

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): Customized UI design and Programming

based on Web client technology

院 (系): 元宇宙产业学院

专 业: 21 计科 2 班

学生姓名: 俞嘉美

学 号: 20213612

指导教师: 李健宏

2024 年 6 月 17 日

目录

基于 Web 客户端技术的个性化 UI 的设计和编程	5
(Customized UI design and Programming based on Web client technology)	5
1. 前言	5
1.1 毕设任务分析	5
1.2 研学计划	6
1.3 研究方法	6
2. 技术总结和文献综述	7
2.1 Web 平台和客户端技术概述	7
2.2 项目的增量式迭代开发模式	9
3. 内容设计概要	10
3.1 分析和设计	11
3.2 项目的实现和编程	12
3.3 项目的运行和测试	14
3.4 项目的代码提交和版本管理	15
4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计	16
4.1 分析和设计	16
4.2 项目的实现和编程	17
4.3 项目的运行和测试	20
4.4 项目的代码提交和版本管理	21
5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI	21
5.1 分析和设计	22
5.2 项目的实现和编程	24
5.3 项目的运行和测试	30
5.4 项目的代码提交和版本管理	31
6. 个性化 UI 设计中对鼠标交互的设计开发	32
6.1 分析和设计	32
6.2 项目的实现和编程	34
6.3 项目的运行和测试	38
6.4 项目的代码提交和版本管理	39
7. 对触屏和鼠标的通用交互操作的设计开发	40
7.1 分析和设计	40
7.2 项目的实现和编程	41
7.3 项目的运行和测试	47
7.4 项目的代码提交和版本管理	48
8. UI 的个性化键盘交互控制的设计开发	49
8.1 分析和设计	49
8.2 项目的实现和编程	51
8.3 项目的运行和测试	61
8.4 项目的代码提交和版本管理	62
9. 谈谈本项目中的高质量代码	62
10. 用 gitBash 工具管理项目的代码仓库和 http 服务器	63
10.1 经典 Bash 工具介绍	63

10.2 通过 gitHub 平台实现本项目的全球域名	63
10.3 创建一个空的远程代码仓库	63
10.4 设置本地仓库和远程代码仓库的链接	64
参考文献:	67
写作指导:	67

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 XXX

【摘要】

本项目基于 Web 客户端技术开发,以适应移动互联网时代 Web application 的前端需求。本文通过广泛查阅技术资料和相关文献,特别是 mozilla 组织的 MDN 社区的技术实践文章,通过了 HTML 内容建模、CSS 样式设计和 JavaScript 功能编程的基本技术和技巧。结合本学科的核心课程知识,实现了一个个性化的基于 web 客户端技术的响应式用户界面(UI),该 UI 能够较好的适配各种屏幕。在功能上,通过 DOM 技术和事件监听实现了对鼠标、触屏和键盘的底层事件的支持并实现了流畅地响应。基于面向对象的思想,为鼠标和触屏设计了对象模型,并通过代码实现了对这类指向性设备的模拟。为了能更好地应用工程思想、设计和开发管理项目,本项目采用了软件工程的增量式开发模式。共进行了 6 次项目迭代开发,每次迭代都经历了 ADIT(分析(Analysis)、设计(Design)、实施(Implementation)、测试(test))四个经典开发阶段。通过逐步求精的方式编写了 UI 应用程序。最后,为了分享和共享代码,与其他开发者合作,使用了 git 工具进行代码和开发过程日志记录。总共进行了 7 次代码提交操作,详细记录和展现了开发思路和代码优化的过程,然后通过 gitbash 将项目上传到 GitHub 并建立了自己的代码仓库,将该代码仓库设置成为了 HTTP 服务器,实现了全球便捷访问。

1. 前言

我们想要开发一个网页,运用了 web 和 javascript 的技术设计了一个项目,这个项目要运用三段式结构,对自己喜欢的书籍进行推荐,还需要有响应式设计,不论在 pc 端还是在手机都能运行,因为开发人员较少,我们采用增量模型来开发这个项目,这样才能用较少的人员逐步开发一个项目。在开发项目前,我们需要有一系列的准备阶段,以下就是这个项目的准备阶段。

1.1 毕设任务分析

首先要确定研究应用软件的开发方向,特别注意 Web 应用开发,因为这是一个需求大且与现代技术紧密相关的领域。要实现跨平台使用,就要利用 Web 标准如 HTML5 和 ECMAScript,实现代码的跨硬件和操作系统运行,最终实现一次编写,多处运行的目标。面对要实现面对对象的编程需求,我们要使用 JavaScript 等面向对象编程语言,结合 HTML 和 CSS,采用 MVC 设计模式,构建 Web 应用。运用 ADIT(分析、设计、实施、测试)开发模式,结合版本管理工具 Git,进行项目管理和多人合作。

1.2 研学计划

1.在项目开始的初期，我将投入时间对毕业设计涉及的领域和所需技能进行初步了解，包括阅读相关书籍和在线文档，以建立一个坚实的基础。

2.选择自己感兴趣的技术实践路线，整合学习核心技术，参考导师的案例项目，理解技术之间的关系及其在项目中的作用。

3.在理解技术之后，开始实际编写一些简单的代码，提升代码能力，实现技术部署，并在实践中发现问题、解决问题。

4.学会编写简单的代码后，按照软件工程的标准，从问题分析和设计开始，研究技术方案，编写代码，进行调试和性能优化。在写项目的同时，撰写开发文档，总结实践经验，并结合理论知识撰写毕业论文。

参考资料：

1. 经典书籍：参考"O'Reilly Media"和"Manning Publications"出版的计算机科学技术书籍，以及网络上的电子版资源。

2. 在线文档：利用 W3C 组织的 Web 标准和 Mozilla 基金会的 MDN 文档网站，学习 Web 技术和最佳实践。

3. 学术平台：使用知网等学术平台，了解毕设选题的理论研究和学术规范，为论文撰写提供参考。

1.3 研究方法

1. 文献法：通过阅读书籍、在线文档、社区论坛、期刊和会议报告，深入学习和总结本领域的知识和技术。

2. 模型研究法：对软硬件对象进行抽象建模，使用 UML 建立模型，再通过面向对象编程语言实现模型，验证和优化设计。

3. 实践验证：将所学知识和理论应用到实际开发中，通过实践来验证代码和系统的有效性和高效性。

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

1989 年，蒂姆·伯纳斯-李爵士发明了万维网（参见原始提案）。他创造了“万维网”这一术语，编写了第一个万维网服务器“httpd”和第一个客户端程序（浏览器和编辑器）“WorldWideWeb”，时间是 1990 年 10 月。

他编写了第一个“超文本标记语言”（HTML）版本，这是一种具有超文本链接能力的文档格式语言，成为 Web 的主要出版格式。随着 Web 技术的普及，他对 URI、HTTP 和 HTML 的初始规范进行了完善和讨论。

2.1.2 万维网联盟

1994 年，在许多公司不断向 Web 投入更多资源的情况下，决定成立万维网联盟。蒂姆·伯纳斯-李爵士开始领导万维网联盟团队进行关键性工作，以促进一致的架构，适应构建网站、浏览器和体验 Web 所有功能的设备的快速进步的网络标准。

在创立万维网联盟时，蒂姆·伯纳斯-李爵士创建了一个同辈社区。网络技术的发展速度如此之快，因此组建一个专门的组织来协调网络标准变得至关重要。蒂姆接受了麻省理工学院的邀请，后者在组建行业联盟方面有着丰富的经验，并负责在麻省理工学院设立 W3C。他从一开始就要求 W3C 具有全球影响力。

2.1.3 Web 平台与 Web 编程

让我们先简要了解一下 Web，简称为万维网。大多数人说“Web”而不是“万维网”，我们也将遵循这一惯例。Web 是由世界各地计算机用户共享的一系列文档，称为网页。不同类型的网页执行不同的功能，但至少，它们都在计算机屏幕上显示内容。这里的“内容”指的是文本、图片以及诸如文本框和按钮之类的用户输入机制。^[2]

Web 编程是一个庞大的领域，由不同的工具实现不同类型的 Web 编程。所有工具都与核心语言 HTML 协同工作，因此几乎所有 Web 编程书籍都或多或少地介绍了 HTML。本教材涵盖了 HTML5、CSS 和 JavaScript，并对其进行了深入的介绍。这三种技术被公认为客户端 Web 编程的支柱。在客户端网页编程中，所有网页计算都在客户端计算机（即用户的计算机）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）^[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。

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。

2.1.4 客户端网页应用的生命周期

客户端网页应用的生命周期通常从用户打开浏览器访问网页开始，经历加载、解析、执行 JavaScript、渲染 DOM 树、页面交互响应、资源加载和优化等阶段，直至用户关闭浏览器标签页或窗口结束，期间可能还包括错误处理、用户输入响应、网络请求、状态管理和性能优化等动态交互过程。

