 第一次日志记录

## 第 5 章 移动互联时代的响应式设计和窄屏代码实现

早年设计 Web 时，页面是以适配特定的屏幕大小为考量创建的。如果用户正在使用比设计者考虑到的更小或者更大的屏幕，那么结果从多余的滚动条，到过长的行和没有被合理利用的空间，不一而足。随着人们使用的屏幕尺寸的种类越来越多，出现了响应式网页设计的概念（responsive web design, RWD），RWD 指的是允许 Web 页面适应不同屏幕宽度因素等，进行布局和外观的调整的一系列实践。

响应式 Web 设计不是单独的技术，它是描述 Web 设计的一种方式、或者是一组最佳实践的一个词，它是用来建立可以响应查看内容的设备的样式的一个词。Web 的响应式工作已经成为了默认做法，现代的 CSS 布局方式基本上就是响应式的，而且我们在 Web 平台上内置了新的东西，使得设计响应式站点变得容易。

响应式页面设计的基本原理是通过媒体查询不同的设备屏幕尺寸，根据屏幕尺寸，对页面进行布局和样式的设置，使页面在不同设备上看起来像是专门定制的页面，为了处理移动端，页面头部必须使用 meta 标签声明视口

## 5.1 设计和分析

随着移动互联网的发展，人们越来越多地使用手机和平板电脑进行网页浏览和信息获取。因此，针对移动设备的网页优化已成为前端开发中的重要环节。响应式网页设计作为一种灵活的解决方案，在不同设备上提供一致性的用户体验，因此在移动设备上的优化策略尤为重要，同时这种策略对 pc 端和移动端都能达到一种合理的页面展示，无论是市面上各种型号或者品牌的手机、平板以及电脑，都能使其在页面中呈现较为理想的布局状态。本章将整体内容分为四个区域：标题区、导航区、附加信息区，不同的设备各区域占据整体的内容高度百分比都是一致的，窄屏手机的 WebUI 设计如图 5-1 所示，分析如图 5-2 所示。

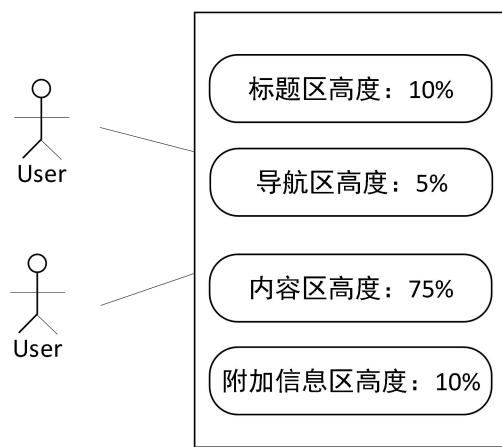


图 5-1 响应三星和苹果手机的 WebUI 用例图

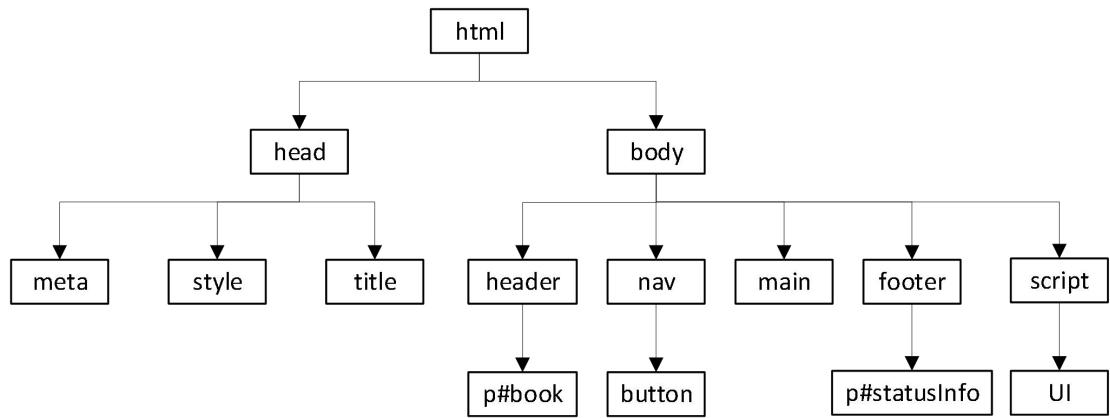


图 5-2 响应窄屏 DOM 树

## 5.2 项目的实现和编程

1.HTML 代码: 本阶段 HTML 部分代码如代码块 5-3 所示, body 分为 header 标题区、nav 导航区、main 内容区、footer 附加信息区四部分。

```

<header>
    <p id="book">书籍推荐</p>
</header>
<nav>
    <button>导航 1</button>
    <button>导航 2</button>
    <button>导航 3</button>
</nav>
<main>
</main>
<footer>
    <p id="statusInfo">
        CopyRight from 黄丹丹 江西科技师范大学 20213644
    </p>
</footer>
  
```

代码块 5-3 第二次 html 代码

2.CSS 代码: 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, 部分代码如代码块 5-4 所示。

```

footer {
    border: 2px solid #7ea8df;
    height: 10%;
    background-color: #7ea8df;
    color: white;
}
  
```

```
p#book {  
    font-size: 1.5em;  
    letter-spacing: 0.2em;  
}  
button {  
    font-size: 1.2em;  
    margin-right: 1em;  
}
```

代码块 5-4 第二次 CSS 设计代码

3.JS 代码：用汉语言来描述我们是如何实现的：与上一阶段比较，本阶段首次使用了 JavaScript，首先创建了一个 UI 对象，然后把系统的宽度 window.innerWidth 和高度 window.innerHeight 记录在 UI 对象 deviceWidth 和 deviceHeight 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，代码块如图 5-5 所示：

```
var UI = {};  
UI.deviceWidth = window.innerWidth;  
UI.deviceHeight = window.innerHeight;  
document.body.style.height = UI.deviceHeight + 'px';  
document.body.style.fontSize = UI.deviceWidth / 75 + 'px';
```

代码块 5-5 全屏代码设置

### 5.3 项目的运行和测试

本阶段测试了项目在不同尺寸设备（iPhone SE 和 Samsung Galaxy S8+）中的响应情况，iPhone SE 响应效果如图 5-6，Samsung Galaxy S8+ 响应效果如图 5-7。本阶段项目 URL 为：<https://hdd1654231824.github.io/exp/1.2.html>，移动端用户可以通过扫描下方图 5-8 二维码查看效果。



