

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）：Design and implementation of Personalized UI 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：周志能

学号：20213658

指导教师：李健宏

2024 年 6 月 1 日

目录

第一章前言	1
1.1 项目概要	1
1.2 研学计划	1
1.3 研究方法	2
第二章技术总结和文献综述	2
2.1 Web 平台和客户端技术概述	2
2.2 项目的增量式迭代开发模式	3
第三章内容设计概要	4
3.1 分析和设计	4
3.2 项目的实现和编程	5
3.3 项目的运行和测试	7
3.4 项目的代码提交和版本管理	7
第四章移动互联时代的 UI 开发初步——窄屏终端的响应式设计	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	12
4.4 项目的代码提交和版本管理	13
第五章应用响应式设计技术开发可适配窄屏和宽屏的 UI	1
5.1 分析与设计	1
5.2 项目的实现和编程	2
5.3 项目的运行和测试	6
5.4 项目的代码提交和版本管理	7
第六章个性化 UI 设计中对鼠标交互的设计开发	8
6.1 分析与设计	8
6.2 项目的实现和编程	9
6.3 项目的运行和测试	16
6.4 项目的代码提交和版本管理	17
第七章对触屏和鼠标的通用交互操作的设计开发	18
7.1 分析和设计	18

7.2 项目的实现和编程	19
7.3 项目的运行和测试	22
7.4 项目的代码提交和版本管理	23
第八章 UI 的个性化键盘交互控制的设计开发	24
8.1 分析与设计	24
8.2 项目的实现和编程	25
8.3 项目的运行和测试	30
8.4 项目的代码提交和版本管理	31
第九章 谈谈本项目中的高质量代码	32
9.1 编程的作用	32
9.2 项目的高质量代码	32
第十章 用 gitBash 工具管理项目的代码仓库和 http 服务器	34
10.1 经典 Bash 工具介绍	34
10.2 通过 GitHub 平台实现本项目的全球域名	34
10.2.1 创建一个空的远程代码仓库	34
10.3 设置本地仓库和远程代码仓库的链接	35
参考文献	38

摘要：近十年间，以 HTML5 为核心的 web 标准软件开发技术凭借其跨平台以及开源的特性，在各个领域的软件开发中得到了广泛运用。经由对本次课程任务的剖析，本项目将 HTML5 的 web 客户端技术选定为技术路线，进而对程序设计与软件开发展开研究及实践，设计并开发出了一款具有个性化的用户界面（UI）应用程序。在开发过程中，运用 HTML 进行内容建模，通过 CSS 展开 UI 的外观设计，利用 JavaScript 编程实现 UI 的交互功能。从工程管理层面来看，本项目采用了增量式开发模式，以逐步求精的方式进行了六次代码的增量式项目迭代（A: Analysis, D: Design, I: Implementation, T: Testing），较为顺利地完成了项目的设计开发与测试。最终借助 gitBash 将项目代码上传至 Github，达成了本 UI 在全球互联网的部署，用户能够通过 URL 和二维码便捷且高效地跨平台访问该程序。

关键词：html5；个性化 UI；增量式开发模式；

第一章 前言

1.1 项目概要

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，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，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 展开 UI 的外观设计、javascript 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，来实现了一个个性化的用户界面（UI: userinterface）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 端和移动端设备；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）。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，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。

1.2 研学计划

本项目计划设定六个阶段进行迭代递增，并通过使用 gitbash 工具进行项目提交，并上传到 github 仓库上进行代码的存储和管理，同时该仓库以开源式的方式与全球开发人员进行项目的共享。这六个阶段从内容设计到移动互联的实现，再到响应式 UI 的设计实现层层开发，第一阶段的内容设计主要是使用 html 技术加 css 技术实现简单的内容展示，第二阶段是基于第一阶段的基础，使用了 css 更为高级的语法和 JavaScript 技术使得内容的展示更为饱满、同时通过添加导航区便于精准定位内容的具体展示，第三阶段的实现则是在第二阶段的内容上开始实现移动互联的响应式 UI 设计，通过键鼠的响应的展示 pc 端和移动端的个性化 UI 设计，同时针对市面上主要的两大移动端系统 ios 和

android，解决不同系统的内容展示问题，在接下了的三个阶段则都是针对个性化 UI 设计的键鼠响应和移动互联的时代的宽窄屏的优化，去完善与实现更多的个性化 UI 设计。最后通过 http 服务来实现全球的 UI 应用的访问。

1.3 研究方法

首先，通过相关的 Web 客户端技术在个性化 UI 设计和实现的文献的研究，确定用户对移动互联时代的个性化 UI 设计的需求和期望，研究分析项目的可行性，并通过使用软件工程中常用的开发方法中的经典模型设计：迭代增量式开发模型，它将整个软件开发过程分为多个小步骤（迭代），每个迭代都会产生一个可执行的部分（增量），并且在后续的迭代中逐步完善和扩展这个部分。基于定性和定量研究所得数据，进行统计分析和数据挖掘，根据其中所得的相应数据，通过使用迭代增量式开发模型来逐步实现移动互联时代中个性化 UI 设计，最后总结归纳 Web 客户端技术中个性化 UI 设计和实现的关键问题和未来发展方向，探讨可能的发展趋势和创新方向，为相关研究和实践提供启示和指导。

第二章 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”蒂姆·伯纳斯-李在构建出 Web 的基本技术架构后，便创立了 W3C 组织。该组织于 2010 年后推出的 HTML5 国际标准，与欧洲 ECMA 组织维护的 ECMAScript 国际标准相结合，近乎完美地铸就了全球开发者实现开发平台统一的愿景。时至今日，科学家与 Web 行业仍始终致力于完善这一伟大且光荣的理想^[1]。而学习 Web 标准与 Web 技术，学习编写 Web 程序以及运用相关工具，最终构建一套能跨平台运行的高质量代码应用，这便是我毕业设计项目所采用的技术路线。

1989 年，为助力 CERN（位于瑞士日内瓦的欧洲粒子物理实验室）的合作研究，蒂姆·伯纳斯-李提出了在研究论文中添加“超文本链接”这一构想，如此一来，当一篇论文引用另一篇论文时，读者便能点击链接并迅速跳转至另一篇论文。从 1989 年至 1991 年，伯纳斯-李成果斐然：（1）他设计了 HTML，将超文本链接作为关键特性；（2）他规划了万维网背后的理念，涵盖 HTTP 协议；（3）他创建了一个运用 HTML 网页浏览互联网的浏览器原型。1993 年，蒂姆·伯纳斯-李和丹康·诺利向互联网工程任务组（IETF）提交了首个关于 HTML 的正式提案。1994 年，蒂姆·伯纳斯-李于麻省理工学院创建了万维网联盟（W3C），并接手了 HTML 标准的管理工作。他编写了“超文本标记语言”（HTML）的首个版本，这种具备超文本链接功能的文档格式语言成为了 Web 的主要发布格式。伴随 Web 技术的传播，他对 uri、HTTP 以及 HTML 的初始规范进行了改进与探讨。