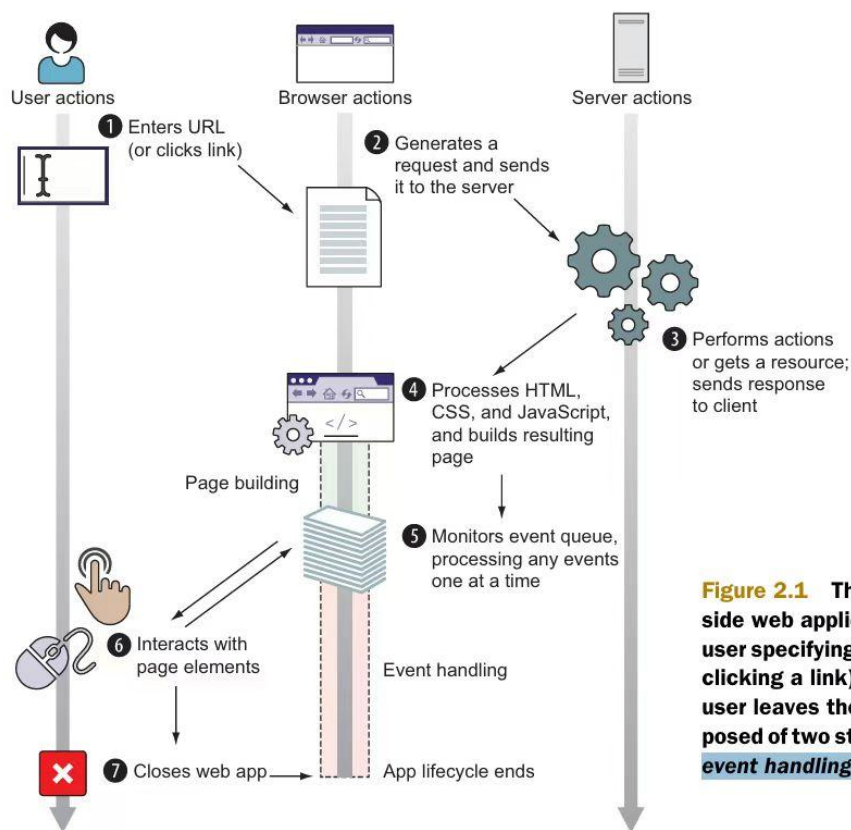


图 2-1 软件生命周期图

2.2 项目的增量式迭代开发模式

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

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

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

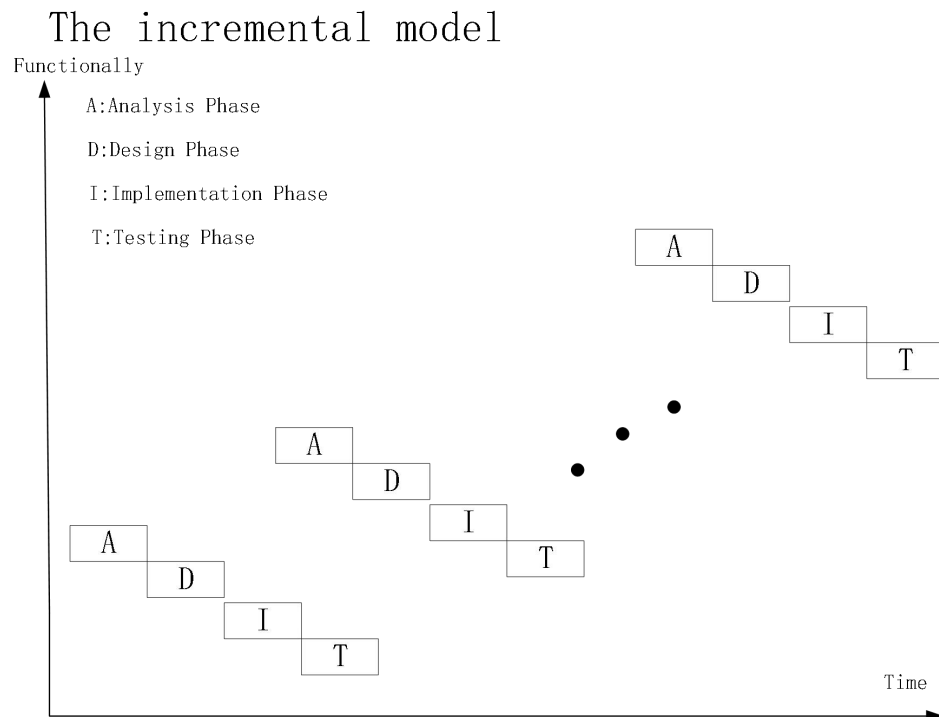


图 2-1

增量模型 在增量模型中，软件分一系列步骤进行开发。开发人员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，但不包括详细信息。图中显示了增量模型的概念。在第二个版本中，添加了更多的细节，而一些没有完成，系统再次测试。如果有问题，开发人员就知道问题在于新功能。在现有的系统正常工作之前，它们不会添加更多的功能。此过程，直到添加所有所需的功能^[5]。

2.3 瀑布模型

瀑布模型（Waterfall Model）是一种经典的软件开发生命周期模型，它将软件开发过程划分为一系列阶段性的活动，每个阶段完成后才能进入下一个阶段。这些阶段通常包括需求分析、设计、实现（编码）、测试、部署和维护。每个阶段都像瀑布的水流一样，顺序流动且不可逆，即前一个阶段的输出是后一个阶段的输入，任何阶段的变更都可能影响整个项目的进度。

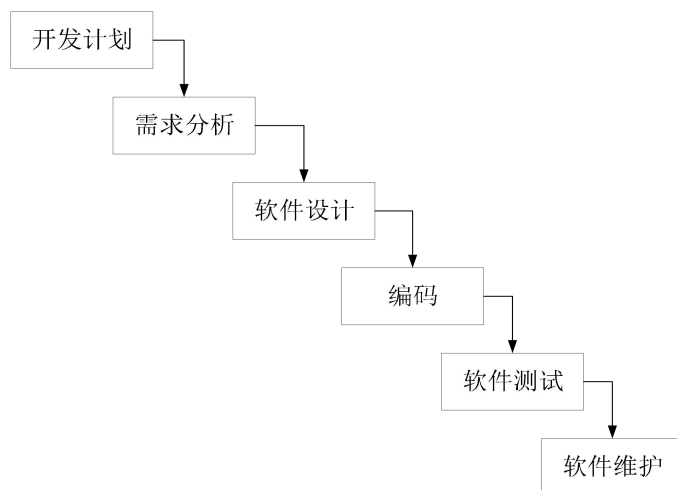


图 2-3 瀑布模型图

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：

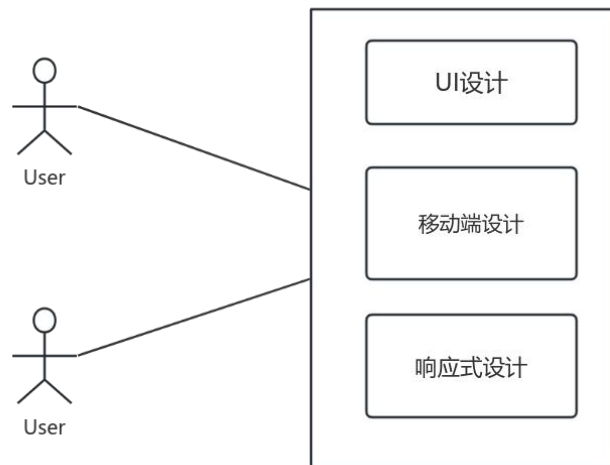


图 3-1 用例图

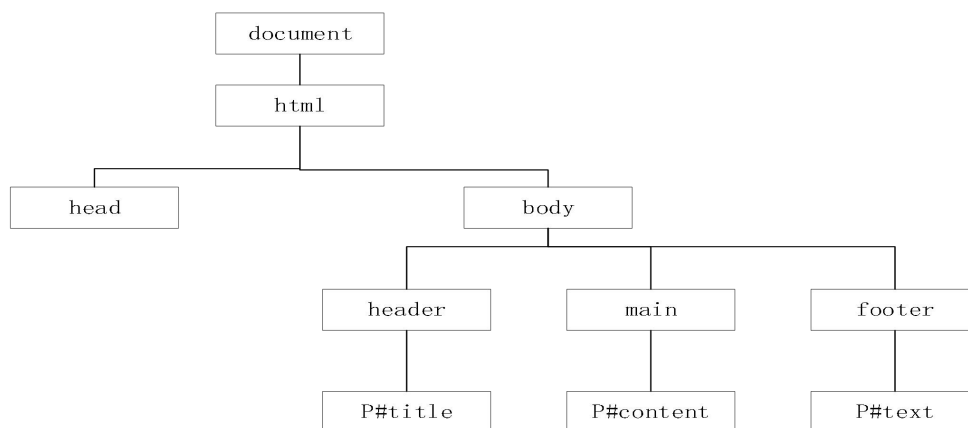


图 3-2Dom 树

3.2 项目的实现和编程

JavaScript 由三个文件组成，分别是 html 文件，css 文件，script 文件。这三个文件用的语言分别组成了前端技术的基础。其中 html 语言用于网页的架构。CSS 文件负责网页的样式，美化。JavaScript (JS) 用于负责网页的行为。

其中 html 可以分为三段式结构，分别为 header、main、footer。header 为头部，用于放置标题和导航栏，main 文件用于放置内容文件，包括书的封面和图片等，还包括了部分按钮用于打开书文件，下载书文件，书的介绍等。Footer 部分用于放置自己的信息文件等。如代码块 4.2 所示。

一、HTML 代码编写如下：

```
<header>
  <p id="book">
    傲慢与偏见
```

```

    </p>
</header>
<nav>
    <button>向前</button>
    <button>向后</button>
    <button>其他</button>
</nav>
<main id = 'main'>

```

《傲慢与偏见》是英国作家简·奥斯汀于 1813 年出版的一部经典小说。这部小说以 19 世纪英国上层社会为背景，讲述了女主角伊丽莎白·班内特与富有但自大的达西先生之间的爱情故事。故事开始时，班内特家五个姐妹中的长女珍和宠儿们的母亲都希望她们能够嫁给有钱人，以改善家庭状况。然而，聪明机智且有主见的伊丽莎白并不愿意因为金钱而妥协自己的幸福。她不被达西的高傲态度所动摇，并对他抱有偏见。随着故事的发展，伊丽莎白逐渐认识到自己的偏见，并得知了达西的真实情况。他原来并非如表面上那般傲慢，反而是一个善良、正直和谦虚的人。最终，两人克服了彼此间的误解与困难，相互坦诚相待，走到了一起。
《傲慢与偏见》展示了早期 19 世纪社会的等级观念和婚姻观念，并深入探讨了人性的复杂性和成长的重要性。这部小说以生动的人物形象、精致的语言和幽默风趣的对话而闻名，被誉为英国文学的经典之作，深受读者欢迎。

```

</main>

<footer>
    <p id="statusInfo">
        俞嘉美 @ 江西科技师范大学 2025
    </p>
</footer>

```

二、CSS 代码编写如下：

```

<style>
*{
    margin: 10px;
    text-align: center;
font-size:30px ;
}
header{
    border: 2px solid rgb(247, 0, 255);
    height: 160px;
}

main{
    border: 2px solid rgb(241, 120, 245);
    height: 350px;
}

```

```

    footer{
        border: 2px solid rgb(238, 148, 231);
        height: 80px;
    }
    a{
        display: inline-block ;
        padding:10px ;
        color: white;
        background-color: rgb(126, 162, 212);
        text-decoration: none ;
    }
</style>

```

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。

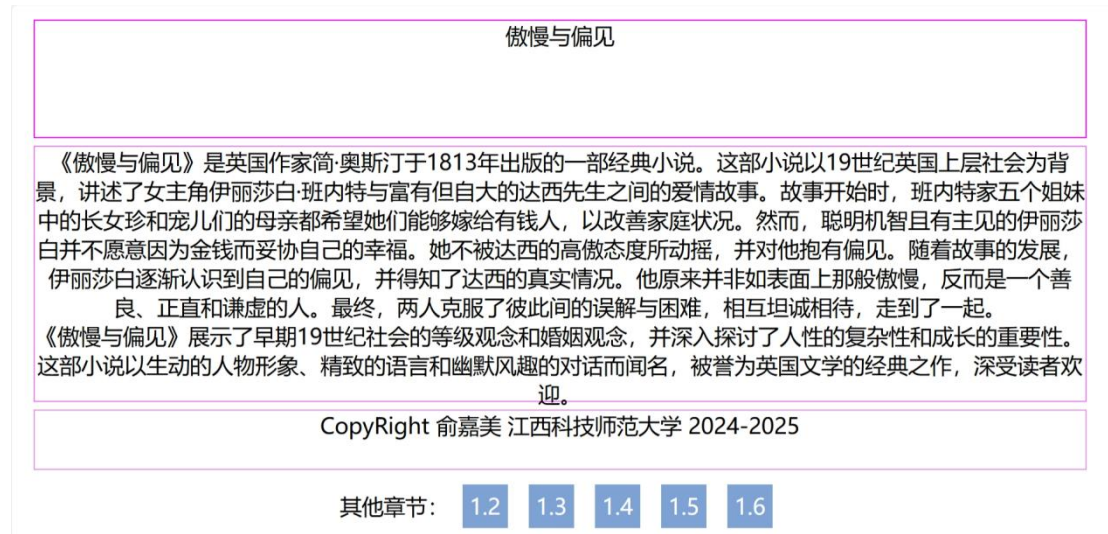


图 3-3 PC 端运行效果图

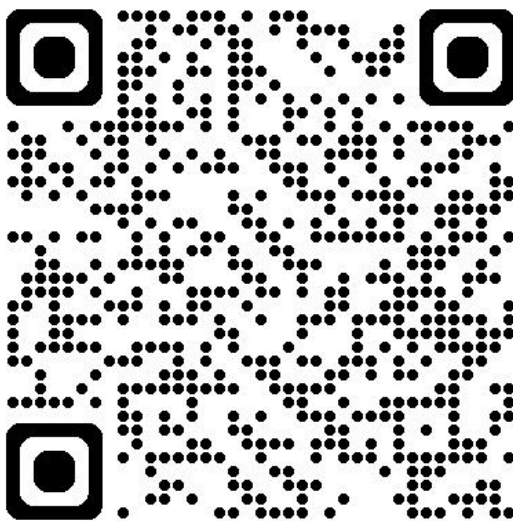


图 3-4 移动端运行效果

3.4 项目的代码提交和版本管理

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

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /  
$ mkdir webUI  
$ cd webUI  
$ git init  
$ git config user.name 俞嘉美@科师大  
$ git config user.email 1426674952@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 项目第一版：“三段论”式的内容设计概要开发
```

成功提交代码后，gitbash 的反馈如下所示：