图 5-6 iPhoneSE 效果

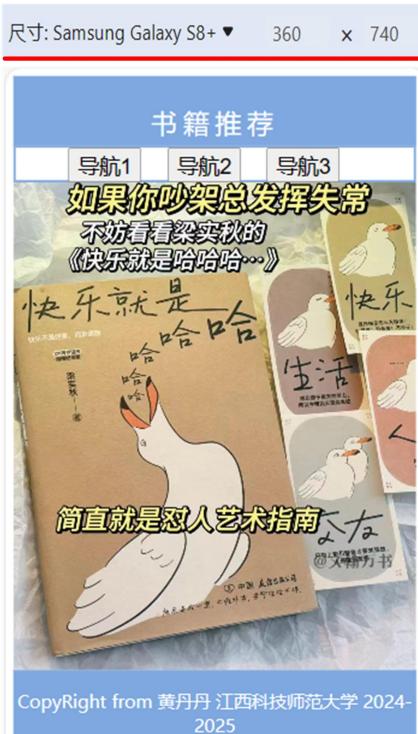


图 5-7 三星手机效果

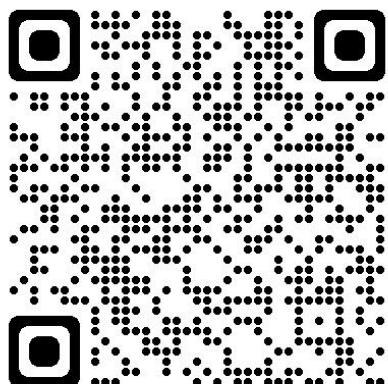


图 5-8 第二次二维码

## 5.4 项目代码提交和版本管理

本阶段在代码编辑器中完成 index.html 代码，测试运行成功后，执行 git add 和 git commit 命令提交，成功提交代码后，gitbash 的反馈如图 5-9 所示。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/study/webUI (master)
$ git add .

pc@LAPTOP-SBOSJNSS MINGW64 /d/study/webUI (master)
$ git commit -m"移动互联网时代的响应式设计（窄屏）"
[master abb51a5] 移动互联网时代的响应式设计（窄屏）
 7 files changed, 205 insertions(+), 66 deletions(-)
  create mode 100644 .vscode/launch.json
  create mode 100644 exp/1.2.html
  create mode 100644 images/sj1.jpg
  rewrite index.html (88%)
  rewrite myCSS.css (76%)
```

图 5-9 git 第二次管理代码

成功提交后，使用 git log 查看历史提交记录如图 5-10 所示。

```
$ git log
commit abb51a5a0603dbfbc89685b1aa71f6a4bdd36626 (HEAD -> master)
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date:   Mon Jun 10 16:10:23 2024 +0800

  移动互联网时代的响应式设计（窄屏）
```

图 5-10 第二次日志记录

## 第 6 章 应用响应式设计技术开发可适配窄屏和宽屏的 UI

随着移动设备的普及和多样化，网页在不同尺寸的屏幕上呈现出来的效果也愈发重要。响应式设计就是一种能够确保网站在各种设备上都能有良好表现的设计理念。在实际开发中，通过媒体查询、流式布局等技术手段，可以实现网页在不同分辨率下的自适应布局和样式调整。这样一来，无论用户使用台式机、笔记本、平板还是手机，都能获得良好的浏览体验。

而在响应式设计的基础上，移动优先原则则强调在开发网页时应首先考虑移动设备的访问体验，然后再逐步扩展到更大屏幕的设备上。这种思路可以帮助开发者更好地专注于核心内容和功能，并通过渐进增强的方式来丰富页面的交互体验。在具体实现上，可以采用弹性布局、图片优化、模块化组件等技术手段来满足移动设备的需求，同时也需要注重性能优化和加载速度，以提升移动端用户的访问体验。总之，响应式设计和移动优先原则已经成为前端开发中不可或缺的重要部分。通过合理运用这些概念和技术，可以为用户提供更加统一、流畅的跨设备访问体验，也能更好地适应日益多元化的移动互联网环境。

## 6.1 分析和设计

该部分设计当设备屏幕宽度大于 600px 时，屏幕右边会增加一个用户键盘响应区，但是当设备屏幕宽度不足 600px 时，UI 界面没有用户键盘响应区，分析如图 6-1，用户键盘响应区 JS 部分设计如图 6-2。

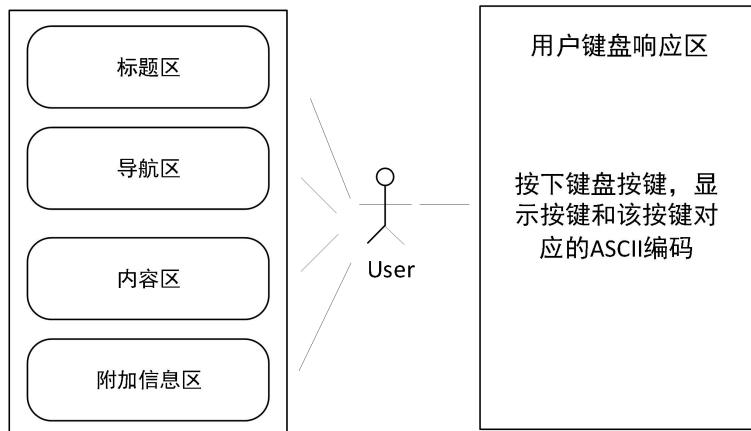


图 6-1 响应式屏幕用例图

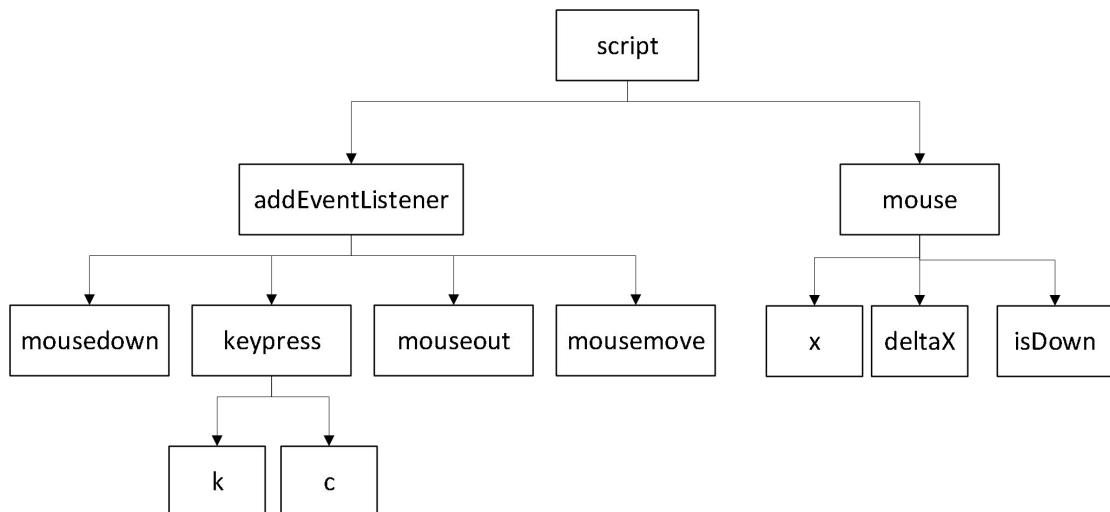


图 6-2 响应式屏幕 DOM 树设计

## 6.2 项目的实现和编程

本阶段最重要的代码部分是 JS 代码。本阶段通过把 body 的高度 (`document.body.style.width`) 设置为屏幕高度 `UI.appWidth`，实现纵向全屏；同时通过 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，代码如代码块 6-3 所示；还增加了 `addEventListener` 事件监听器监听 `keypress` 事

件。当用户在用户键盘响应区按下键盘按键时，会触发 `keypress` 事件，`k` 记录当前按下的键，`c` 记录该按键对应的 ASCII 编码值，最后用户键盘响应区会显示“按键是：`k`；编码是：`c`”信息，代码如代码块 6-4 所示。