万维网联盟（WorldWideWebConsortium，简称 W3C）乃是由万维网的发明者蒂姆·伯纳斯-李于 1994 年 10 月在美国麻省理工学院计算机科学实验室（MIT/LCS）创立，且得到了欧洲核子研究中心、DARPA（美国国防高级研究计划局）以及欧盟委员会的支持。W3C 的创立标志着万维网迈入正式规范化发展阶段，旨在推动网络技术的统一与标准化，确保网络信息具备普遍可访问性与兼容性。

起初，W3C 主要聚焦于 HTML 等基础网页技术标准的制定，随着互联网的迅猛发展，其工作范畴逐渐拓展至 CSS（层叠样式表）、XML（可扩展标记语言）、DOM（文档对象模型）、SVG（可缩放矢量图形）、WebAccessibility（网页无障碍）、WebServices（网络服务）以及后来的 HTML5 等广泛领域。W3C 至今已发布了数百项影响深远的 Web 技术标准及实施指南，极大地促进了互联网技术

的进步与应用的普及。

随着时间的推移，W3C 的成员构成愈发国际化，从最初的几家创始机构拓展至涵盖全球 40 多个国家的 400 余个会员组织，并在全球多地设立了办事处，彰显了其作为国际性标准组织的广泛影响力与中立性。W3C 通过其开放的标准制定流程，鼓励多方参与，以保障网络技术的持续创新与健康发展。

下面先简要介绍一下 Web，即万维网的简称。多数人会说“Web”而非“WorldWideWeb”，我们也遵循这一惯例。网络是文档的集合，被称作网页，由世界各地的计算机用户所共享（大部分）。不同类型的网页承担着不同的功能，但至少，它们都会在电脑屏幕上显示内容。所谓“内容”，我们指的是文本、图片以及用户输入机制，如文本框和按钮等，我们可以运用不同的技术来处理对应的“内容”，例如：HTML（超文本标记语言）：用于明确网页内容的结构与意义；CSS（层叠样式表）：用于描绘网页的外观与布局；JavaScript：一门编程语言，用于实现网页的动态功能与交互性；DOM（文档对象模型）：一个编程接口，使得 JavaScript 能够操作网页的内容、结构与样式；WebAPIs：一系列由浏览器提供的 API，如 FetchAPI 用于数据获取、WebStorage 用于客户端存储、WebWorkers 用于后台处理等，这些极大地丰富了 Web 应用的功能^[2]。

Web 编程是一个广阔的领域，不同类型的 Web 编程由不同的工具来实现。所有工具都运用核心语言 HTML，所以几乎所有的 Web 编程书籍都会在一定程度上对 HTML 进行描述^[3]。每本教科书都会全面涵盖 HTML5、CSS 和 JavaScript。这三种技术被视为客户端 Web 编程的支柱。通过客户端 Web 编程，所有的网页计算都在最终用户的计算机（客户端计算机）上执行。

2.2. 项目的增量式迭代开发模式

本项目乃是本科专业学生课程设计的一个软件作品，其相较于单一用途的程序要复杂许多，本项目所包含的手写代码量远超简单的一两个数量级之上，从分析问题到初步着手写代码并非短短几天就能够实现的，完全可以称本项目为一个系统工程，故而需要以软件工程的管理视角来对待并规范该项目的编写流程。

本项目在软件工程开发过程管理模式上考虑选取两种经典模型：即瀑布模型(The waterfall model) 和增量式迭代模型(The incremental model)^[4]。而任何一种开发模式都必定要历经以下四个阶段：分析（Analysis）、设计（Design）、实施（Implementation）、测试（test）。

瀑布模型给项目提供了依阶段划分的检查点，突出开发的阶段性，且每个阶段仅执行一次，这就需要专业团队的完美协作，对于普通开发者而言不太实际，身为小微开发者难以一次性完美地完成任一阶段的工作，例如在实施过程中发觉设计阶段存在问题，那就必须在下一次迭代项目时改进设计^[5]。而在增量模型中，如下图 2.1 所示。

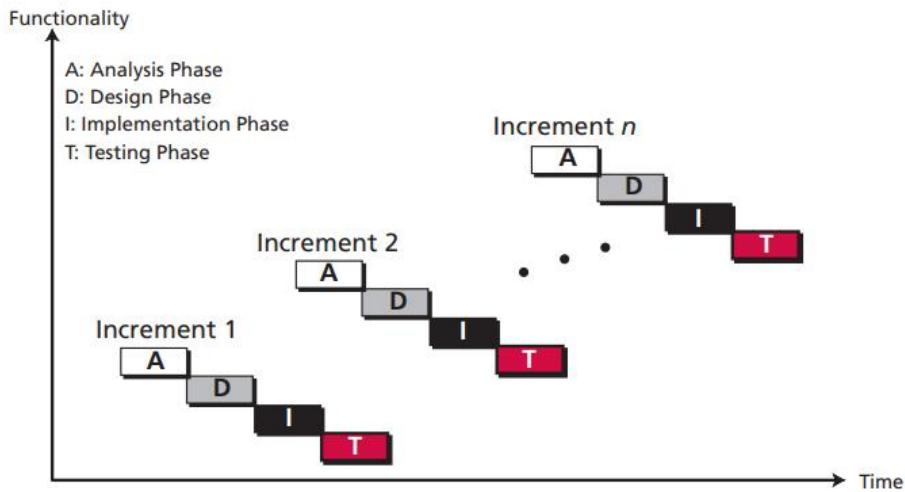


图 2.1 增量迭代开发模式

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，而一些没有完成的，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，他们不会添加更多的功能。这个过程一直持续到添加所有所需的功能。

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提高代码质量。因此在本项目的开发中，采用了增量模型的开发模式。本项目历经六次迭代最终完成。

第三章 内容设计概要

3.1. 分析和设计

这一步是项目的初次开发，本项目最初使用人们习惯的“三段论”式简洁方式开展内容设计，首先用一个标题性信息展示 logo 或文字标题，吸引用户的注意力，迅速表达主题；然后展现主要区域，也就是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的重点；最后则是足部的附加信息，用来显示一些用户可能关心的细节变化。如图 3.1 用例图所示：