```
yyy@LAPTOP-5TB3C0CJ MINGW64 /yu/example/yujiamei (main|REBASE 1/1)  
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发  
[detached HEAD 967beb8] 项目第一版：“三段论”式的内容设计概要开发  
1 file changed, 56 insertions(+)  
create mode 100644 example/yujiamei/1.1.html
```

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：

Author: 俞嘉美@科师大 <1426674952@qq.com>
Date: Sun Jun 2 12:34:49 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

响应式设计——适应显示硬件

计算机所使用的显示硬件千差万别，显示器的大小和分辨率取决于成本。设计师们选择让网页给出一般的布局准则，并允许浏览器选择如何在给定的计算机上显示页面，而不是为每种类型的显示提供每个网页的版本。因此，一个网页不能提供很多细节。例如，网页的作者可以指定一组句子组成一个段落，但作者不能指定诸如一行的确切长度或是否缩进段落开头等细节。^[1]

允许浏览器选择显示细节会产生一个有趣的结果：当通过两个浏览器或在硬件不同的两台计算机上浏览时，网页可能会显示不同的内容。如果一个屏幕比另一个屏幕宽，则可以显示的文本行的长度或图像的大小不同。重点是：网页给出了关于期望呈现的一般指导方针；浏览器在显示页面时选择详细信息。因此，同一网页在两台不同的计算机或不同的浏览器上显示时可能会略有不同。^[1]

用 JavaScript 开动态读取显示设备的信息，然后按设计，使用 js+css 来部署适配当前设备的显示的代码。

4.1 分析和设计

这一步是项目的第二次迭代开发，我们设想应用先支持世界上用的最多的机型，然后用百分比的方式大致对多样化的屏幕进行设计区域的划分。本次还在标题和内容之间增加了一个导航栏。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 5-1 用例图所示：

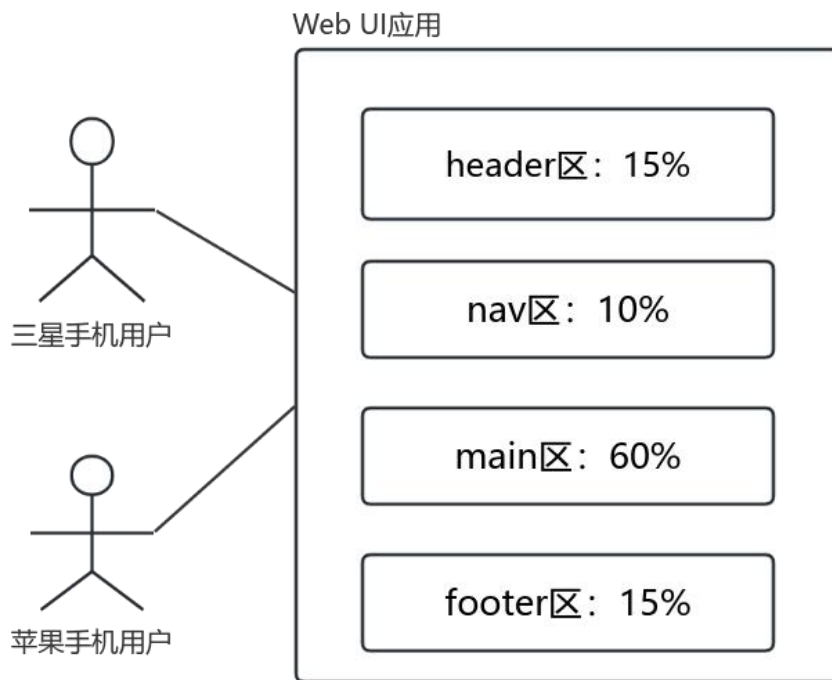


图 4-1 窄屏手机用户的用例图

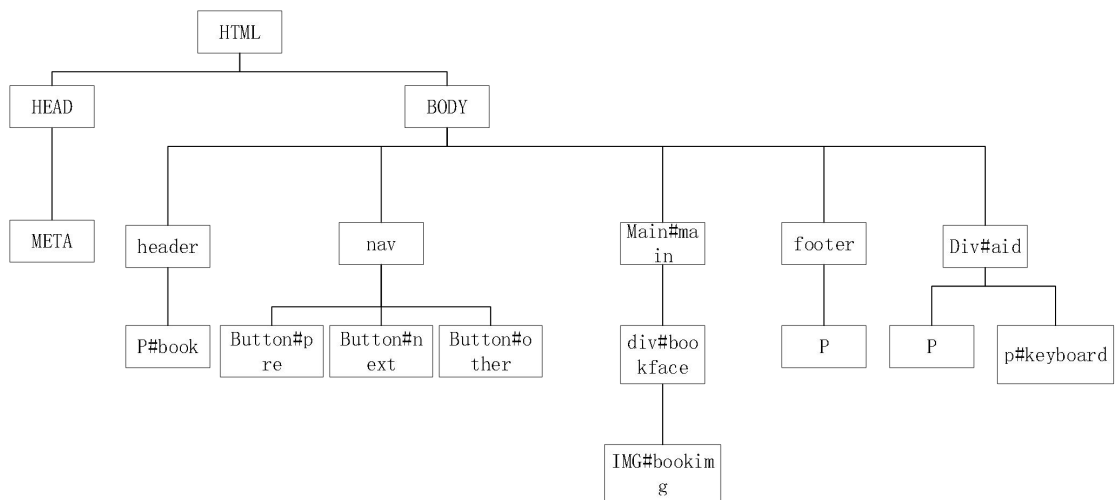


图 4-2 1.3.htmlDom 树

4.2 项目的实现和编程

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 rgb(247, 0, 255);
        height: 15%;
        font-size: 1.66em;
    }
    main{
        border: 2px solid rgb(241, 120, 245);
        height: 70%;
        font-size: 1.2em;
        overflow: auto; /* 添加滚动条, 防止内容过多时溢出 */
    }
    nav{
        border: 2px solid #b09cd5;
        height: 10%;
    }
    nav button{
        font-size: 1.1em;
    }
    footer{
        border: 2px solid rgb(238, 148, 231);
        height: 8%;
    }
}
</style>

```

代码块 4-1

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 = 35;
    const baseFont = UI.appWidth / LETTERS;

    document.body.style.fontSize = baseFont + "px";
    document.body.style.width = UI.appWidth - 3 * baseFont + "px";
    document.body.style.height = UI.appHeight - 6 * baseFont + "px";

    // 假设的内容段落数组
    var contentParagraphs = [
        "《傲慢与偏见》是英国作家简·奥斯汀于 1813 年出版的一部经典小说。

```

这部小说以 19 世纪英国上层社会为背景，讲述了女主角伊丽莎白·班内特与富有但自大的达西先生之间的爱情故事。故事开始时，班内特家五个姐妹中的长女珍和宠儿们的母亲都希望她们能够嫁给有钱人，以改善家庭状况。然而，聪明机智且有主见的伊丽莎白并不愿意因为金钱而妥协自己的幸福。她不被达西的高傲态度所动摇，并对他抱有偏见。随着故事的发展，伊丽莎白逐渐认识到自己的偏见，并得知了达西的真实情况。他原来并非如表面上那般傲慢，反而是一个善良、正直和谦虚的人。最终，两人克服了彼此间的误解与困难，相互坦诚相待，走到了一起。",

"《傲慢与偏见》展示了早期 19 世纪社会的等级观念和婚姻观念，并深入探讨了人性的复杂性和成长的重要性。这部小说以生动的人物形象、精致的语言和幽默风趣的对话而闻名，被誉为英国文学的经典之作，深受读者欢迎。"

```
];
```

```
var currentPage = 0; // 当前显示的页面索引
```

```
// 添加改变内容的函数
```

```
function changeContent(direction) {  
    var content = document.getElementById('content');  
    var contentLength = contentParagraphs.length;  
  
    if (direction === 'prev') { // 向前翻页  
        if (currentPage > 0) {  
            currentPage--;  
        } else {  
            currentPage=contentParagraphs.length-1;  
            alert("已经是第一页了！接下来将翻转到最后一页");  
        }  
    } else if (direction === 'next') { // 向后翻页  
        if (currentPage < contentLength - 1) {  
            currentPage++;  
        } else {  
            currentPage=0;  
            alert("已经是最后一页了！接下来将翻转到第一页");  
        }  
    } else {  
        alert("未知的操作！");  
        return;  
    }  
  
    // 更新显示内容  
    content.textContent = contentParagraphs[currentPage];  
}
```

```
// 初始化内容显示第一页
```

```
changeContent('next'); // 假设第一页通过点击“向后”按钮来加载
</script>
```

代码块 4-2

4.3 项目的运行和测试



图 4-3 pc 端运行截图



图 4-4 窄屏运行效果图



图 4-5 二维码

4.4 项目的代码提交和版本管理

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 1.2.html
```

```
$ git commit -m 项目第二版：响应式设计的概要开发
```

成功提交代码后，`gitbash` 的反馈如下所示：

```
yyy@LAPTOP-5TB3C0CJ MINGW64 /d/Git/作业 (main)
$ git commit -m 项目第二版：响应式设计的概要开发
[main d5cd5c9] 项目第二版：响应式设计的概要开发
1 file changed, 5 insertions(+), 4 deletions(-)
```

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：

```
commit fd9f0843c172421cf24bc7ab23047d2e78772b88
Author: YUJM <YUJM@163.com>
Date: Sun Jun 2 13:20:32 2024 +0800
```

项目第二版窄屏代码实现：在实现响应式变化的基础上，增添了点击切换功能。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。

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.1 分析和设计

在移动互联时代，用户通过各种设备访问互联网，从小巧的智能手机到宽敞的桌面显示器，设备的屏幕尺寸和分辨率千差万别。这种多样性要求网页设计必须灵活适应，以确保在任何设备上都能提供出色的用户体验。

响应式设计的核心在于使用 CSS 语言的媒体查询功能，它允许开发者根据不同的屏幕尺寸和设备特性应用不同的样式规则。例如，可以为小屏幕设备设置较小的字体和简化的布局，而为大屏幕设备提供更丰富的视觉元素和更复杂的布局。此外，CSS 的流式布局和弹性盒子模型也是实现响应式设计的重要工具，它们让元素的大小和位置能够根据屏幕大小动态调整。

JavaScript 在响应式设计中扮演着增强交互性和动态功能的角色。它可以用来检测和适应屏幕大小的变化，动态加载或隐藏内容，以及创建适应不同设备的交互元素。例如，JavaScript 可以控制图片的加载，确保在小屏幕上不会加载过大的图片，从而加快加载速度并减少数据使用。

总之，通过 CSS 和 JavaScript 的结合使用，开发者可以构建出既美观又实用的响应式网页，无论用户使用何种设备，都能获得优质的访问体验。这要求开发者不断测试和优化他们的设计，确保它们在各种设备和屏幕尺寸上都能正常工作。

A:我们要做一个响应式设计的网页，这个页面中要包括页眉、导航、主内容区域、页脚以及一个额外的键盘响应区。页面应能根据不同屏幕尺寸进行自适应调整（是否增添这个键盘响应区来填补 `width>600px` 时的右边部分的空缺。**D:**在维持之前代码百分比的同时，增加更多断点来维持页面的精细化需求，同时为了方便增加断点，添加\$可以捕捉元素的代码，便于我们编程，对代码进行模块化操作，把不同的功能的代码放置到不同模块中去。由于页面中可能会有图片，网页中图片的缓存较慢，故考虑使用延迟加载图片、异步加载 JavaScript 脚本等技术来提升页面性能