```
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 + "px";
document.body.style.height = UI.appHeight - 62 + "px";
if (window.innerWidth < 1000) {
    $("aid").style.display = 'none';
}
$("aid").style.width = window.innerWidth-UI.appWidth - 30 +'px';
$("aid").style.height = UI.appHeight - 62 + 'px';
```

代码块 6-3 响应宽屏和窄屏代码

```
$( "body" ).addEventListener( "keypress" , function( ev ) {
    let k = ev.key;
    let c = ev.keyCode;
    let s1 = "按键是： ";
    let s2 = "编码是： "
    $("keyboard").textContent = s1 + k + ";" + s2 + c;
});
```

代码块 6-4 `keypress` 事件代码

### 6.3 项目的运行和测试

PC 端测试运行，在用户键盘响应区按下 `a` 键，测试结果如图 6-5。本阶段项目 URL 为：（<https://hdd1654231824.github.io/exp/1.3.html>），我们在 PC 端可以复制上述地址在现代浏览器中运行。部分移动设备平台屏幕宽度不足 600px，所以没有用户键盘响应区，可以用移动设备点击上述地址或者通过扫描图 6-6 中的二维码在移动端中查看。

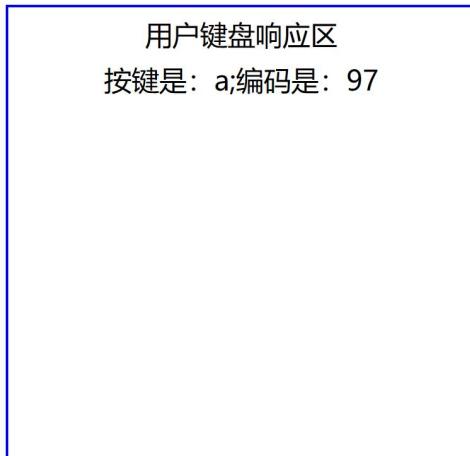


图 6-5 用户键盘响应 keypress 事件

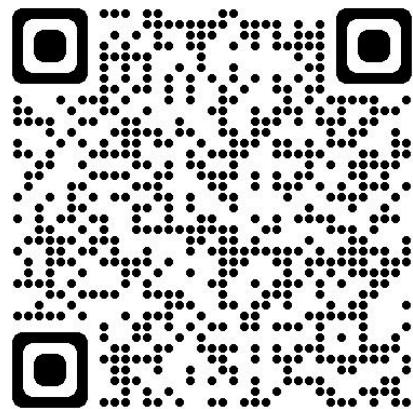


图 6-6 第三次二维码

## 6.4 项目代码的提交和版本管理

本阶段初步完成了宽屏和窄屏的响应式交互设计，在代码编辑器中完成 index.html 代码并测试运行成功后，执行 git add 和 git commit 命令提交，成功提交代码后，gitbash 的反馈如图 6-7 所示。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/study/webUI (master)
$ git add .

pc@LAPTOP-SBOSJNSS MINGW64 /d/Stu/dy/webUI (master)
$ git commit -m"响应式设计（宽屏和窄屏）"
[master d258231] 响应式设计（宽屏和窄屏）
 4 files changed, 350 insertions(+), 137 deletions(-)
  create mode 100644 exp/1.3.html
  rewrite index.html (69%)
  rewrite myCSS.css (80%)
  rewrite myJS.js (93%)
```

图 6-7 git 第三次管理代码

git log 查看本阶段提交日志记录如图 6-8 所示。

```
$ git log
commit d258231eb39ac7fcb25194137a1e9a7a032e56a4 (HEAD -> master)
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date:   Mon Jun 10 22:25:41 2024 +0800

  响应式设计（宽屏和窄屏）
```

图 6-8 第三次日志记录

# 第 7 章 个性化 UI 设计中鼠标模型

## 7.1 分析和设计

在人机交互过程中，鼠标扮演着非常重要的角色，用户通过鼠标能够更方便快捷地操作计算机设备各类 GUI 界面，所以我们在设计个性化用户界面时，初步探究了 UI 设计中 Web API 鼠标事件，特别是关注了鼠标在计算机中的坐标信息。我们希望当在设备固定区域进行鼠标点击事件时，页面会显示响应的鼠标的坐标信息，分析如图 7-1，JS 鼠标模型部分设计如图 7-2。