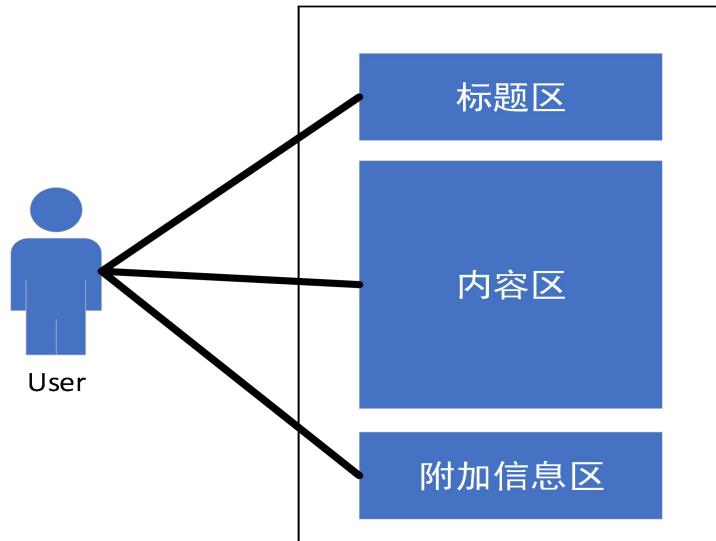


图 3.1 用例图

在设计方面，我画了一个 DOM 树，直观的表达代码结构，DOM 树如下图 3.2 所示：

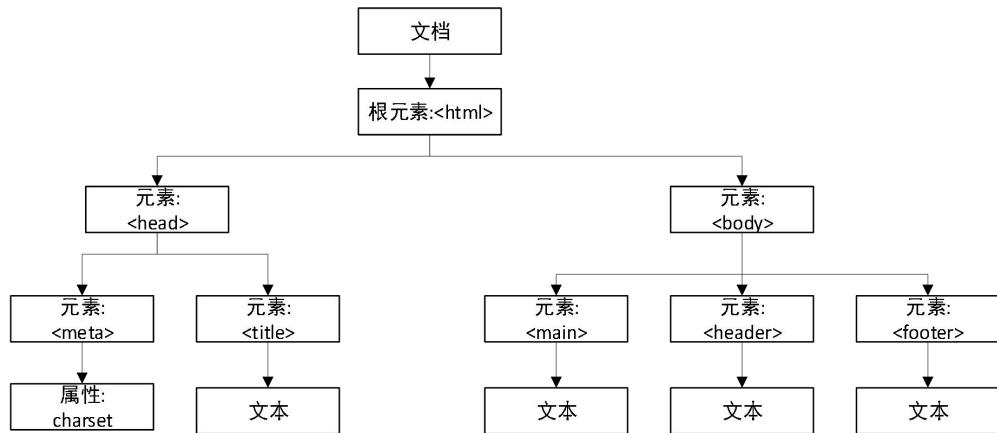


图 3.2 DOM 树 1.1

3.2 项目的实现和编程

本项目第一阶段以“三段论”方式展开的内容代码在第一部分标题区中使用 HTML+CSS 来达到项目的实现，因为这是项目的初始阶段，我们使用简单的语句来初步介绍我们想要表达的内容，同时作为移动互联设计的第一阶段，我们将在 pc 端和移动端同步实现项目的初始展示，其次在项目的代码提交和版本管理上，我们利用 gitBash 工具进行项目的管理。

HTML 代码编写如代码块 3-1：

```

<header>
  《WebAppUI 设计开发》
</header>
  
```

```
<main>

</main>

<footer>
    CopyRight 周志能江西科技师范大学 2024-2025
</footer>
```

代码块 3-1

CSS 代码编写如代码块 3-2:

```
*{
margin:10px;
text-align:center;
font-size:30px;
}

header{
border:2px solid blue;
height:200px;
}

main{
border:2px solid blue;
height:400px;
}

footer{
border:2px solid blue;
height:100px;
}

a{
display:inline-block;
padding:10px;
```

```
color:white;  
background-color:blue;  
text-decoration:none;  
}
```

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 Chrome 浏览器打开项目的结果，如下图 3.3 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 3.4 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 3. 3PC 端运行效果图



图 3. 4 移动端二维码

3.4 项目的代码提交和版本管理

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

```
$cd/  
$mkdir webUI  
$cd webUI  
$git init  
$git config user.name eyk
```

```
$gitconfig user.email 2031552294@qq.com  
$touch index.html myCss.css myjs.js
```

编写好 index.html 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 第一次提交，我们完成了软件的设计概要，完成了三个部分。header 部分放了软件的标题，main 部分放了软件的内容，footer 放了软件的动态反馈。
```

成功提交代码后，gitbash 的反馈如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)  
$ git commit -m 第一次提交，我们完成了软件的设计概要，完成了三个部分。header 部分放了软件的标题，main 部分放了软件的内容，footer 放了软件的动态反馈。  
[master (root-commit) 7178843] 第一次提交，我们完成了软件的设计概要，完成了三个部分。header 部分放了软件的标题，main 部分放了软件的内容，footer 放了软件的动态反馈。  
 3 files changed, 0 insertions(+), 0 deletions(-)  
 create mode 100644 index.html  
 create mode 100644 myCss.css  
 create mode 100644 myjs.js
```

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：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)  
$ git log  
commit 7178843965c52570dafb7821401fbc11b3ebc077 (HEAD -> master)  
Author: 江科大周志能 <2031552294@qq.com>  
Date:   Wed Jun 12 15:02:39 2024 +0800
```