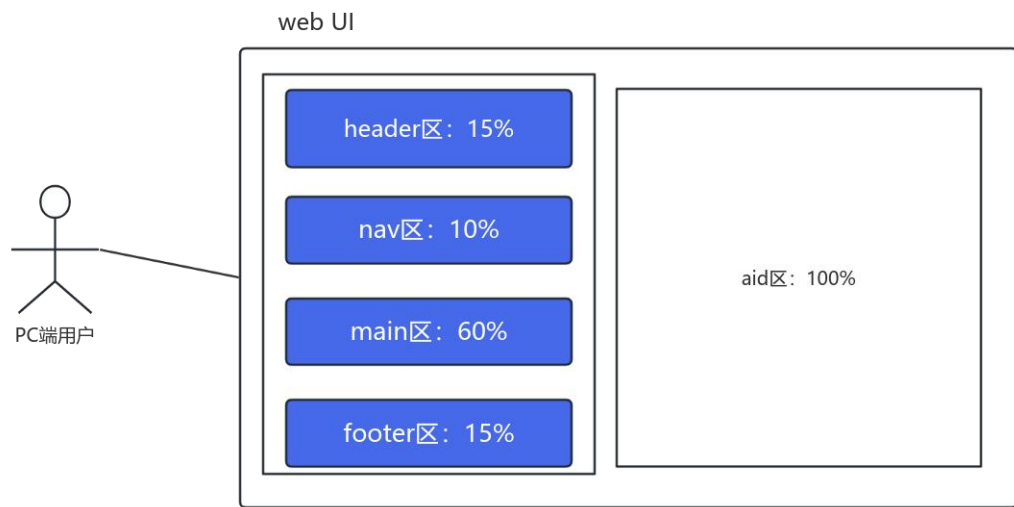


图 5-1 PC 端的用例图

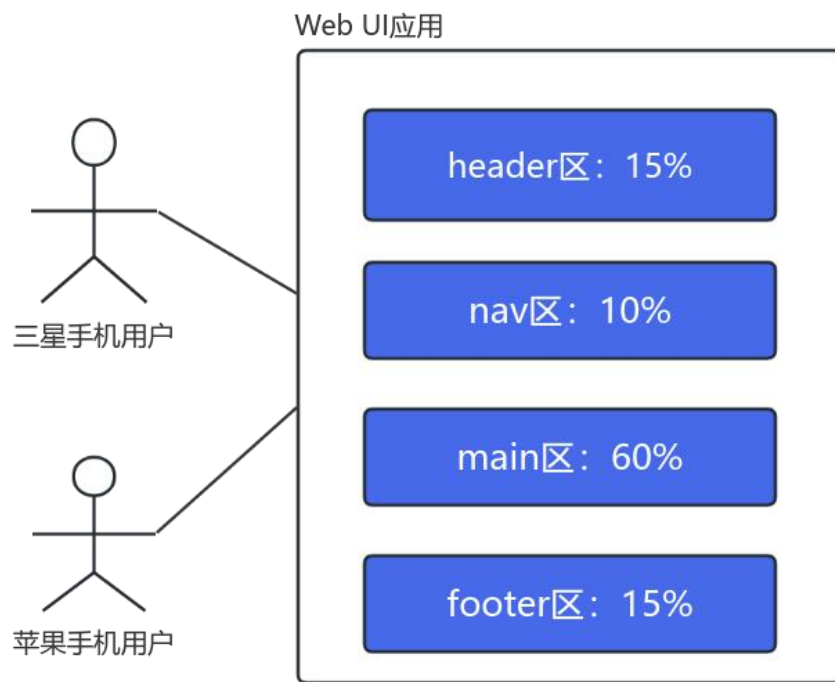


图 5-2 窄屏手机用户的用例图

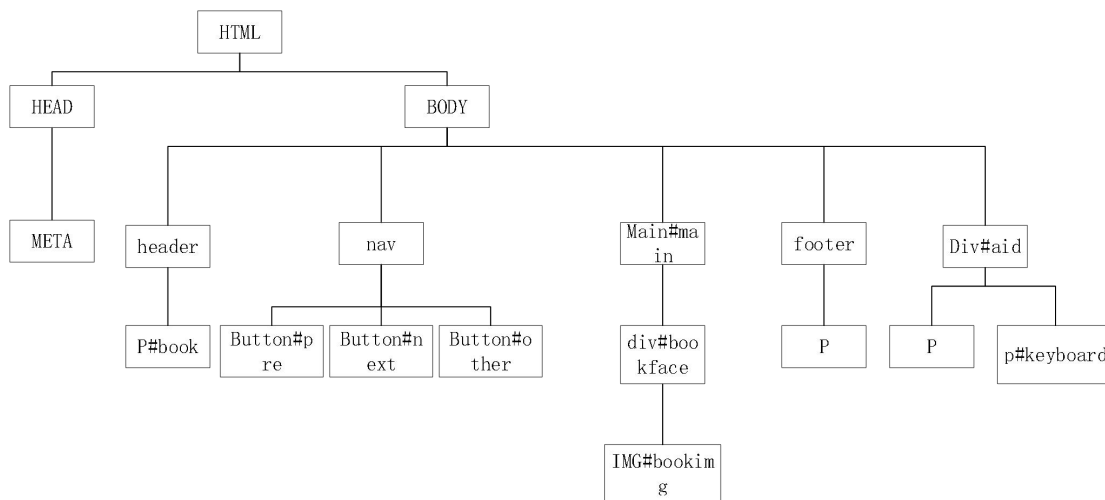


图 5-3 1.3.htmlDom 树

5.2 项目的实现和编程

1: 对于 HTML 架构，我们需要对页面比率设置百分比，确保在各个页面和谐融洽，对于超出 600px 的页面，我们把内容固定在 600px（用目录 `window.innerWidth` 就可以显示当前页面的宽度），小于的就全屏显示，这样就可以保证也页面的宽度，基于这个的基础上，我们用 `UI.appWidth` 接收内容的宽度，并根据这个宽度改进 `baseFont` 的计算逻辑，使其更加平滑地响应屏幕尺寸变化。项目代码块如代码块 5.1 所示：

```

<!doctype html>

<html lang="en">

  <head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width , initial-scale=1,
user-scalable=no">

    <title>响应式设计（宽屏和窄屏）</title>

  <style>

    *{

      margin: 10px;
    }
  
```



```

        text-align: center;
    }

    header{

        border: 2px solid rgb(247, 0, 255);

        height: 15%;

        font-size: 1.66em;
    }

    main{

        border: 2px solid rgb(241, 120, 245);

        height: 70%;

        font-size: 1.2em;

        overflow: auto; /* 添加滚动条, 防止内容过多时溢出 */
    }

    nav{

        border: 2px solid #b09cd5;

        height: 10%;
    }

    nav button{

        font-size: 1.1em;
    }

    footer{

        border: 2px solid rgb(238, 148, 231);

        height: 8%;
    }

    body{

        position: relative ;
    }

    #aid{

        position: absolute;

        border: 3px solid rgb(235, 155, 224);
    }

```

```

        top: 0.5em;

        left: 600px;
    }

    #bookface{

        width: 80%;

        height: 80%;

        border:1px solid red;

        background-color: blanchedalmond;

        margin:auto;
    }

    #bookimg{

        width: 100%;

        height:100%;

        object-fit:cover;

        margin: 0% 0%;
    }

</style>

</head>

<body >

    <header>

        <p id="book">

            傲慢与偏见

        </p>

    </header>

    <nav>

        <button>导航 1</button>

        <button>导航 2</button>

        <button>导航 3</button>

    </nav>

```

```

<main id="main">

<div id="bookface">

</div>

</main>


<footer>


    俞嘉美 @ 江西科技师范大学 2025


</footer>

<div id="aid">

    <p>用户键盘响应区</p>

    <p id="keyboard"></p>

</div>

<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 - 3*baseFont + "px" ;

    document.body.style.height = UI.appHeight - 5*baseFont + "px";

```

```

if(window.innerWidth < 900) {

    $("aid").style.display='none';

}

$("aid").style.width=window.innerWidth - UI.appWidth - 2*baseFont + 'px';

$("aid").style.height= document.body.clientHeight + 'px';


//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;

    $("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 $

</script>

</body>

</html>

```

代码块 5. 1

5.3 项目的运行和测试

手机端测试截图如下：



图 5-4 手机端测试截图

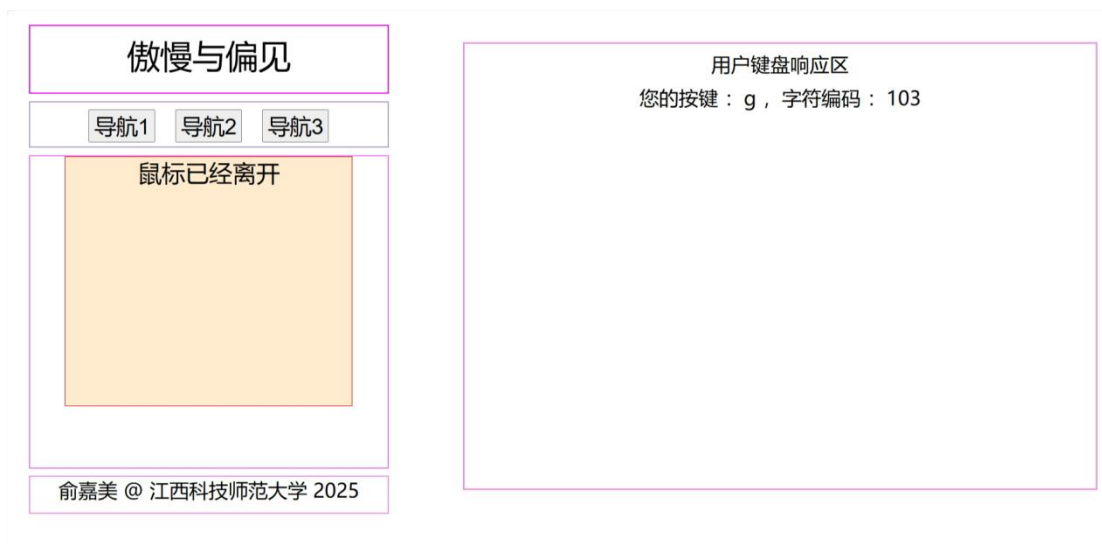


图 5-5 PC 端测试图



图 5-6 1.3.html 二维码

5.4 项目的代码提交和版本管理

编写好 1.3.html 代码，测试运行成功后，执行下面命令提交代码：

```
$ git add 1.3.html
```

`$ git commit -m 项目第三版：适用于宽屏和窄屏设计的内容设计概要开发（使用 CSS 语言的媒体查询功能，它允许开发者根据不同的屏幕尺寸和设备特性应用不同的样式规则）`

成功提交代码后，gitbash 的反馈如下所示：

```
yyy@LAPTOP-5TB3C0CJ MINGW64 /d/Git/作业 (main)
$ git commit -m 项目第三版: 适用于宽屏和窄屏设计的内容设计概要开发 (使用CSS语言的媒体查询功能, 它允许开发者根据不同的屏幕尺寸和设备特性应用不同的样式规则)
[main 8ab0286] 项目第三版: 适用于宽屏和窄屏设计的内容设计概要开发 (使用CSS语言的媒体查询功能, 它允许开发者根据不同的屏幕尺寸和设备特性应用不同的样式规则)
1 file changed, 10 insertions(+), 4 deletions(-)
```

项目代码仓库自此也开启了严肃的历史记录, 我们可以输入日志命令查看,

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示:

```
$ git log
commit 8ab028630ecb6ec8885557dd697f4daa3b730805 (HEAD -> main)
Author: 俞嘉美@科师大 <1426674952@qq.com>
Date: Sun Jun 16 22:22:33 2024 +0800

    项目第三版: 适用于宽屏和窄屏设计的内容设计概要开发 (使用CSS语言的媒体查询功能, 它允许开发者根据不同的屏幕尺寸和设备特性应用不同的样式规则)
```

6. 个性化 UI 设计中对鼠标交互的设计开发

6.1 分析和设计

我们需要展示一种为触屏和鼠标操作建立统一对象模型的方法, 通过响应式设计实现跨设备的兼容性。JavaScript 通过定义 UI 对象来动态调整页面尺寸和字体大小, 实现真正的响应式设计。同时, `mouse` 对象存储了鼠标状态信息, 并通过事件监听器捕捉鼠标行为, 如按下、移动和离开, 将状态更新到页面上。