请注意：此类事件不仅可能来自于“鼠标设备”，还可能来自于对此类操作进行了模拟以实现兼容性的其他设备，例如手机和平板电脑。

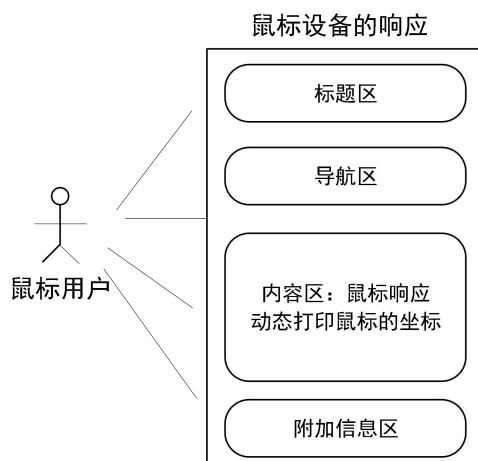


图 7-1 鼠标模型

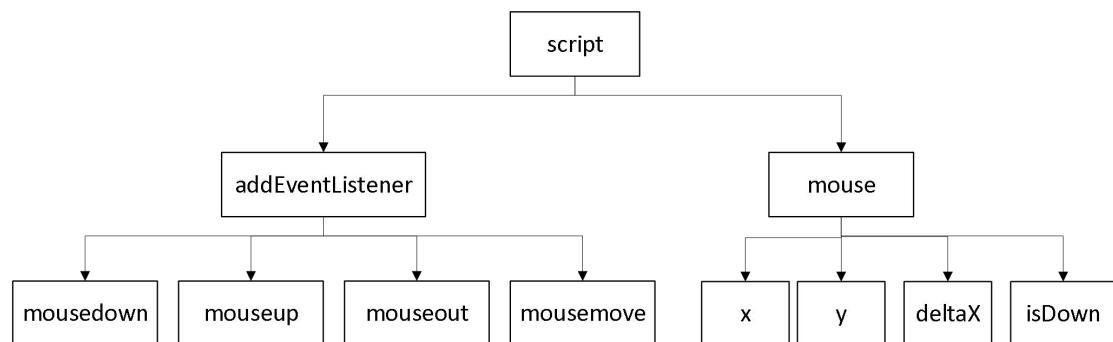


图 7-2 鼠标模型 DOM 树设计

## 7.2 项目的实现和编程

本阶段重点是 JS 代码。在 JavaScript 中，鼠标事件是 Web 开发中最常用的

事件类型，本阶段增加了监听器 addEventListener 和 mousedown、mouseup、mousemove 和 mouseout 事件，有关鼠标事件的部分代码如代码块 7-3-1 至 7-3-4。

1.mousedown 和 mouseup 事件：mousedown 事件在定点设备（如鼠标或触摸板）按钮在元素内按下时，会在该元素上触发，mouseup 事件与 mousedown 事件相对应。即鼠标按下触发 mousedown 事件，松开鼠标触发 mouseup 事件。一直按着鼠标不松开时，会连续触发 mousedown 事件，而 mouseup 事件不能连续触发。

2.mousemove 和 mouseout 事件：mousemove 事件类型是一个实时响应的事件，当鼠标指针的位置发生变化时（至少移动一个像素），就会触发 mousemove 事件，该事件响应的灵敏度主要参考鼠标指针移动速度的快慢以及浏览器跟踪更新的速度；mouseout 事件在定点设备（通常是鼠标）移动至元素或其子元素之外时，会在该元素上触发。当指针从一个元素移入其子元素时，因为子元素遮盖了父元素的可视区域，所以 mouseout 也会被触发。

```
$( "bookface" ).addEventListener( "mousedown" , function( ev ) {  
    mouse.isDown = true;  
    mouse.x = ev.pageX;  
    mouse.y = ev.pageY;  
    console.log("mouseDown at x: " + "(" + mouse.x + "," + mouse.y + ")");  
    $("bookface").textContent="鼠标按下,坐标:"+"("+mouse.x+","+mouse.y+")";  
});
```

代码块 7-3-1 鼠标设计 mousedown 事件

```
$( "bookface" ).addEventListener( "mouseup" , function( ev ) {  
    mouse.isDown = false;  
  
    $("bookface").textContent = "鼠标松开!";  
    if (Math.abs(mouse.deltaX) > 100) {  
        $("bookface").textContent += ", 这是有效拖动！";  
    } else {  
        $("bookface").textContent += " 本次算无效拖动！";  
        $("bookface").style.left = '7%';  
    }  
});
```

代码块 7-3-2 鼠标设计 mouseup 事件

```
$( "bookface" ).addEventListener( "mouseout" , function( ev ) {  
    ev.preventDefault();  
    mouse.isDown = false;  
    $("bookface").textContent = "鼠标松开!";
```

```
        if (Math.abs(mouse.deltaX) > 100) {
            $("bookface").textContent += " 这次是有效拖动！";
        } else {
            $("bookface").textContent += " 本次算无效拖动！";
            $("bookface").style.left = '7%';
        }
    });
});
```

代码块图 7-3-3 鼠标设计 mouseout 事件

```
$("bookface").addEventListener("mousemove", function(ev) {
    ev.preventDefault();
    if (mouse.isDown) {
        console.log("mouse isDown and moving");
        mouse.deltaX = parseInt(ev.pageX - mouse.x);
        $("bookface").textContent = "正在拖动鼠标， 距离：
" + mouse.deltaX + "px 。";
        $('bookface').style.left = mouse.deltaX + 'px';
    }
});
```

代码块 7-3-1 鼠标设计 mousemove 事件

### 7.3 项目的运行和测试

PC 端测试运行，在鼠标响应区任一位置按下鼠标，触发 mousedown 事件，响应区显示“鼠标按下，坐标：(474, 503)”，如图 7-4；松开鼠标，触发 mouseup 事件，因鼠标立马松开，移动横向距离移动小于 100px，所以响应区显示“鼠标松开！本次算无效拖动！”，如图 7-5；沿 X 轴拖动鼠标，触发 mousemove 事件，响应区显示“正在拖动鼠标，距离：82px。”，如图 7-6；松开鼠标，触发 mouseout 事件，因鼠标移动横向距离移动大于 100px，所以响应区显示“鼠标松开！这次是有效拖动！”，如图 7-7。鼠标在鼠标响应区域移动，UI 界面会显示鼠标移动的坐标，也会在控制台触发鼠标移动和鼠标按下事件，如图 7-8 所示。

本项阶段项目 URL 为：<https://hdd1654231824.github.io/exp/1.4.html>，移动设备可以复制上述地址在浏览器中打开或者通过扫描图 7-9 中的二维码在移动端查看。

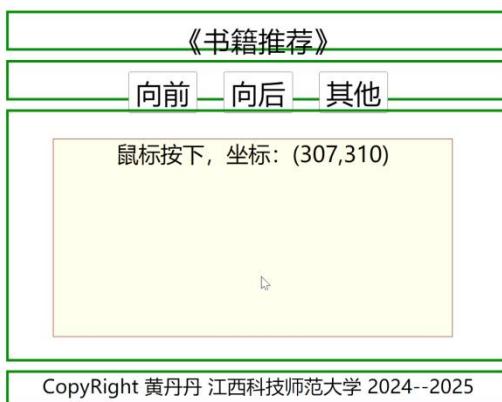


图 7-4 keydown 触发效果



图 7-5 keyup 触发效果



图 7-6 keymove 触发效果



图 7-7 keyout 触发效果

```
mouseDown at x: (275,407)  
mouseDown at x: (377,322)  
mouseDown at x: (257,437)
```

[1.4.html:135](#)

[1.4.html:135](#)

[1.4.html:135](#)

**14** mouse isDown and moving

[1.4.html:166](#)

图 7-8 控制台输出

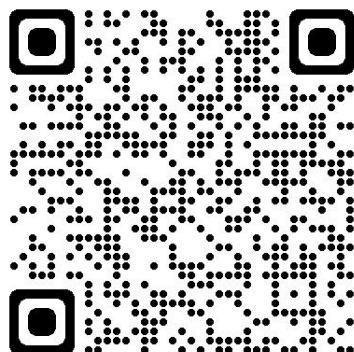


图 7-9 第四次二维码

## 7.4 项目代码的提交和版本管理

本阶段初步完成了 UI 设计中的鼠标模型，成功提交代码后，gitbash 的反馈如图 7-10 所示。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git add .