```
第一次提交，我们完成了软件的设计概要，完成了三个部分。header 部分放了软件的标题，main 部分放了软件的内容，footer 放了软件的动态反馈。
```

第四章 移动互联时代的 UI 开发初步——窄屏终端的响应式设计

4.1 分析和设计

因为在计算机上使用的显示器硬件差别很大，所以每个品牌或者不同类型的显示器的大小和分辨率都不同。因此设计师并没有选择网页的版本作为具体布局，而是选择让网页给出总体布局指南，并允许浏览器选择如何在给定的计算机上显示页面。例如，一个网页的作者可以指定一组句子组成一个段落，但作者不能指定细节，如一行的确切长度或是否缩进段落的开头。但是如果允许一个浏览器选择网页布局的细节可能会出现一个有趣的结果：当通过两个浏览器或在两个硬件不同的计算机上查看时，一个网页可能会出现不同的外观。如果一个屏幕比另一个宽，一行文本的长度或可以

显示的图像的大小就不同。重点是：网页给出了关于所需演示文稿的一般指南；浏览器在显示页面时选择详细信息。因此，当同一网页在两台不同的计算机上显示或通过显示不同时，可能会出现略有不同。所以，在第二阶段的设计中，我们以第一阶段为基础，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，调节当前页面的宽窄，以达到对设备的完美适应，同时这种设置对于 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是各种品牌的手机以及电脑，都能使其在页面种呈现较为合理及理想的状态。用例图如图 4.1 所示：

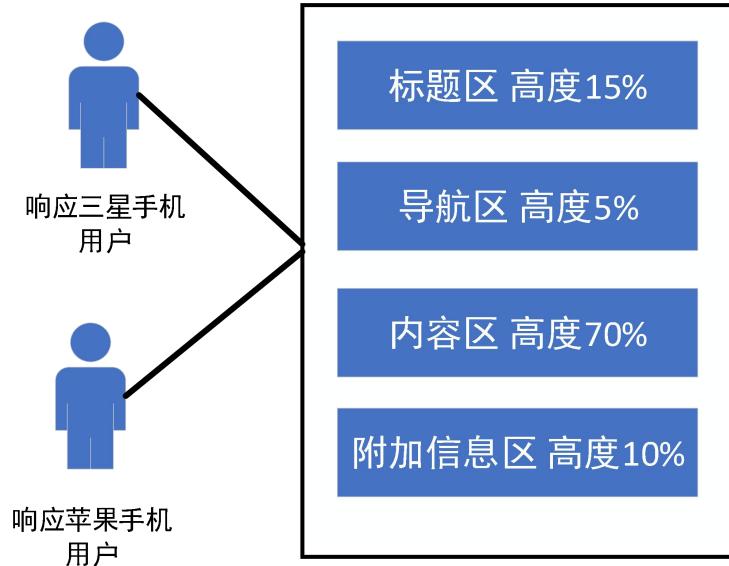


图 4.1 不同手机用户的用例图

我画了一个 DOM 树，直观的表达代码结构，DOM 树如下图 4.2 所示：

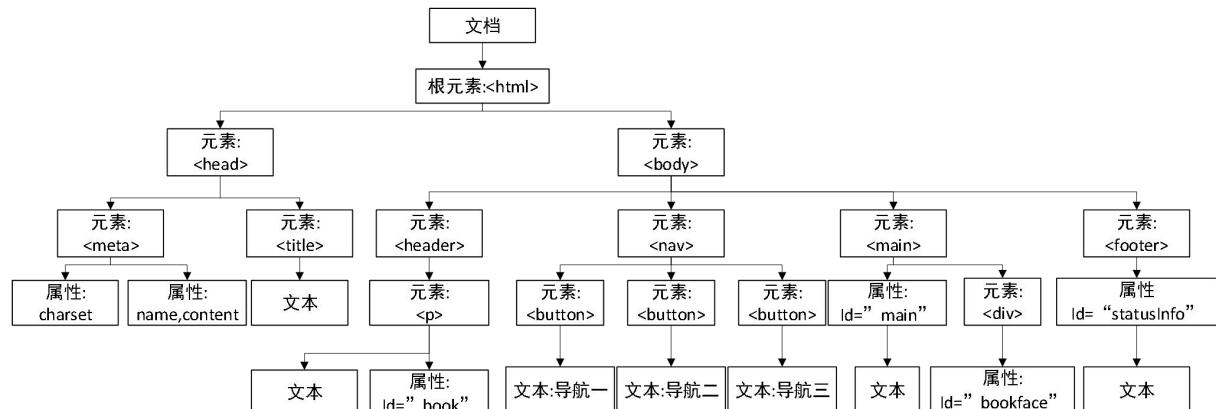


图 4.2 DOM 树 1.2

4.2 项目的实现和编程

HTML 代码编写如代码块 4-1：

```
<header>
  <pid="book">
```

WebAppUI 设计开发

```
</p>
</header>
<nav>
  <button>导航一</button>
  <button>导航二</button>
  <button>导航三</button>
</nav>
<main id='main'>

</main>
<footer>
  <p id="statusInfo">
    周志能江西科技师范大学 2024-2025
    </p>
</footer>
```

代码块 4-1

用汉语言来描述我们是如何实现的，与上一阶段比较，本阶段初次引入了 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 css 代码块 4-2 所示：

```
*{
  margin:10px;
  text-align:center;
  font-size:30px;
}

header{
  border:2px solid blue;
  height:200px;
}

main{
```

```

border:2pxsolidblue;
height:400px;
}

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

```

代码块 4-2

用汉语言来描述我们是如何实现的：与上一阶段比较，本阶段首次使用了 JavaScript，首先创建了一个 UI 对象，然后把系统的宽度和高度记录在 UI 对象中，又计算了默认字体的大小，最后再利用动态 CSS，实现了软件界面的全屏设置^[6-7]。如 js 代码块 4-3 所示：

```

<script>

varUI={};

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

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

```

```
document.body.style.height=UI.appHeight-4*baseFont+"px";  
</script>
```

代码块 4-3

4.3 项目的运行和测试

本次阶段的项目运行和测试是基于在 MicrosoftEdge 浏览器中去运行并测试在两个不同移动端设备的显示，以确保可以有效地满足移动互联时代的响应式设计的需求，如图 4.3 三星手机移动端展示图和图 4.4 苹果手机移动端展示图如下图所示：

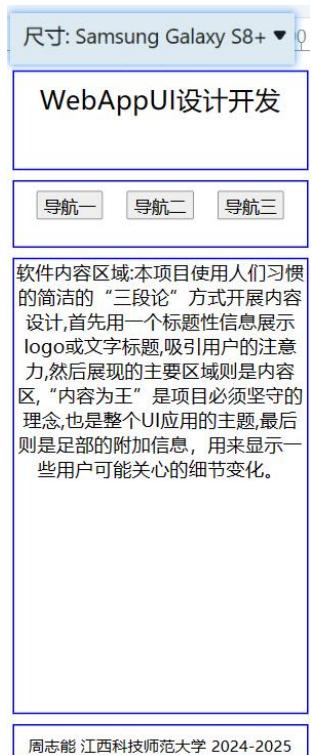


图 4.3 三星手机端用户测试图

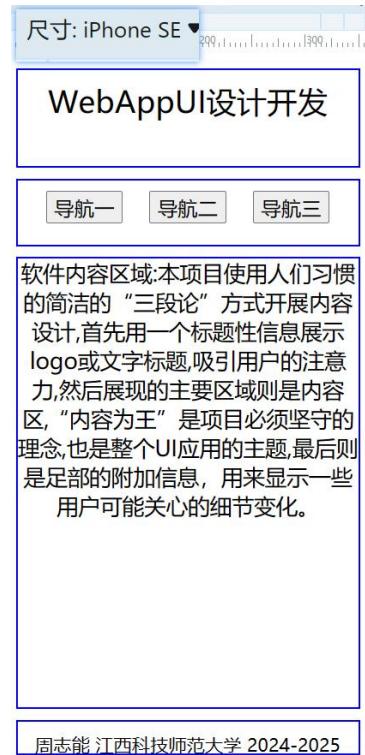


图 4.4 苹果手机端用户测试图

移动端用户可以通过扫描图 4.5 的二维码，运行测试本项目的第二次开发的阶段性效果。



图 4.5 移动端二维码

4.4 项目的代码提交和版本管理

再次编写好 index.html, myCss.cssm,myjs.js 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.cssm myjs.js
```

\$git commit -m 第二次提交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并且还能使不同的设备更好地显示内容。

成功提交代码后，gitbash 的反馈如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git commit -m 第二次提交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并且还能使不同的设备更好地显示内容。
[master 47aa838] 第二次提交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并且还能使不同的设备更好地显示内容。
 3 files changed, 18 insertions(+)
```

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：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git log
commit 47aa83888611fa6783484408e3a4fc8fbeec9b2a (HEAD -> master)
Author: 江科大周志能 <2031552294@qq.com>
Date:   Wed Jun 12 15:05:11 2024 +0800
```