我们还有自定义的`\$`函数, 用于简化 DOM 元素的选择过程, 使得代码更加简洁和易于维护。键盘事件也被纳入考虑, 通过监听`keypress`事件, 页面能够响应键盘输入并提供反馈。最后, 通过简单的逻辑判断, 页面元素如`#aid`可以根据屏幕宽度动态显示或隐藏, 进一步优化了用户体验。整体而言, 这段代码通过整合 HTML、CSS 和 JavaScript, 创建了一个能够自适应触屏和鼠标操作的响应式网页, 无论用户使用什么设备, 都能提供一致的交互体验和界面展示。

在分析阶段, 我们首先确定了页面需要满足的功能和性能要求。分析了用户在不同设备(触屏和鼠标)上的交互行为, 以及响应式设计的必要性。我们考虑到了不同屏幕尺寸和分辨率的适应性, 以及如何通过键盘与页面交互。此外, 还分析了如何通过媒体查询和动态样式调整来实现响应式布局。我们设计了 JavaScript 对象模型, 包括 UI 对象来管理页面尺寸和字体大小, 以及 mouse 对象来处理鼠标事件。此外, 还设计了一个自定义的`\$`函数来简化 DOM 操作。

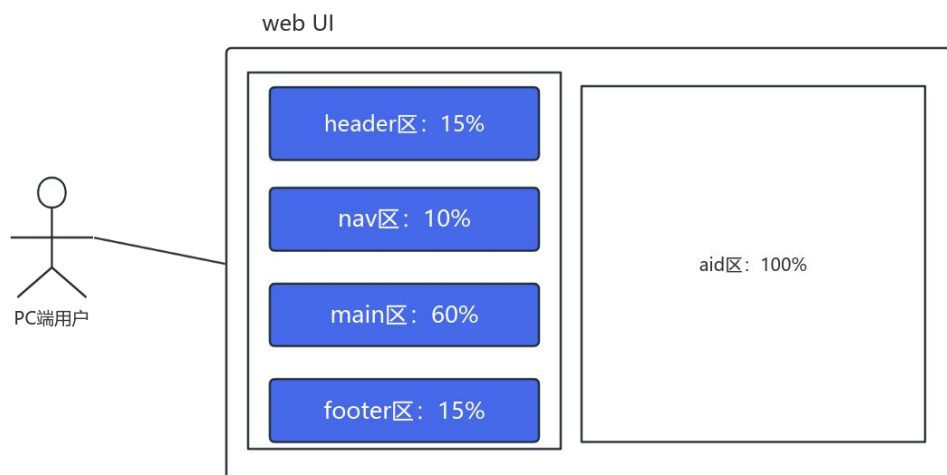


图 6-1 PC 端的用例图

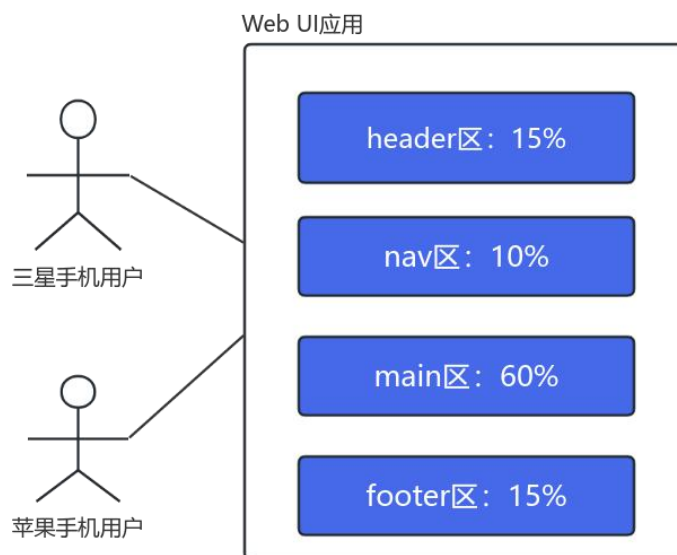


图 6-2 窄屏手机用户的用例图

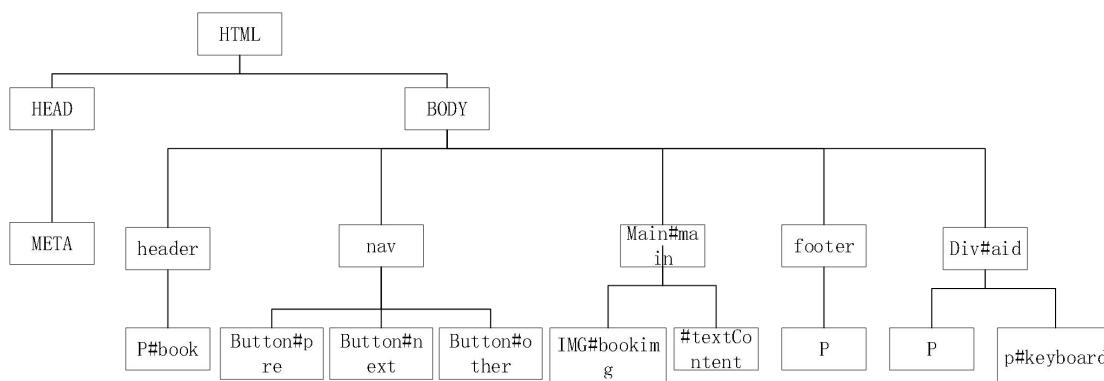


图 6-3 1.4.htmlDom 树

6.2 项目的实现和编程

HTML 提供了页面的结构，CSS 用于布局和样式的应用。JavaScript 用于实现响应式设计，通过监听窗口大小变化来动态调整页面尺寸。同时，实现了鼠标事件监听器来捕获和响应鼠标操作，以及键盘事件监听器来处理键盘输入。自定义的\$函数也在这个阶段实现，以方便地获取 DOM 元素。实现代码如下：

测试实现的代码以确保在不同设备和屏幕尺寸上都能正确响应鼠标和触屏操作，并进行必要的调整以优化用户体验。代码块如下：

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width , initial-scale=1,
user-scalable=no">

    <title>UI 设计之鼠标模型的控制</title>

  <style>
    *{
      margin: 10px;
      text-align: center;
    }
    header{
      border: 3px solid rgb(235, 126, 211);
      height: 10%;
      font-size: 1em;
    }
    nav{
      border: 3px solid rgb(197, 132, 207);
      height: 10%;
    }
    main{
      border: 3px solid rgb(214, 194, 122);
      height: 70%;
      font-size: 0.8em;
      position: relative;
    }

    #box{
      position: absolute;
      right: 0;
```

```

        width: 100px;
    }

    footer{
        border: 3px solid green;
        height:10%;
        font-size: 0.7em;
    }
    body{
        position: relative;
    }
    button{
        font-size:1em;
    }
    #aid{
        position: absolute;
        border: 3px solid rgb(232, 178, 228);
        top:0px;
        left:600px;
    }
    #bookface{
        position: absolute;
        width: 80%;
        height: 80%;
        border:1px solid rgb(228, 187, 230);
        background-color: blanchedalmond;
        left:7% ;
        top: 7% ;
        background-image: url("0.jpg");
    }
    #bookimg{
        width: 100%;
        height:100%;
        object-fit:cover;
        margin: 0% 0%;
    }

</style>
</head>
<body >
    <header>
        <p id="book">
            《我的毕设题目》
        </p>

```

```

</header>
<nav>
  <button>向前</button>
  <button>向后</button>
  <button>其他</button>
</nav>

<main id="main">
<div id="bookface">
  
  在此对象范围拖动鼠标(本例触屏无效)
</div>
</main>

<footer>

```

俞嘉美 @ 江西科技师范大学 2025

```

</footer>
<div id="aid">
  <p>用户键盘响应区</p>
  <p id="keyboard"></p>
</div>
<script>
  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';

//尝试对鼠标设计 UI 控制
var mouse={};
mouse.isDown= false;
mouse.x= 0;
mouse.y= 0;
mouse.deltaX=0;
$("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+")";
});
$("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%';
    }

});
$("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%';
    }

});
$("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' ;
    }

});

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 $

</script>
</body>
</html>

```

6.3 项目的运行和测试

测试样例如下：

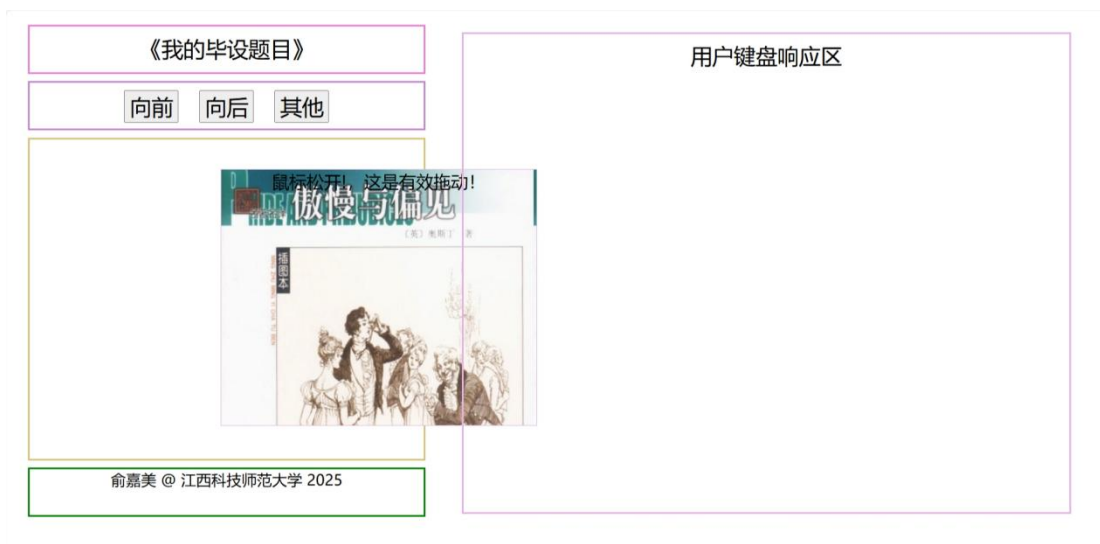


图 6-4 电脑端界面图



图 6-5 手机端界面图



图 6-6 1.4.html 二维码

6.4 项目的代码提交和版本管理

```
$ git add 1.4.html
```

```
$ git commit -m 项目第四版: ui 中鼠标模型设计的内容设计概要开发  
成功提交代码后, gitbash 的反馈如下所示:
```

```

yyy@LAPTOP-5TB3C0CJ MINGW64 /d/Git/作业 (main)
$ git commit -m 项目第四版: ui中鼠标模型设计的内容设计概要开发
[main 44806bc] 项目第四版: ui中鼠标模型设计的内容设计概要开发
1 file changed, 197 insertions(+)
create mode 100644 1.4.html

```

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：