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git commit -m"UI设计之鼠标模型的分析和控制"
[master 9a423a2] UI设计之鼠标模型的分析和控制
 4 files changed, 295 insertions(+), 55 deletions(-)
  create mode 100644 exp/1.4.html
```

图 7-10 git 第四次管理代码

git log 查看本阶段提交日志记录如图 7-11 所示。

```
commit 9a423a2a5c9c5d71929c773a460cfabdaafb6f8 (HEAD -> master)
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date:   Wed Jun 12 09:48:30 2024 +0800

    UI设计之鼠标模型的分析和控制
```

图 7-11 第四次日志记录

## 第 8 章 通用的 UI 设计，用一套代码同时为触屏和鼠标建模

### 8.1 分析和设计

上一阶段中，我们研究了鼠标模型，并通过监听鼠标的移动等事件实现响应区的移动。遗憾的是，这种方式只适用于 PC 端，当我们使用移动端进行操作时，是无法实现该效果的。本阶段设计了一个通用的 UI，使其既适用于 PC 端鼠标操作又适用于移动端触屏操作，通用 UI 分析用例图如图 8-1，DOM 树 8-2 如图。在此对象范围拖动鼠标/滑动触屏超过 100px，视为有效的 UI 互动！

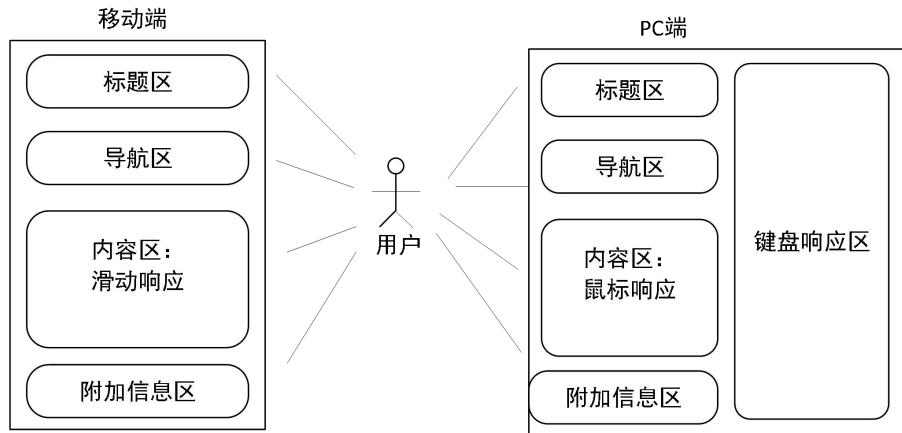


图 8-1 通用 UI 分析用例图

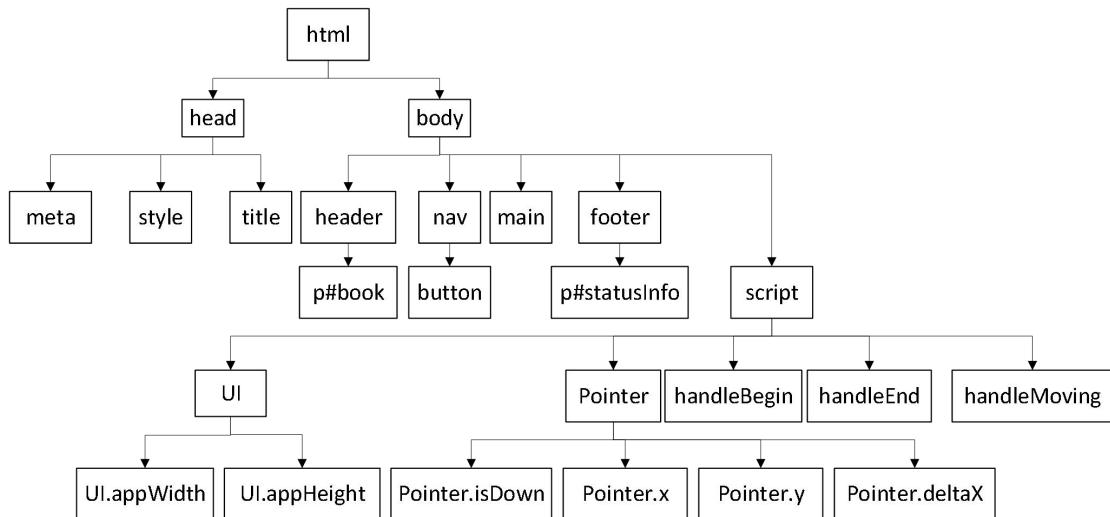


图 8-2 通用 UI 的 DOM 树设计

## 8.2 项目的实现和编程

本阶段重点是 JS 代码，尝试对鼠标和触屏设计一套代码实现 UI 控制。定义了 handleBegin、handleEnd 和 handMoving 三个功能函数。

(1) handleBegin 定义了事件开始时发生的事情，如果是 ev.touches 事件触发，则是触屏操作，否则是鼠标操作。两种操作发生时，都会改变 id 为 bookface 中 textContent 内容，代码如代码块 8-3 所示。

```
//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 + ")";
    };
}

```

代码块 8-3 鼠标和触屏通用 UI 控制 JS 代码

(2) handleEnd 功能会获取触屏滑动/鼠标移动的纵向距离，在此对象范围拖动鼠标/滑动触屏超过 100px，视为有效的 UI 互动，否则为无效互动，代码如代码块 8-4 所示。

```

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%';
        }
        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%';
    }
}

```

代码块 8-4 拖动鼠标/滑动触屏 JS 设计代码

(3) 当 Pointer.isDown 为 true，即触屏/鼠标移动时，才会触发 handMoving 功能。用户触屏滑动/鼠标移动时，控制台和 UI 响应界面都会反馈移动信息和坐标给用户。代码如代码块 8-5。