第二次提交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并且还能使不同的设备更好地显示内容。

5.1 分析与设计

移动互联时代的用户终端多样性体现在不同设备类型、屏幕尺寸、分辨率、操作系统和浏览器等方面。用户可能使用智能手机、平板电脑、笔记本电脑、台式电脑以及甚至智能手表等设备来访问网站或应用程序，并且在此次的项目更新中还初步添加了鼠标交互和键盘交互。为了确保用户在不同设备上都能够获得良好的用户体验，响应式设计成为了一种必要的方法。以下是响应式设计的分析与设计。

先根据当前窗口的宽度来确定应用的宽度，如果窗口宽度大于 600 像素，则将应用宽度设置为 600 像素，否则保持窗口宽度不变。然后获取当前窗口的高度，并计算基础字体大小。

然后通过 JavaScript 来操作页面的 CSS 样式，将页面的字体大小设置为基础字体大小，宽度设置为应用宽度减去两倍的基础字体大小，高度设置为应用高度减去 8 倍的基础字体大小。

如图 5.1 用例图所示：

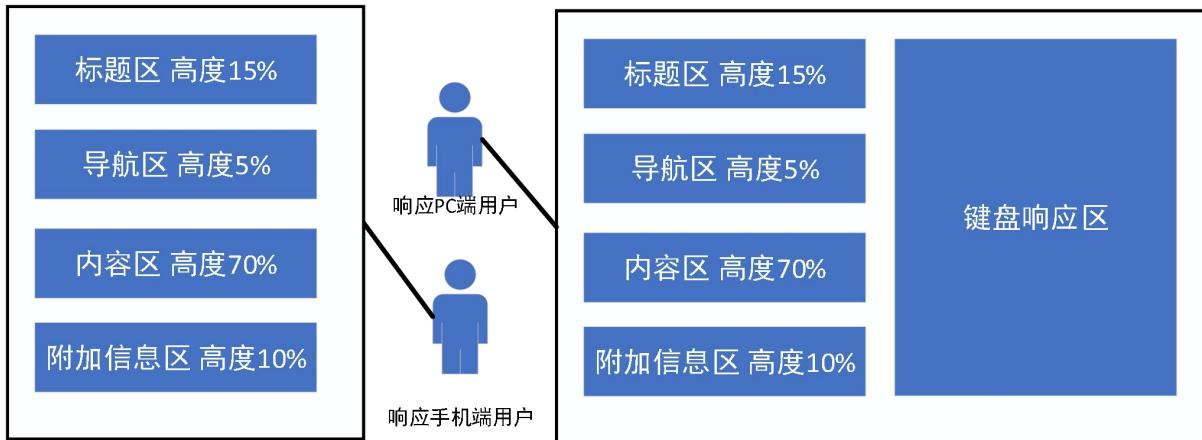


图 5.1 用例图

DOM 树如下图 5.2 所示：

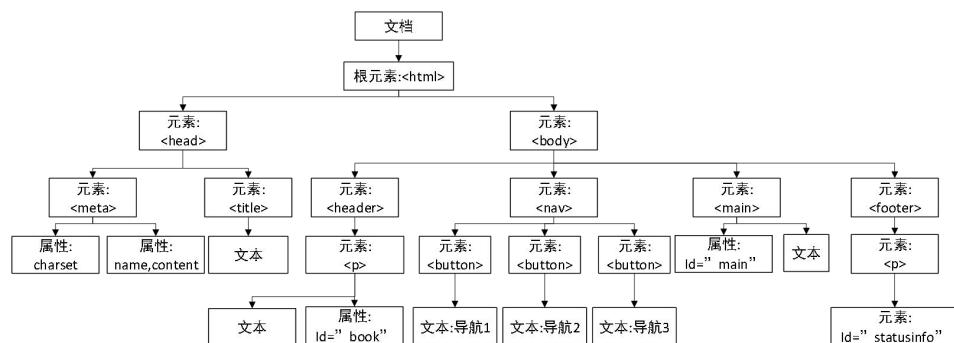


图 5.2 DOM 树 1.3

第五章 应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.2 项目的实现和编程

HTML 代码块如下代码块 5-1 所示:

```
<header>
<p id="book">
    《WebAppUI 设计开发》
</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">
        书的封面图
    </div>
</main>

<footer>
    CopyRight 周志能江西科技师范大学 2024--2025
</footer>
```

代码块 5-1

Css 代码块下代码块 5-2 所示:

```
*{
    text-align:center;
    box-sizing:border-box;
}
```

```
header,main,div#bookface,nav,footer{
    margin:1em;
}

header{
border:2pxsolidblue;
height:15%;
font-size:1.0em;

}

main{
border:2pxsolidblue;
height:70%;
font-size:1.2em;

}

nav{
border:2pxsolidblue;
height:10%;
}

navbutton{
    font-size:1.1em;
}

footer{
border:2pxsolidblue;
height:10%;
}

body{
    position:relative;
}

#aid{
```

```

position: absolute;
border: 3px solid blue;
top: 0.5em;
left: 600px;
}

#bookface{
width: 80%;
height: 80%;
border: 1px solid red;
background-color: blanchedalmond;
margin: auto;
background-image: url("../imgs/CSS.jpg");
background-size: cover;
}

```

代码块 5-2

与上一阶段比较，本阶段使用了 JavaScript，最新创建了一个 `mouse` 对象以及 `keypress` 对象，可以初步地接收鼠标按下地坐标以及接收用户键盘按下的按键及其对应的编码，最后再利用动态 CSS，实现了软件界面的全屏设置，如 js 代码块 5-3 所示：

```

var UI={};

UI.appWidth=window.innerWidth>600?600:window.innerWidth;
UI.appHeight=window.innerHeight;

const LETTERS=22;

const baseFont=UI.appWidth/LETTERS;

//通过更改 body 对象的字体大小，这个属性能够遗传其子子孙孙
document.body.style.fontSize=baseFont+"px";
//通过把 body 对象的宽度和高度设置为设备/屏幕的宽度和高度，实现全屏。
//通过 CSS 对子对象百分比（纵向）的配合，从而实现响应式设计的目标。
document.body.style.width=UI.appWidth-2*baseFont+"px";
document.body.style.height=UI.appHeight-8*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" ).addEventListerner( "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$函数未能在页面上获取任何元素，请自查问题！");  
return;  
}  
}  
}  
}//endof$
```

代码块 5-3

5.3 项目的运行和测试

此次测试主要针对 PC 端设备和手机端进行鼠标交互以及键盘交互的功能性测试，但由于手机端屏幕大小没有超过 600，所以无法显示出手机端的键盘交互区页面，PC 端如下图 5.3 所示，手机端如下图 5.4 所示：