```

yyy@LAPTOP-5TB3C0CJ MINGW64 /d/Git/作业 (main)
$ git log
commit 44806bc12beadbdc32094bef96f40f4037bb4ed9 (HEAD -> main)
Author: 俞嘉美@科师大 <1426674952@qq.com>
Date: Sun Jun 16 22:25:14 2024 +0800

```

项目第四版: ui中鼠标模型设计的内容设计概要开发

7. 对触屏和鼠标的通用交互操作的设计开发

7.1 分析和设计

在移动互联时代，用户终端的多样性要求网页设计能够适应触屏和鼠标操作。本代码为了实现这个功能，定义了一个 **Pointer** 对象用于跟踪用户的指针操作。分析用户需求和设备特性，确定需要支持鼠标和触屏事件，并实现响应式设计以适配不同屏幕尺寸。

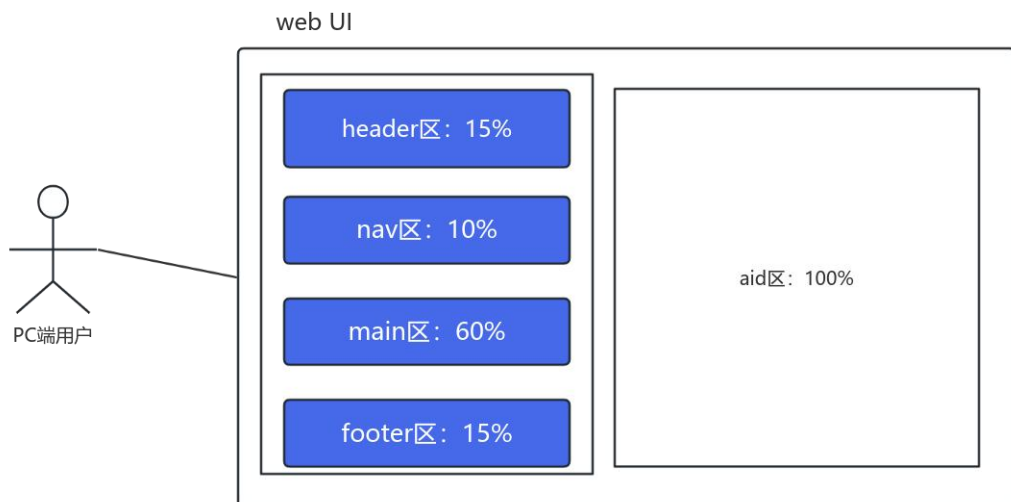


图 7-1 pc 端的用例图

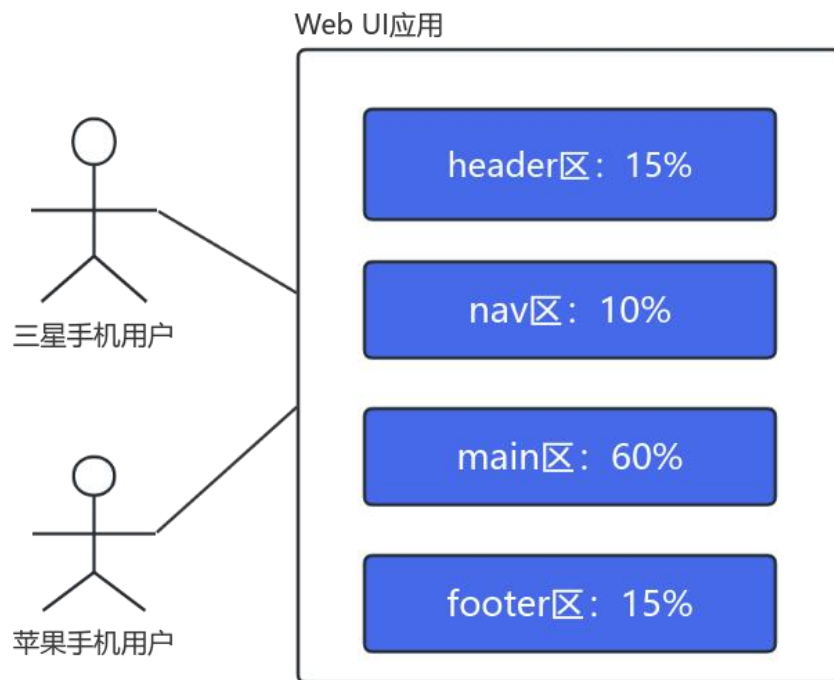


图 7-2 窄屏手机用户的用例图

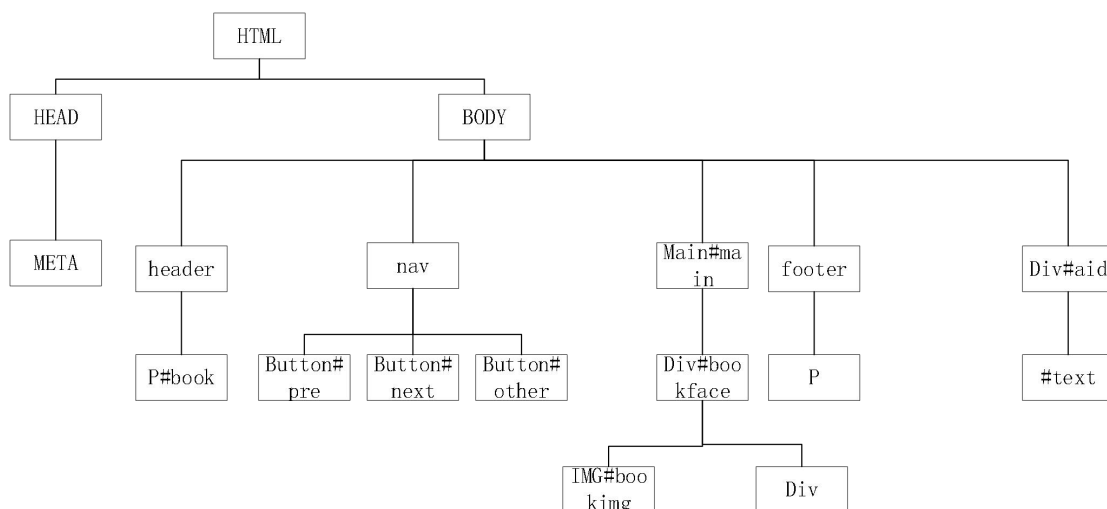


图 7-3 1.5.htmlDom 树

7.2 项目的实现和编程

代码块中定义了开始、移动和结束事件的处理函数，这些函数能够处理鼠标的 `mousedown`、`mousemove`、`mouseup` 事件，以及触屏的 `touchstart`、`touchmove`、`touchend` 事件。还为 `#bookface` 元素添加了多个事件监听器，它们会触发相应的处理函数。这些监听器能够区分鼠标和触屏事件，并更新 `Pointer` 对象的状态。当用户拖动或滑动 `#bookface` 元素超过 100 像素时，会被视为有效的 UI 互动。设计一个统一的指针事件处理模型，使用 `Pointer`

对象来跟踪和响应鼠标和触屏操作，并通过 CSS 实现响应式布局。

I: 实现 HTML 和 CSS 以构建页面结构和样式。通过 JavaScript 添加事件监听器，实现 Pointer 对象的逻辑，以统一处理 mousedown、mousemove、mouseup 和对应的触屏事件。

代码如代码块所示：

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width , initial-scale=1,
user-scalable=no">

    <title>通用的 UI 设计，用一套代码同时为触屏和鼠标建模</title>

  <style>
    *{
      margin: 10px;
      text-align: center;
    }
    header{
      border: 3px solid rgb(221, 164, 216);
      height: 10%;
      font-size: 1em;
    }
    nav{
      border: 3px solid rgb(176, 135, 179);
      height: 10%;
    }
    main{
      border: 3px solid rgb(202, 129, 190);
      height: 70%;
      font-size: 0.8em;
      position: relative;
    }

    #box{
      position: absolute;
      right: 0;
      width: 100px;
    }

    footer{
      border: 3px solid rgb(182, 129, 191);
```

```

        height:10%;
        font-size: 0.7em;
    }
    body{
        position: relative;
    }
    button{
        font-size:1em;
    }
    #aid{
        position: absolute;
        border: 3px solid rgb(200, 143, 192);
        top:0px;
        left:600px;
    }
    #bookface{
        position: absolute;
        width: 80%;
        height: 80%;
        border:1px solid rgb(216, 168, 168);
        background-color: blanchedalmond;
        left:7% ;
        top: 7% ;
        background-image: url("0.jpg");
    }
    #booking{
        width: 100%;
        height:100%;
        object-fit:cover;
        margin: 0% 0%;
    }
</style>
</head>
<body >
    <header>
        <p id="book">
            《傲慢与偏见》
        </p>
    </header>
    <nav>
        <button onclick="moveBookface(10)">向前</button>
        <button onclick="moveBookface(-10)">向后</button>
        <button>其他</button>
    </nav>

```

```

    <main id="main">
    <div id="bookface">
        
        在此对象范围拖动鼠标/滑动触屏<br>
        拖动/滑动超过 100 像素，视为有效 UI 互动！
    </div>
    </main>

    <footer>

        俞嘉美 @ 江西科技师范大学 2025

    </footer>
    <div id="aid">
        <p>用户键盘响应区</p>

    </div>
    <script>
        var UI = {};
        if(window.innerWidth>600){
            UI.appWidth=600;
        }else{
            UI.appWidth = window.innerWidth;
        }

        function moveBookface(pixels) {
            var bookface = $("bookface"); // 获取 bookface 元素
            var currentLeft = parseInt(window.getComputedStyle(bookface).getPropertyValue('left'));
            // 获取当前 left 属性值
            bookface.style.left = (currentLeft + pixels) + 'px'; // 更新 left 属性值
        }
        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 $

</script>
</body>
</html>

```

7.3 项目的运行和测试



图 7-4 pc 端界面图



图 7-5 手机端测试图



图 7-6 1.5.html 二维码

7.4 项目的代码提交和版本管理

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 1.5.html
$ git commit -m 项目第五版: 监控不同的设备的操作的概要内容设计
```

成功提交代码后, gitbash 的反馈如下所示:

```
yyy@LAPTOP-5TB3C0CJ MINGW64 /d/Git/作业 (main)
$ git commit -m 项目第五版: 监控不同的设备的操作的概要内容设计
[main d85f8de] 项目第五版: 监控不同的设备的操作的概要内容设计
1 file changed, 225 insertions(+)
create mode 100644 1.5.html
```

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:

```
yyy@LAPTOP-5TB3C0CJ MINGW64 /d/Git/作业 (main)
$ git log
commit d85f8de609b490f46706d3d92d2c82deddd75ae6 (HEAD -> main)
Author: 俞嘉美@科师大 <1426674952@qq.com>
Date: Sun Jun 16 22:29:09 2024 +0800
```

项目第五版: 监控不同的设备的操作的概要内容设计

8. UI 的个性化键盘交互控制的设计开发

8.1 分析和设计

键盘事件是 web 开发中用于处理用户键盘输入的重要机制。keydown 和 keyup 是两种常见的键盘事件类型, 它们分别在用户按下和释放键盘上的键时触发。这些事件为开发者提供了丰富的信息, 包括按键的代码、位置和状态, 从而可以创建出响应用户输入的交互式界面。键盘事件是 Web 开发中用于处理用户键盘输入的重要机制。keydown 和 keyup 是两种常见的键盘事件类型, 它们分别在用户按下和释放键盘上的键时触发。这些事件为开发者提供了丰富的信息, 包括按键的代码、位置和状态, 从而可以创建出响应用户输入的交互式界面。

项目包含了键盘事件响应功能, 用户可以通过键盘输入与页面进行交互。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, 我们在键盘响应过程中, 为了美观, 设置了输入界面, 将所有敲击的键盘的记录放置在输入框中, 同时在输入框下方放置了一个文本用于随时响应键盘的操作, 实现与键盘的交互。

我们通过定义`UI`和`Pointer`对象来封装页面布局和用户交互的状态, 从而降低了对全局变量的依赖, 使状态管理更加集中和明确。我们还考虑了如何通过事件委托减少事件监听器的数量, 以及如何使用闭包来隐藏和保护内部状态, 避免意外的全局污染。我们的目标是创建一个结构清晰、功能独立且易于扩展和维护的代码架构。

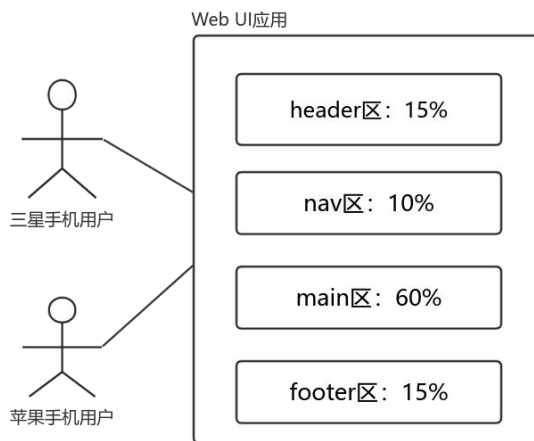


图 8-2 窄屏手机用户的用例图

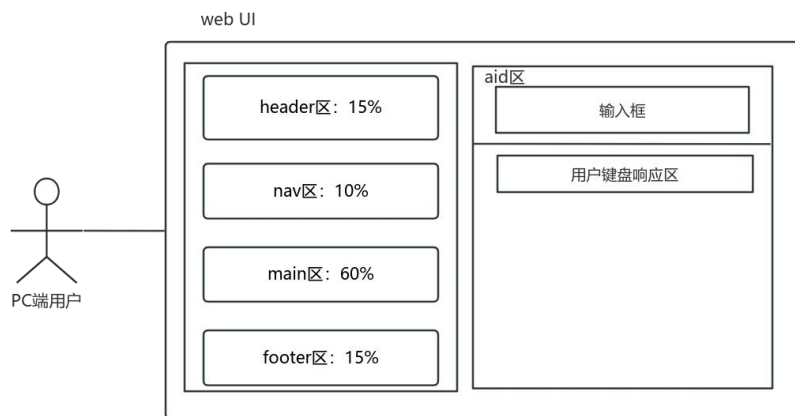


图 8-3 pc 端用户用例图

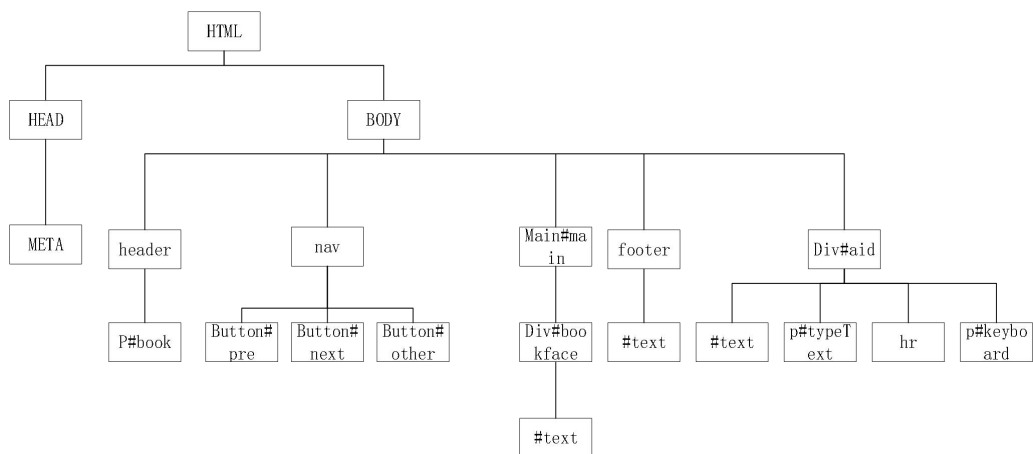


图 8-4 1.6html 的 Dom 树

8.2 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。

项目的总体代码块如下所示：

```
<!doctype html>

<html lang="en">

  <head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width , initial-scale=1">

    <meta name="viewport" content="width=device-width , initial-scale=1, user-scalable=no">

    <title>宽屏 UI 设计之底层键盘事件的研究</title>

  <style>

    *{

      margin: 10px;

      text-align: center;

      user-select:none;

    }

    header{

      border: 3px solid rgb(169, 209, 169);

      height: 10%;

      font-size: 1em;

    }

    nav{

      border: 3px solid rgb(171, 206, 203);

      height: 10%;

    }

    main{

      border: 3px solid green;
```

```
height: 70%;

font-size: 0.8em;

position: relative;
}

#box{

position: absolute;

right: 0;

width: 100px;
}

footer{

border: 3px solid green;

height: 10%;

font-size: 0.7em;
}

body{

position: relative;
}

button{

font-size: 1em;
}

#aid{

position: absolute;

border: 3px solid blue;

top: 0px;

left: 600px;
}

#bookface{

position: absolute;
```

```

width: 80%;

height: 80%;

border: 1px solid red;

color: rgba(0,0,0,0.6);

text-shadow: 2px 2px 3px rgba(255,255,255,0.5);

left: 7% ;

top: 7% ;

background-image: url("0.jpg");

}

</style>

</head>

<body >

  <header>

    <p id="book">

      《傲慢与偏见》

    </p>

  </header>

  <nav>

    <button id="pre">向前</button>

    <button id="next">向后</button>

    <button id="other">其他</button>

  </nav>

  <main id="main">

    <div id="bookface">

```

在此对象范围拖动鼠标/滑动触屏

拖动/滑动超过 100 像素，视为有效 UI 互动！

```

</main>

<footer>

    俞嘉美 @ 江西科技师范大学 2025

</footer>

<div id="aid">

    用户键盘响应区

    <p id="typeText"></p>

    <hr>

    <p id="keyboard"></p>

</div>

<script>

var UI = {};

if (window.innerWidth > 600) {

    UI.appWidth = 600;

} else {

    UI.appWidth = window.innerWidth;

}

UI.appHeight = window.innerHeight;

let baseFont = UI.appWidth / 20;

document.body.style.fontSize = baseFont + "px";

document.body.style.width = UI.appWidth - baseFont + "px";

document.body.style.height = UI.appHeight - baseFont * 5 + "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';


var images = ['0.jpg', '1.png', '2.png'];

var currentIndex = 0;


// 图片切换函数

function switchImage(newIndex) {

    if (newIndex < 0) {

        newIndex = images.length - 2; // 如果索引小于 0，则跳到最后一张图片

    } else if (newIndex >= images.length-1) {

        newIndex = 0; // 如果索引大于等于图片数量，则跳到第一张图片

    }

    currentIndex = newIndex; // 更新当前索引

    $("#bookface").style.backgroundImage = "url(" + images[currentIndex] + ")";

}


// 为按钮添加点击事件监听器

$("#pre").addEventListener("click", function() {

    switchImage(currentIndex - 1); // 点击向前按钮时，索引减 1

});


$("#next").addEventListener("click", function() {

    switchImage(currentIndex + 1); // 点击向后按钮时，索引加 1

});


$("#other").addEventListener("click", function() {

    $("#bookface").style.backgroundImage = "url(" + images[2] + ")";

```

```

});

//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);

```

略

**** 添加"keydown"和"keyup"这 2 个键盘底层事件处理后, keypress 这个高层键盘事件响应被系统忽略

```

$("body").addEventListener("keypress", function(ev){

    $("typeText").textContent += ev.key ;

});

$("body").addEventListener("keydown",function(ev){

    ev.preventDefault() ;

    let k = ev.key;

    let c = ev.keyCode;

    $("keyboard").textContent = "您已按键 : " + k + " , "+"字符编码 : " + c;

});

$("body").addEventListener("keyup",function(ev){

    ev.preventDefault() ;

    let k = ev.key;

    let c = ev.keyCode;

    $("keyboard").textContent = "松开按键 : " + k + " , "+"字符编码 : " + c;

```

```

});

*****/

//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() ;

    let k = ev.key;

    let c = ev.keyCode;

    $("keyboard").textContent = "您已按键 ： " + k + " ， "+ "字符编码 ： " + c;

});

$("body").addEventListener("keyup",function(ev){

    ev.preventDefault() ;

    let key = ev.key;

    let code = ev.keyCode;

    $("keyboard").textContent = "松开按键 ： " + key + " ， "+ "字符编码 ： " + code;

    if (printLetter(key)){

        $("typeText").textContent += key ;

    }

    function printLetter(k){

        if (k.length > 1){ //学生必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)

});

} //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 $

</script>

</body>

</html>

```

8.3 项目的运行和测试

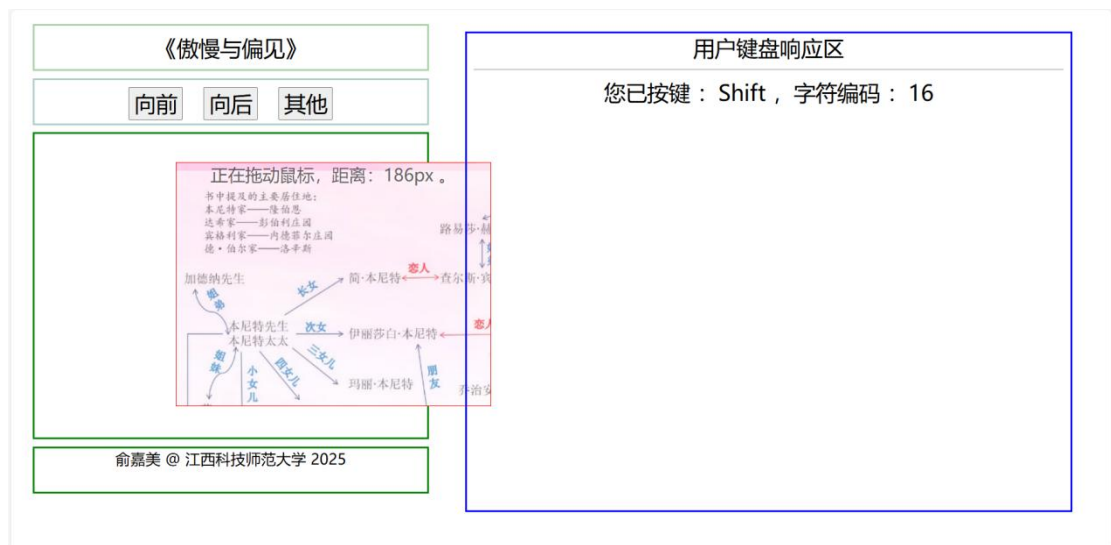


图 8-4 pc 端界面



图 8-5 手机端界面



图 8-6 1.6.html 二维码

8.4 项目的代码提交和版本管理

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 1.6.html
```

```
$ git commit -m 项目第六版：UI 的个性化键盘控制的设计开发
```

成功提交代码后，`gitbash` 的反馈如下所示：

```
yyy@LAPTOP-5TB3C0CJ MINGW64 /d/Git/作业 (main)
$ git commit -m 项目第六版：UI的个性化键盘控制的设计开发
[main 531d424] 项目第六版：UI的个性化键盘控制的设计开发
1 file changed, 295 insertions(+)
create mode 100644 1.6.html
```

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：

```
yyy@LAPTOP-5TB3C0CJ MINGW64 /d/Git/作业 (main)
$ git log
commit 531d424616c188997af08b23aa40fd12fa3a4e60 (HEAD -> main)
Author: 俞嘉美@科师大 <1426674952@qq.com>
Date: Sun Jun 16 22:32:50 2024 +0800

    项目第六版：UI的个性化键盘控制的设计开发
```

9. 谈谈本项目中的高质量代码

这是一本关于指导计算机的书。如今，电脑和螺丝刀一样常见，但它们相当复杂，让它们做你想让它们做的事情并不总是那么容易。

如果你的计算机任务是一个常见的，很容易理解的任务，比如显示你的电子邮件或像一个计算器，你可以打开适当的应用程序，并开始工作。但是对于唯一的或开放式的任务，可能没有应用程序。这就是编程可能会出现的地方。编程是构建一个程序的行为——一组精确的指令，告诉计算机要做什么。因为计算机是愚蠢的，迂腐的野兽，编程从根本来说来说是乏味和令人沮丧的。

幸运的是，如果你能克服这个事实，甚至可以享受到愚蠢的机器能够处理的严格思考，那么编程可能是值得的。它允许你在几秒钟内完成一些永远需要手工完成的事情。这是一种让你的电脑工具做一些它以前不能做的事情的方法。它提供了一个很好的抽象思维的练习^[6]。