```
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';
            }
        }
    }
};
```

代码块 8-5 handMoving 功能函数

### 8.3 项目的运行和测试

本阶段项目 URL 为：<https://hdd1654231824.github.io/exp/1.5.html>，PC 端效果和上一阶段效果一样。移动端用户可以扫描下方图 8-6 二维码查看运行效果。

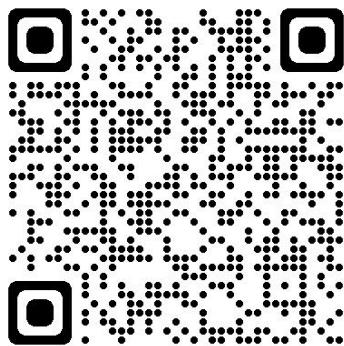


图 8-6 第五次二维码

## 8.4 项目代码的提交和版本管理

本阶段初步完成了 PC 端和移动端通用 UI 的设计，成功提交代码后，gitbash 的反馈如图 8-7 所示。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/study/webUI (master)
$ git add .

pc@LAPTOP-SBOSJNSS MINGW64 /d/Stuuy/webUI (master)
$ git commit -m"通用的UI设计，用一套代码同时为触屏和鼠标建模"
[master 7d5720d] 通用的UI设计，用一套代码同时为触屏和鼠标建模
 2 files changed, 299 insertions(+), 53 deletions(-)
 create mode 100644 exp/1.5.html
```

图 8-7 git 第五次管理代码

git log 查看本阶段提交日志记录如图 8-8 所示。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/Stuuy/webUI (master)
$ git log
commit 7d5720d36e196c9854911467168dda8c5f98ea7a (HEAD -> master)
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date:   Wed Jun 12 10:15:00 2024 +0800

    通用的UI设计，用一套代码同时为触屏和鼠标建模
```

图 8-8 第五次日志记录

## 第 9 章 UI 的个性化键盘控制——应用 keydown 和 keyup 键盘底层事件

### 9.1 分析和设计

上一阶段实现了一套通用代码对 PC 端和移动端的鼠标拖动/触屏滑动操作。对 PC 端用户而言，只有鼠标交互设计是远远不够的，很多页面需要通过键盘输入来触发事件，键盘事件是 Web 开发中非常重要的一部分，它们可以极大地提升用户的交互体验。keydown 和keyup 是 JavaScript 中两种常见的键盘事件类型，本阶段只探究了这两个基础 API 操作，用户可以在键盘响应区输入字符，响应区会反馈用户有关 keydown 和keyup 事件的响应信息，分析如图 9-1，部分设计如图 9-2。

1.keydown 事件：当按下键盘上的任意键时触发，并且会持续触发直到键被

释放。这个事件可以用来实现连续的输入响应，例如在用户持续按住某个键时，可以连续执行某些操作。

2.keyup 事件：当释放键盘上的键时触发。这个事件通常用于响应用户释放键的动作，比如在用户停止输入时进行数据处理或提交表单。

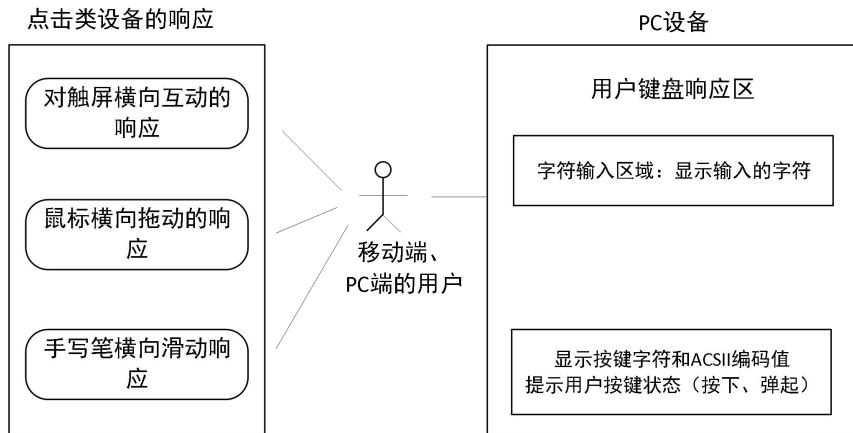


图 9-1 个性化 UI 键盘控制用例图

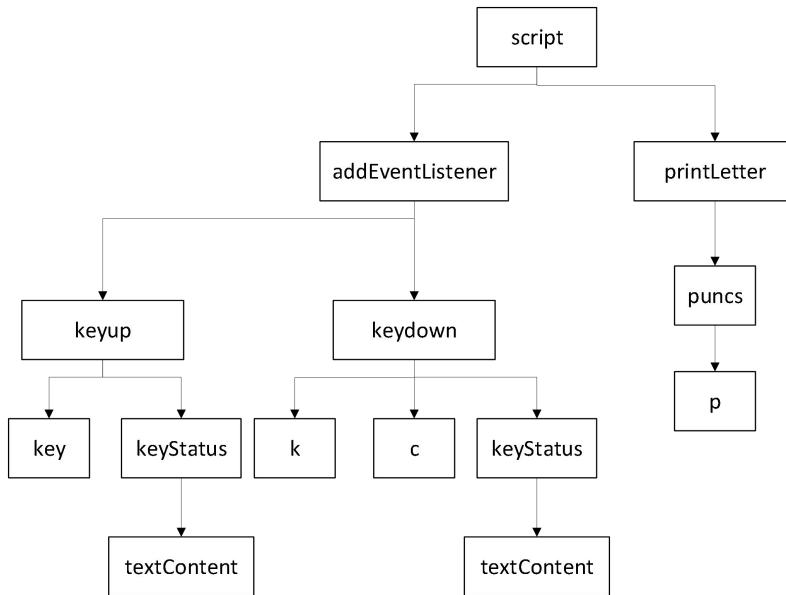


图 9-2 个性化 UI 键盘控制 DOM 树

## 9.2 项目的实现和编程

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，代码如代码块 9-3 所示。

```

//提出问题：研究利用"keydown"和"keyup"2个底层事件，实现同时输出按键状态和文本
//内容
$("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;
    }
})

```

代码块 9-3 键盘底层事件 JS 代码

### 9.3 项目的运行和测试

PC 端测试，在用户键盘响应区按下 a 键不松开，keydown 事件被触发，keyStatus 区域显示“按下键：a， 编码：65”，此时keyup 事件还未触发如图 9-4；松开 a 键，keyup 事件发生，keyStatus 区域显示“a 键已弹起”typeText 区域会显示“a 键已弹起”，如图 9-5。



图 9-4 keydown 底层事件触发



图 9-5 keydown 底层事件触发

本阶段项目 URL 为：<https://hdd1654231824.github.io/exp/1.6.html>，移动端用  
户可以扫描下方图 9-6 二维码查看运行效果。

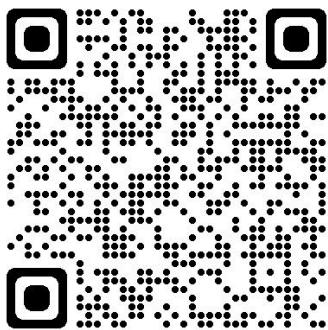


图 9-6 第六次二维码

## 9.4 项目代码的提交和版本管理

本阶段初步实现了 UI 的个性化键盘控制，成功提交代码后，gitbash 的反馈如图 9-7 所示。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git add .

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git commit -m"UI的个性化键盘控制--应用keydown和keyup键盘底层事件"
[master 5ea3013] UI的个性化键盘控制--应用keydown和keyup键盘底层事件
 11 files changed, 401 insertions(+), 34 deletions(-)
  create mode 100644 exp/1.6.html
```