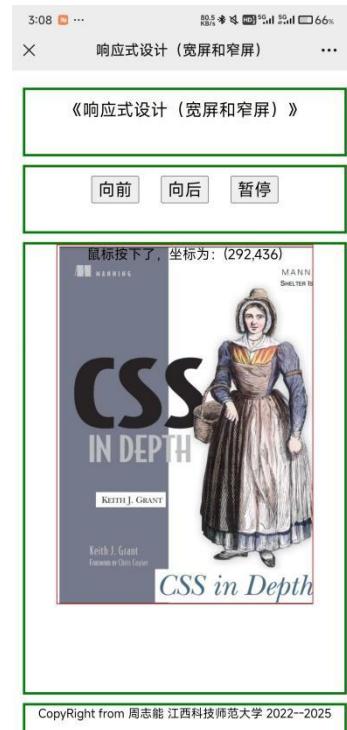


图 5.3PC 端键鼠功能测试图

图 5.4 手机端键鼠功能测试图

移动端用户可以通过扫描图 5.5 的二维码，运行测试本项目的第三次开发的阶段性效果。



图 5.5 移动端二维码

5.4 项目的代码提交和版本管理

再次编写好 index.html, myCss.cssm,myjs.js 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.cssm myjs.js
```

\$git commit -m 第三次提交，我们增加了一个鼠标模型，可以判断鼠标的移动和坐标，并显示了鼠标坐标，并在该页面使用了教材的背景图片。还增加了一个键盘模型，可以识别键盘按下的字母以

及 ASCII 码的编号。

成功提交代码后，gitbash 的反馈如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git commit -m 第三次提交，我们增加了一个鼠标模型，可以判断鼠标的移动和坐标，并显示了鼠标坐标，并在该页面使用了教材的背景图片。还增加了一个键盘模型，可以识别键盘按下的字母以及ASCII码的编号。
[master cbc8193] 第三次提交，我们增加了一个鼠标模型，可以判断鼠标的移动和坐标，并显示了鼠标坐标，并在该页面使用了教材的背景图片。还增加了一个键盘模型，可以识别键盘按下的字母以及ASCII码的编号。
 3 files changed, 3 insertions(+), 2 deletions(-)
```

我们可以输入日志命令查看，

```
$gitlog
```

gitbash 反馈代码的仓库日志如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git log
commit cbc819319acce79b25897069491d61b7f7128949 (HEAD -> master)
Author: 江科大周志能 <2031552294@qq.com>
Date:   Wed Jun 12 15:44:36 2024 +0800
```

第三次提交，我们增加了一个鼠标模型，可以判断鼠标的移动和坐标，并显示了鼠标坐标，并在该页面使用了教材的背景图片。还增加了一个键盘模型，可以识别键盘按下的字母以及ASCII码的编号。

第六章个性化 UI 设计中对鼠标交互的设计开发

6.1 分析与设计

在 UI 中尝试对鼠标设计控制，设计模拟手机端的触屏效果，设置触屏条件横向移动 100px 为有效拖动，否则为无效拖动。使用鼠标按下、松开、移动模拟手指横向滑动屏幕的操作，添加 Eventlistener 实现 mouseup、mousedown、mouseout 以及mousemove 的功能。鼠标移动时会显示正在拖动鼠标和拖动距离，当鼠标横向拖动距离大于 100px 显示“鼠标松开！这次是有效拖动！”，拖动距离小于 100px 显示“鼠标松开！这次是无效拖动！”。

如用例图 6.1 所示：

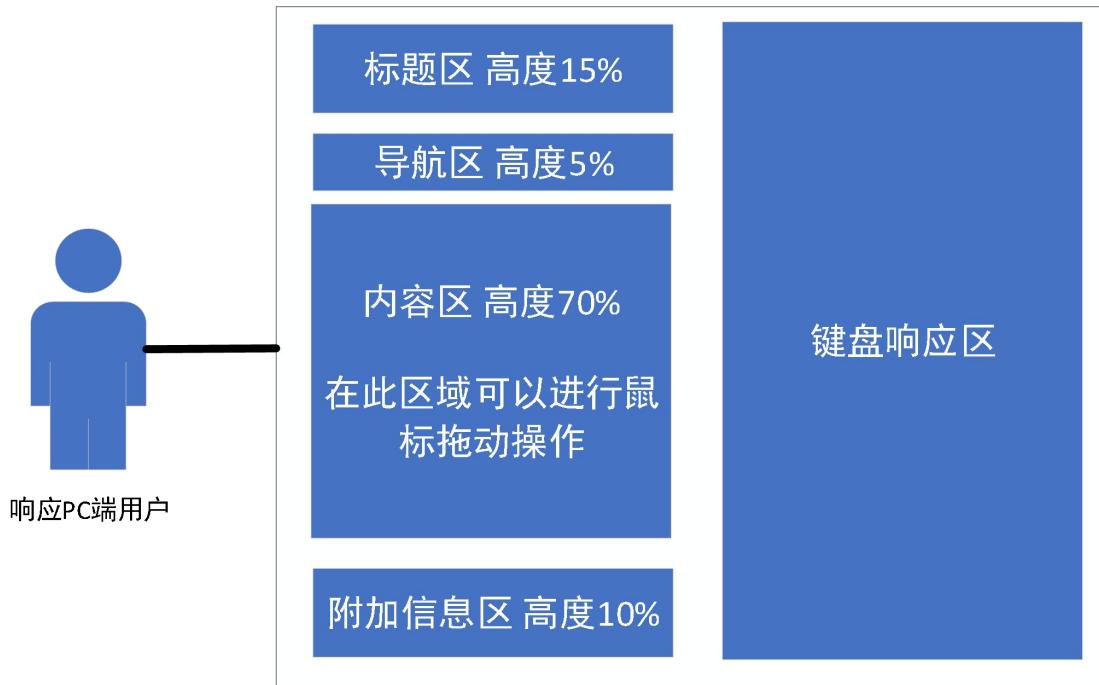


图 6.1 用例图

DOM 树如下图 6.2 所示：

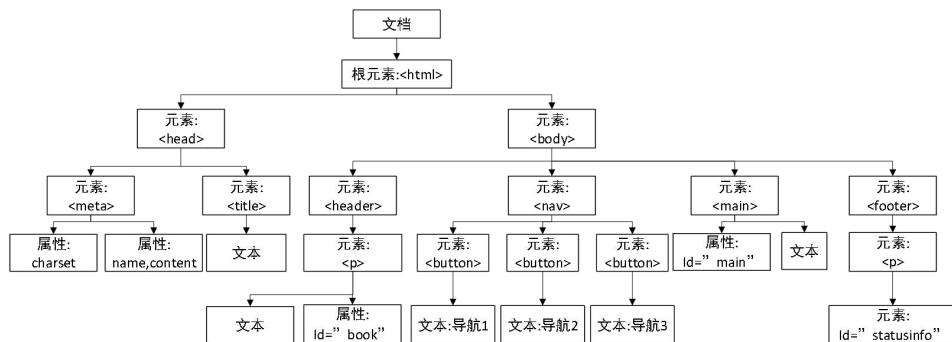


图 6.2 DOM 树 1.4

6.2 项目的实现和编程

以下代码块是对 `mouseup` 和 `mousedown` 的实现，当鼠标按下时，在控制台处显示鼠标按下位置的坐标，当鼠标松开时，通过判断鼠标移动距离，确认鼠标拖动是否有效。

HTML 代码编写如代码块 6-1：