在本项目中，高质量代码的实现遵循了面向对象编程(OOP)的原则，特别是 MVC（模型-视图-控制器）设计模式，以确保代码的可维护性和可扩展性。代码通过封装实现了对鼠标和触屏的控制，同时利用局部变量和函数式编程技术来提高代码的抽象性和逻辑性，减少了全局变量的使用。

具体来说，代码中定义了一个 `Pointer` 对象，封装了与鼠标和触屏相关的属性和方法，

如 `isDown`、`x`、`deltaX` 等，以及处理鼠标按下、移动和释放的函数。这种封装使得代码更加模块化，易于理解和维护。

此外，代码中使用了局部变量来存储临时状态，如在处理鼠标和触屏事件的代码块中，`Pointer` 对象的属性被用来记录鼠标或触屏的位置和移动距离。通过这种方式，代码避免了对全局状态的依赖，提高了代码的可读性和可测试性。

函数式编程的思想也被应用在代码中，例如通过定义 `switchImage` 函数来切换图片，以及通过 `printLetter` 函数来处理键盘输入。这些函数接受参数并返回结果，不依赖于外部状态，使得代码更加简洁和可重用。

代码中还体现了逻辑的清晰性，如在处理键盘事件时，通过 `printLetter` 函数来判断是否应该在文本区域显示按键字符，这增加了代码的逻辑性和可预测性。

最后，代码通过注释和清晰的命名规范，提高了代码的可读性，使得其他开发者能够更容易地理解和使用这段代码。

10. 用 gitBash 工具管理项目的代码仓库和 http 服务器

10.1 经典 Bash 工具介绍

当我们谈到命令行时，我们实际上指的是 `shell`。`shell` 是一个程序，并将它们传递给操作系统执行。几乎所有的 Linux 发行版都提供了一个来自 GNU 项目的 `shell` 程序。这个名字是伯恩-再次外壳的首字母缩写，指的是 `bash` 是 `sh` 的增强替代品，最初的 Unix 外壳程序由史蒂夫·伯恩写。^[7] 和 Windows 一样，类似 `inix` 的 Linux 操作系统以分层目录结构组织文件。这意味着它们被组织成一个树状的目录模式（在其他系统中有时称为文件夹），其中可能包含文件和其他目录。文件系统中的第一个目录称为根目录。根目录包含文件和子目录，其中包含更多的文件和子目录，等等。^[7]

10.2 通过 gitHub 平台实现本项目的全球域名


10.3 创建一个空的远程代码仓库

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 *

 masterLijh

Repository name *

 userName.github.io is available.

Great repository names are short and memorable. Need inspiration? How about [expert-rotary-phone](#) ?

Create repository

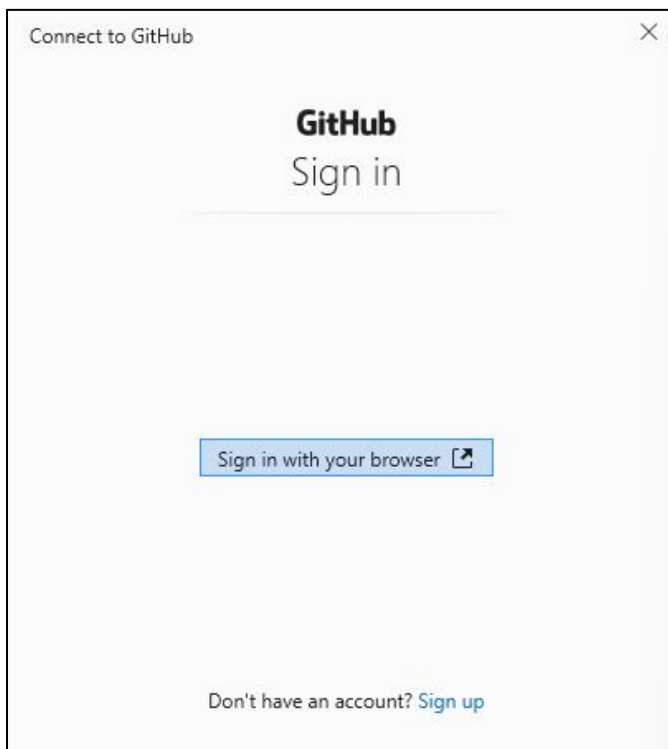
点击窗口右下角的绿色“Create repository”，则可创建一个空的远程代码仓库。

10.4 设置本地仓库和远程代码仓库的连接

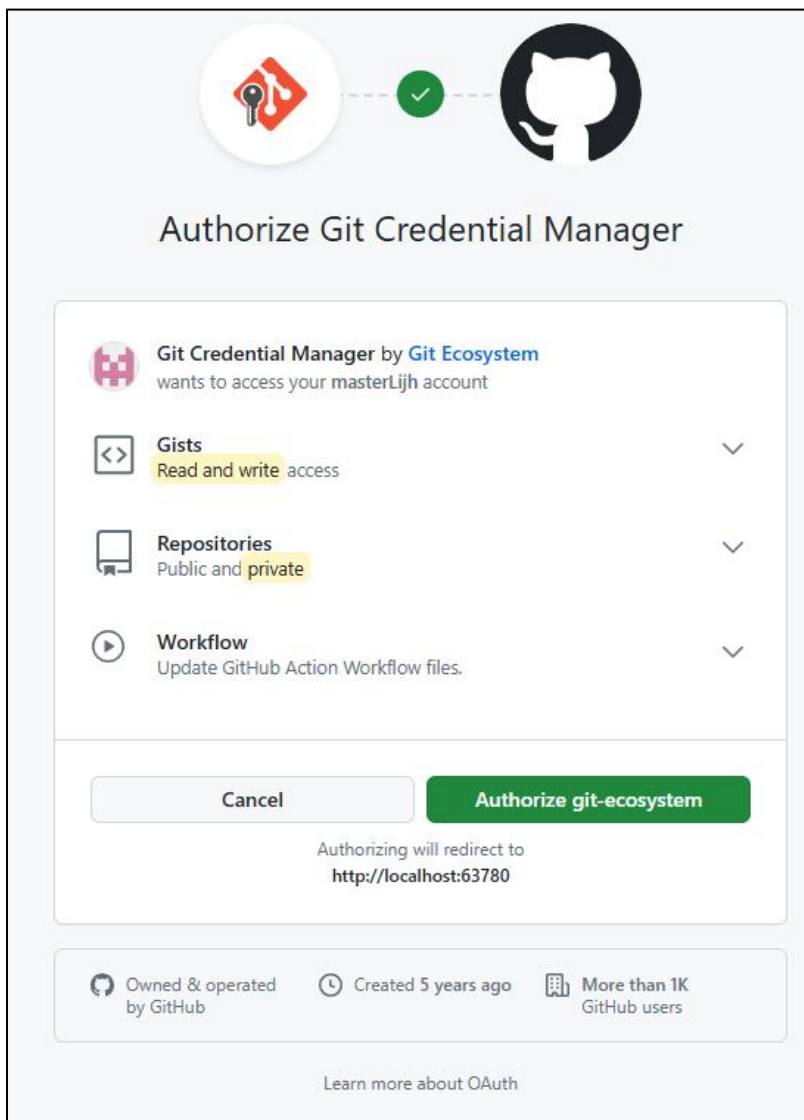
进入本地 webUI 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接

```
$ 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/wkqmzzn/wkqmzzn.github.io.git
$ git push -u origin main
```

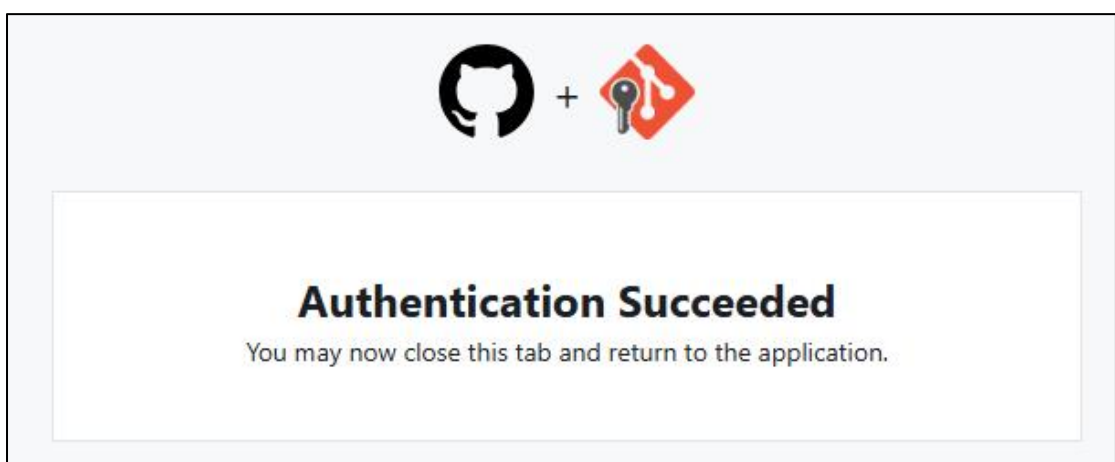
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



再次确认授权 gitBash 拥有访问改动远程代码的权限，如下图所示：



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



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

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

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

写作指导:

实质上任何章节都可以按三段论模式写，第一段是写研究的背景和目标，第二段是写你所开展的工作步骤和工作量，第三段是用了哪些方法和工作的结果或意义。

【摘要案例 1】近十年来，html5 为核心的 web 标准的软件开发技术以其跨平台、开源的优势广泛地运用在各个领域的应用软件开发中。通过分析本次毕设任务，本项目选择 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书籍、开发者论坛和文献，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 语言进行内容建模、css 语言展开 UI 的外观设计、javascript 语言编程实现 UI 的交互功能，除直接使用了 web 客户端最底层的 API 外，本项目的每条代码都是手工逐条编写，没有导入他人的任何的代码（框架和库）。本项目也采用了响应式设计编程，可以智能地适应移动互联网时代用户屏幕多样化的需要；另外大量地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，实现了高质量的代码，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（A:Analysis, D:Design, I: Implementation, T:Testing），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，在漫长的开发过程中重构代码六次并正式做了代码提交，另外在测试中修改提交了代码两次，最后利用 gitbash 工具 把本项目的代码仓库上传到著名的 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。

【摘要案例 2】：Web 技术以其跨操作系统平台的优势成为了广泛流行的软件开发手段，为了适应移动互联网时代软件项目的前端需求，本项目以 Web 客户端技术为研究学习内容，广泛查阅了技术资料与相关文献，尤其是 mozilla 组织的 MDN 社区的技术实践文章，探索了 HTML 内容建模、CSS 样式设计和 JavaScript 功能编程的基本技术和技巧。通过集成上述技术，再应用本科的相关课程的知识，实现了一个个性化的用户界面（UI: uer interface）的项目，该用户界面以响应式技术为支撑做到了最佳适配用户屏幕，程序可以动态适用于当前 PC 端和移动设备；在功能上以 DOM 技术和事件驱动模式的程序为支撑实现了对鼠标、触屏、键盘的底层事件响应和流畅支持，为鼠标和触屏设计了一个对象模型，用代码实现了对这类指向性设备的模拟（这是本项目模型研究法的一次创新实践，也是本项目的亮点。）。为了处理好设计和开发的关系，项目用了工程思想管理，使用了软件工程的增量式开发模式，共做了 6 次项目迭代开发，每次迭代都经历了开发 4 个经典开发阶段（A:Analysis, D:Design, I: Implementation, T:Testing），以逐步求精的方式编写了本 UI 的应用程序。为了分享和共享本代码，与网上的开发者共同合作，本项目还使用了 git 工具进行代码和开发过程日志记录，一共做了 12 次提交代码的操作，详细记录和展现了开发思路和代码优化的过程，最后通过 gitbash 把项目上传到 github 上，建立了自己的代码仓库，并将该代码仓库设置成为了 http 服务器，实现了本 UI 应用的全球便捷访问。