图 9-7 git 第六次管理代码

git log 查看本阶段提交日志记录如图 9-8 所示。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git log
commit 5ea3013263c11f2f10314797f3f83f52d5e8fee4 (HEAD -> master)
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date:   Thu Jun 13 10:58:24 2024 +0800

  UI的个性化键盘控制--应用keydown和keyup键盘底层事件
```

图 9-8 第六次日志记

## 第 10 章 用 git 工具展开代码的版本管理和发布

### 10.1 跨世纪的经典工具 Git bash

Git 是一个版本管理控制系统（缩写 VCS），它可以在任何时间点，将文档的状态作为更新记录保存起来，也可以在任何时间点，将更新记录恢复回来。

Git 是分布式版本控制系统的范式中的明显领导者。Git 最初由 Linus Torvalds 开发，作为 Linux 内核的源代码控制管理（SCM）系统，以取代专有的 SCM 比特管理器，后来征服了大多数开源世界，也被许多组织用于他们的私有/专有项目。Git 提供了一个完整的工具箱，可以帮助我们管理自己的代码。在开发中，Git 已成为现在主流的一种代码托管技术（版本管理工具），基本上大多数的公司都在使用 Git 进行协同开发。很多代码托管平台也是基于 Git 来实现的。

## 10.2 Git 分层操作

git 的工作总共分四层，如图 10-1，其中三层是在自己本地也就是说 git 仓库，包括了工作目录、暂存区和本地仓库。工作目录就是我们执行命令 `git init` 时所在的地方，也就是我们执行一切文件操作的地方；暂存区和本地仓库都是在`.git` 目录下，因为它们只是用来存数据的。远程仓库在中心服务器，也就是我们做好工作之后推送到远程仓库，或者从远程仓库更新下来最新代码到本地。Git 所存储的都是一系列的文件快照，然后 git 来跟踪这些文件快照，发现哪个文件快照有变化他就会提示你需要添加到暂存区或是提交到本地仓库来保证你的工作目录是干净的。

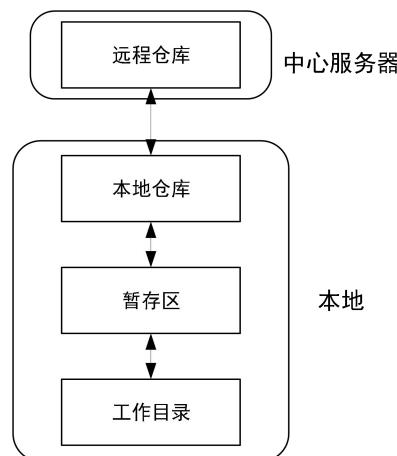


图 10-1 git 工作层

这个要怎么理解呢？git 中的文件有两种状态，一种是被跟踪的，也就是提交到本地仓库的文件，因为本地仓库要保管它们当然要跟踪他们，对他们负责，还有一种就是未被跟踪的。那么当我们添加新的文件时，他不是被跟踪的，因为本地仓库里面没有这个文件，他是外来的，本地仓库目前还不需要对他们负责。但是如果是对仓库已经存在的文件进行修改，那么这些文件就是被跟踪的文件，

就可以通过 `git status` 查看他们的状态来进行相应的操作。当然我们也可以生成一个`.gitignore` 文件，里面指定要忽略的文件类型，然后这些文件就不会被跟踪，不管怎么改变他们，`git status` 都不会提示你需要做什么操作。

所以当我们在工作目录中进行文件操作后，要先使用 `git add` 命令添加到暂存区，然后再将暂存区中刚添加的文件快照用 `git commit` 命令提交到本地仓库，然后再将本地仓库的最新状态文件快照推送到远程仓库。这个文件快照其实就是各个文件在被添加到暂存区时的状态，就和照相的一样，留下每个不同时刻的快照，方便以后查询，而 git 存储的就是这些一系列的快照。说到这个快照就要说说 git 的对象了。

## 10.3 Git 工作

Git 中存在两种类型的仓库，即本地仓库和远程仓库，那么我们如何搭建 Git 远程仓库呢？我们可以借助互联网上提供的一些代码托管服务来实现。其中比较常用的有 Github、码云、GitLab 等。这里用 Github 托管服务。

### 10.3.1 远程工作

Git 是一个版本控制文件的工具，然而，它是基于协作而构建的。在 2005 年，Linus Torvalds 需要一个轻而高效的工具来处理来自众多贡献者向 Linux 内核提出的大量补丁。他想要一个工具，可以让他和其他数百人在不疯狂的情况下工作。指导其开发的实用主义为我们提供了一个非常健壮的层，可以在计算机之间共享数据，而不需要一个中央服务器。Git 远程存储器是与我们的计算机上具有相同存储库的另一台计算机。在共享网络上托管相同存储库的每一台计算机都可以是其他计算机的远程计算机，如图 10-2。

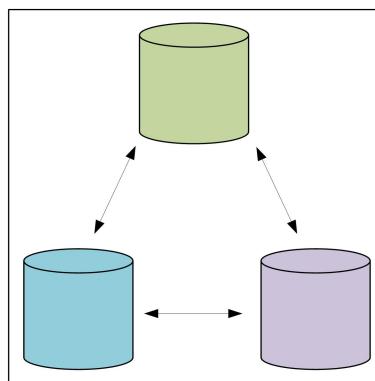


图 10-2 计算机共享

因此，远程 Git 存储库只不过是我们本地创建的同一个 Git 存储库的远程副本。如果我们可以使用 SSH、HTTPS 或自定义 `git://` 协议访问该主机，则可以与修改保持同步。

### 10.3.2 创建 github 远程仓库

如何创建一个 github 仓库呢？用浏览器进去 [github.com](https://github.com), 登录自己的 github, 手工建立一个空仓库，转到 Top Repositories 选项卡，如图 10-3，单击绿色的 New 按钮，然后为仓库选择一个名称，并把空仓库的名字设为 http 服务器的名字，注意图 10-4 中红色的字。

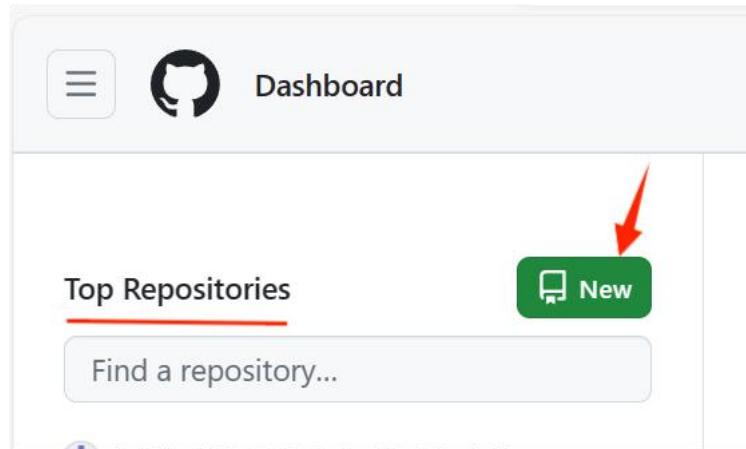


图 10-3 创建 github 仓库第一步

A screenshot of the 'Create a new repository' page. At the top, there's a header with a GitHub icon and the text 'New repository'. Below it is a search bar with the placeholder 'Type / to search'. On the right side of the header are several small icons. The main form area has a title 'Create a new repository'. It asks if the user already has a project repository elsewhere and provides a link to 'Import a repository'. It also notes that required fields are marked with an asterisk (\*). The 'Owner' field is set to 'hdd1654231824'. The 'Repository name' field contains 'hdd1654231824.github.io'. A red annotation with the text '仓库名保持一致' (Keep the repository name consistent) is placed over this field. Below the name field is a note: 'hdd1654231824.github.io is available.' The form includes a 'Description (optional)' field with a text input box. At the bottom, there are two radio buttons for 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is described as 'You choose who can see and commit to this repository.'

图 10-4 创建 github 仓库第二步

然后，我们可以为存储库编写一个描述。这对于让来访问我们的个人资料的人更好地了解我们的项目的目的非常有用。我们将公共存储库创建为公共的，因为如我们前面所述，私人存储库是有成本的。然后，我们用一个自述文件来初始化它。选择这个 GitHub 会为我们进行第一次提交，初始化现在准备使用的仓库。

### 10.3.3 用 Git Bash 建立本地仓库

本地仓库是我们整个项目的容器，它的每个文件或子文件夹都以同样的方式

存储在里面。从物理上讲，本地仓库是一个包含一个特殊的.git文件夹的文件夹。

通过以下步骤可以建立一个本地目录：

1.选择一个你想要的本地仓库的路径位置

2.右击选择 Git Bash Here，如图 10-5

3.输入 git init 初始化本地仓库，如图 10-6



图 10-5 Git Bash Here