```

<header>
<p id="book">
《WebAppUI 设计开发》
</p>
  
```

```
</header>

<nav>
    <button id="prev">向前</button>
    <button id="next">向后</button>
    <button>其他</button>
</nav>

<main id="main">
    <div id="bookface">
        这是书的封面图<br>
        在此对象范围拖动鼠标(本例触屏无效)
    </div>
</main>

<footer>
    CopyRight 刘盛江西科技师范大学 2024--2025
</footer>
<div id="aid">
    <p>用户键盘响应区</p>
    <pid="keyboard"></pid>
</div>
```

代码块 6-1

Css 代码块下代码块 6-2 所示:

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

```
header{  
border:3pxsolidgreen;  
height:10%;  
font-size:1em;  
  
}  
  
nav{  
border:3pxsolidgreen;  
height:10%;  
}  
  
main{  
border:3pxsolidgreen;  
height:70%;  
font-size:0.8em;  
position:relative;  
}  
  
  
#box{  
position:absolute;  
right:0;  
width:100px;  
}  
  
  
footer{  
border:3pxsolidgreen;  
height:10%;  
font-size:0.7em;  
}  
  
body{  
position:relative;
```

```
}

button{
font-size:1em;
}

#aid{
position:absolute;
border:3pxsolidblue;
top:0px;
left:600px;
}

#bookface{
position:absolute;
width:80%;
height:80%;
border:1pxsolidred;
background-color:blanchedalmond;
left:7%;
top:7%;
background-image:url("../img/CSS.jpg");
background-size:cover;
}
```

代码块 6-2

与上阶段相比，本阶段增加了使书的封面前后切换的功能，且书的封面可以左右移动，移动距离大于 100px 的为有效拖动，可以计算拖动距离，js 代码块如 6-3 所示：

```
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("mouseDownatx:"+"("+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("mouseisDownandmoving");
    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';
}

});

$("body").addEventListener("keypress",function(ev){
letk=ev.key;
letc=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
}
letdom=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;
}
}

//endof$

var myDiv=document.getElementById('bookface');

var images=[

"../imgs/css.jpg",
"../imgs/CS.jpg",
"../imgs/CT.jpg",
"../imgs/GIT.jpg",
"../imgs/internet.jpg",
"../imgs/javaScript.jpg",
"../imgs/learnCSS.jpg",
"../imgs/linuxCMD.jpg",
"../imgs/STEM.jpg",
"../imgs/ted.jpg",
"../imgs/UML.jpg",
"../imgs/videos.jpg",
"../imgs/webProgramming.jpg",
];

//当前位置

var currentImageIndex=0;

document.getElementById("next").onclick=function(){

currentImageIndex++;

if(currentImageIndex==images.length)currentImageIndex=0;
myDiv.style.backgroundImage='url('+images[currentImageIndex]+')';
};
```

```
document.getElementById("prev").onclick=function(){
    currentImageIndex--;
    if(currentImageIndex== -1)currentImageIndex=images.length-1;
    myDiv.style.backgroundImage='url('+images[currentImageIndex]+')';
};

};
```

代码块 6-3

6.3 项目的运行和测试

本次测试主要测试 PC 端的封面移动用例，如下图 6.3，6.4 所示：



图 6.3 鼠标模型无效拖动示例图



图 6.4 鼠标模型有效拖动示例图

移动端用户可以通过扫描图 6.5 的二维码，运行测试本项目的第四次开发的阶段性效果。



图 6.5 移动端二维码

6.4 项目的代码提交和版本管理

再次编写好 index.html, myCss.cssm,myjs.js 的代码，测试运行成功后，执行下面命令提交代码：
`$git add index.html myCss.cssm myjs.js`
`$git commit -m“第四次提交，使用鼠标按下、抬起、拖动等操作模拟手段的横向滑动屏幕的操作，并且计算出拖动的距离。还增加向前向后按钮，点击按钮切换书的封面图”`

成功提交代码后，gitbash 的反馈如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git add index.html mycss.css myjs.js

R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git commit -m "第四次提交,使用鼠标按下、抬起、拖动等操作模拟了手机端的横向滑动屏幕的操作，并且计算出拖动的距离，增加了向前向后按钮，点击按钮切换书的封面图。"
[master d49ad92] 第四次提交,使用鼠标按下、抬起、拖动等操作模拟了手机端的横向滑动屏幕的操作，并且计算出拖动的距离，增加了向前向后按钮，点击按钮切换书的封面图。
 3 files changed, 3 insertions(+), 3 deletions(-)
```

我们可以输入日志命令查看，

```
$gitlog
```

gitbash 反馈代码的仓库日志如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git log
commit d49ad92ef9f21c6735202960a5945c062a2355e2 (HEAD -> master)
Author: 江科大周志能 <2031552294@qq.com>
Date:   Wed Jun 12 15:51:56 2024 +0800
```

第四次提交,使用鼠标按下、抬起、拖动等操作模拟了手机端的横向滑动屏幕的操作，并且计算出拖动的距离，增加了向前向后按钮，点击按钮切换书的封面图。

第七章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

当点击鼠标或触屏时显示鼠标在方框内的具体坐标，并在鼠标或触屏拖动时显示拖动是否有效，当拖动有效时显示鼠标或触屏的拖动到拖动距。

创建 Pointer 实现对鼠标和触屏设计一套代码实现 UI 控制。在 bookface 中添加 handleBegin, handleEnd, handleMoving 三个 EventListener。首先判断操作为触屏还是鼠标操作，并进行相应的操作反馈。

用例图如图 7.1 所示：

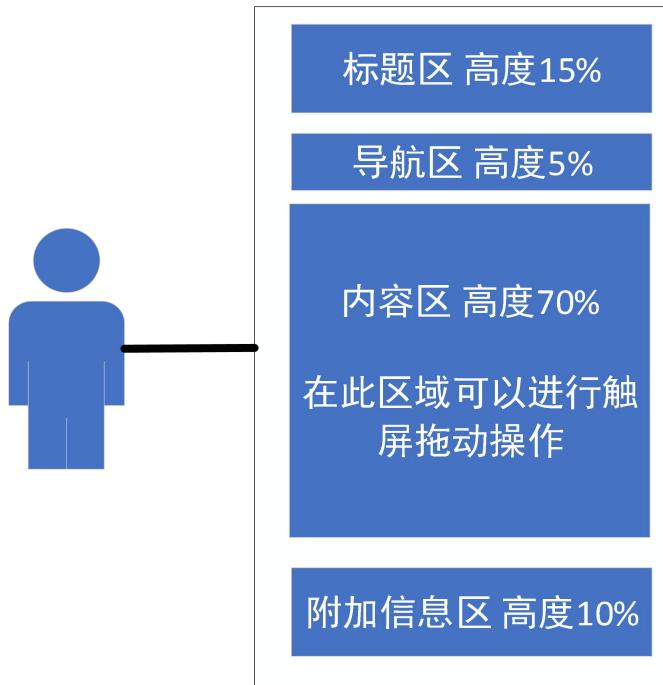


图 7.1 用例图

DOM 树如下图 7.2 所示：

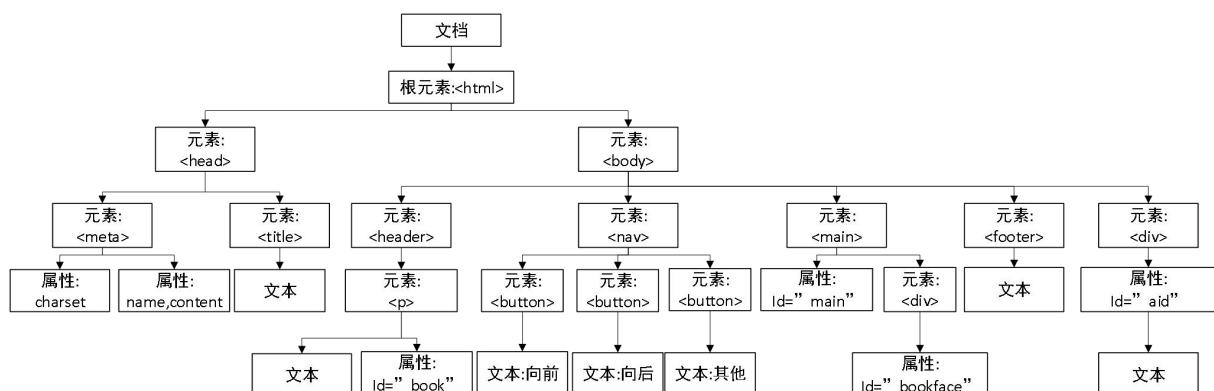


图 7.2 DOM 树 1.5

7.2 项目的实现和编程

添加 EventListener，对 handleBegin 的实现，当点击鼠标或触屏时，显示对应的坐标。对 handleEnd 功能的实现，显示鼠标或触屏的拖动操作，当拖动距离超过 100 时，拖动无效。对 handleMoving 功能的实现，显示鼠标及触屏拖动操作时的移动距离。因为本次迭代没有对 HTML 代码和 CSS 代码进行修改，所以此次代码展示只提交 js 代码，js 代码如代码块 7-1 所示：