```
pc@LAPTOP-SBOSJNSS MINGW64 /  
$ git init  
Initialized empty Git repository in D:/Study/Git/.git/
```

图 10-6 初始化本地仓库

完成上述步骤本地目录就生成了，该命令将创建一个名为.git的子目录，如图 10-7 该目录还有初始化的 git 仓库中所有的必须文件，这些是仓库的骨干，但仅仅是初始化，项目里的文件还没有被跟踪。本地目录也可以省略，会自动生成一个目录。

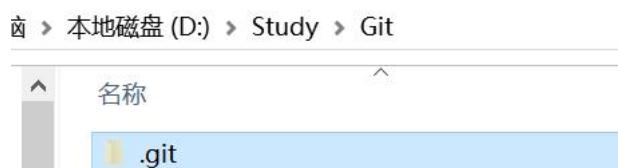


图 10-7 .git 子目录

如果不想要某个路径下的本地仓库，可以使用 rm -rf .git 命令撤销 git init 命令，如图 10-8。

```
MINGW64:/c/Users/pc/Desktop
pc@LAPTOP-SBOSJNSS MINGW64 ~/Desktop
$ git init
Initialized empty Git repository in C:/Users/pc/Desktop/.git/
pc@LAPTOP-SBOSJNSS MINGW64 ~/Desktop (master)
$ rm -rf .git

pc@LAPTOP-SBOSJNSS MINGW64 ~/Desktop
$ |
```

图 10-8 撤销 git init

本项目本地仓库初始化步骤如图 10-9，先使用 mkdir 命令创建一个名为 webUI 的文件夹并将该文件夹 git init 为本地仓库，再设置代码仓库用户名 user.name 和用户邮箱 user.email 分别为黄丹丹@江西科技师范大学、1654231824@qq.com，最后建立本项目代码需要的几个文件夹，myCSS.css 用于存放 css 样式代码，myJS.js 存放 javaScript 代码，index.html 文件存放 html、css、js 整合后的代码。

```
pc@LAPTOP-SBOSJNSS MINGW64 ~
$ cd /d/Study

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study
$ mkdir webUI

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study
$ cd webUI

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI
$ git init
Initialized empty Git repository in D:/study/webUI/.git/

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git config user.name 黄丹丹@江西科技师范大学

pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git config user.email 1654231824@qq.com
```

图 10-9 初始化本地仓库

#### 10.3.4 项目提交

本项目六次迭代提交日志记录如图 10-10 至 10-16。

```
pc@LAPTOP-SBOSJNSS MINGW64 /d/Study/webUI (master)
$ git log
commit 4e63314a82ee878bfae237eae056a5cdf034b2e1 (HEAD -> master)
Author: 黄丹丹 <1654231824@qq.com>
Date:   Tue May 28 10:10:21 2024 +0800

“三段论”式的内容设计概要。
```

图 10-10 第一次代码提交

```
commit abb51a5a0603dbfbc89685b1aa71f6a4bdd36626
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date: Mon Jun 10 16:10:23 2024 +0800
```

移动互联网时代的响应式设计（窄屏）

图 10-11 第二次代码提交

```
commit d258231eb39ac7fcb25194137a1e9a7a032e56a4
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date: Mon Jun 10 22:25:41 2024 +0800
```

响应式设计（宽屏和窄屏）

图 10-12 第三次代码提交

```
commit 9a423a2a5c9c5d71929c773a460cfa2bdaffb6f8
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date: Wed Jun 12 09:48:30 2024 +0800
```

UI设计之鼠标模型的分析和控制

图 10-13 第四次代码提交

```
commit 7d5720d36e196c9854911467168dda8c5f98ea7a
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date: Wed Jun 12 10:15:00 2024 +0800
```

通用的UI设计，用一套代码同时为触屏和鼠标建模

图 10-14 第五次代码提交

```
commit 5ea3013263c11f2f10314797f3f83f52d5e8fee4 (HEAD -> master)
Author: 黄丹丹@江西科技师范大学 <1654231824@qq.com>
Date: Thu Jun 13 10:58:24 2024 +0800
```

UI的个性化键盘控制--应用keydown和keyup键盘底层事件

图 10-15 第六次代码提交