```
//尝试对鼠标和触屏设计一套代码实现 UI 控制
varPointer={};
Pointer.isDown=false;
```

```

Pointer.x=0;

Pointer.deltaX=0;

{//CodeBlockbegin

lehandleBegin=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("Touchbegin:"+("("+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("PointerDownatx:"+("("+Pointer.x+","+Pointer.y+"));

$("bookface").textContent="鼠标按下, 坐标: "+("("+Pointer.x+","+Pointer.y+");

}

};

lehandleEnd=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 is down 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" ).addEventListerner( "mousedown" , handleBegin );  
$( "bookface" ).addEventListerner( "touchstart" , handleBegin );  
$( "bookface" ).addEventListerner( "mouseup" , handleEnd );  
$( "bookface" ).addEventListerner( "touchend" , handleEnd );  
$( "bookface" ).addEventListerner( "mouseout" , handleEnd );  
$( "bookface" ).addEventListerner( "mousemove" , handleMoving );  
$( "bookface" ).addEventListerner( "touchmove" , handleMoving );  
$( "body" ).addEventListerner( "keypress" , function( ev ) {  
    $( "aid" ).textContent += ev.key;  
});  
}//CodeBlockend
```

代码块 7-1

7.3 项目的运行和测试

此次功能测试主要使正对手机端用户，可以进行横向触屏操作功能，显示触屏坐标测，如下图 7.3, 7.4 所示：

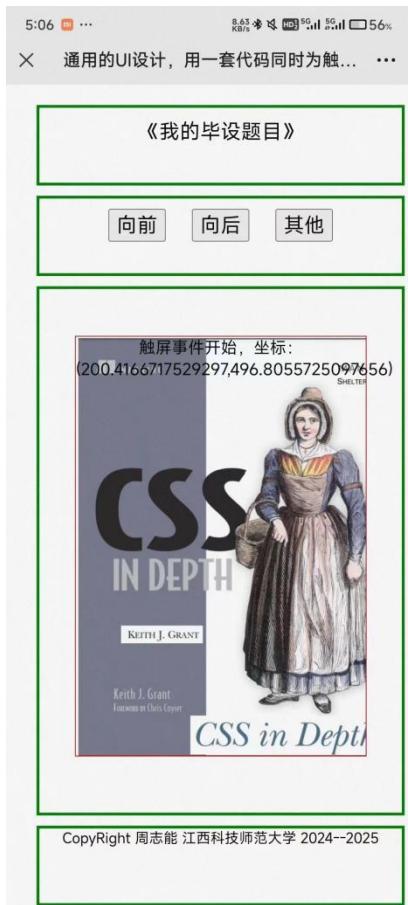


图 7.3 显示触屏坐标测试图



图 7.4 显示触屏拖动距离测试图

移动端用户可以通过扫描图 7.5 的二维码，运行测试本项目的第五次开发的阶段性效果。



图 7.5 移动端二维码

7.4 项目的代码提交和版本管理

再次编写好 index.html, myCss.cssm,myjs.js 的代码，测试运行成功后，执行下面命令提交代码：

```
$git add myjs.js
```

```
$gitcommit-m 第五次提交，这次我们设置了手机用户也可以触屏拖动的功能，还确定了拖动的范围，超过100像素的，我视为有效的UI互动，并且将键盘响应区改为只输出按下的相应按键，不再输出对应的ASCII码。
```

成功提交代码后，gitbash的反馈如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git commit -m 第五次提交，这次我们设置了手机用户也可以触屏拖动的功能，还确定了拖动的范围，超过100像素的，我视为有效的UI互动，并且将键盘响应区改为只输出按下的相应按键，不再输出对应的ASCII码。
[master 7d340df] 第五次提交，这次我们设置了手机用户也可以触屏拖动的功能，还确定了拖动的范围，超过100像素的，我视为有效的UI互动，并且将键盘响应区改为只输出按下的相应按键，不再输出对应的ASCII码。
 3 files changed, 3 insertions(+), 2 deletions(-)
```

我们可以输入日志命令查看，

```
$gitlog
```

gitbash反馈代码的仓库日志如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git log
commit 7d340dfc339592c9f44d45d227ac17ad786e3524 (HEAD -> master)
Author: 江科大周志能 <2031552294@qq.com>
Date:   Wed Jun 12 15:54:00 2024 +0800
```

第五次提交，这次我们设置了手机用户也可以触屏拖动的功能，还确定了拖动的范围，超过100像素的，我视为有效的UI互动，并且将键盘响应区改为只输出按下的相应按键，不再输出对应的ASCII码。

第八章 UI 的个性化键盘交互控制的设计开发

8.1 分析与设计

当敲击键盘和松开键盘时对所按按键的键值和对应代码输出出来。添加两个 EventListener，利用keydown 和keyup 两个底层事件，实现同时输出按键状态和文本内容。

用例图如下图 8.1 所示：

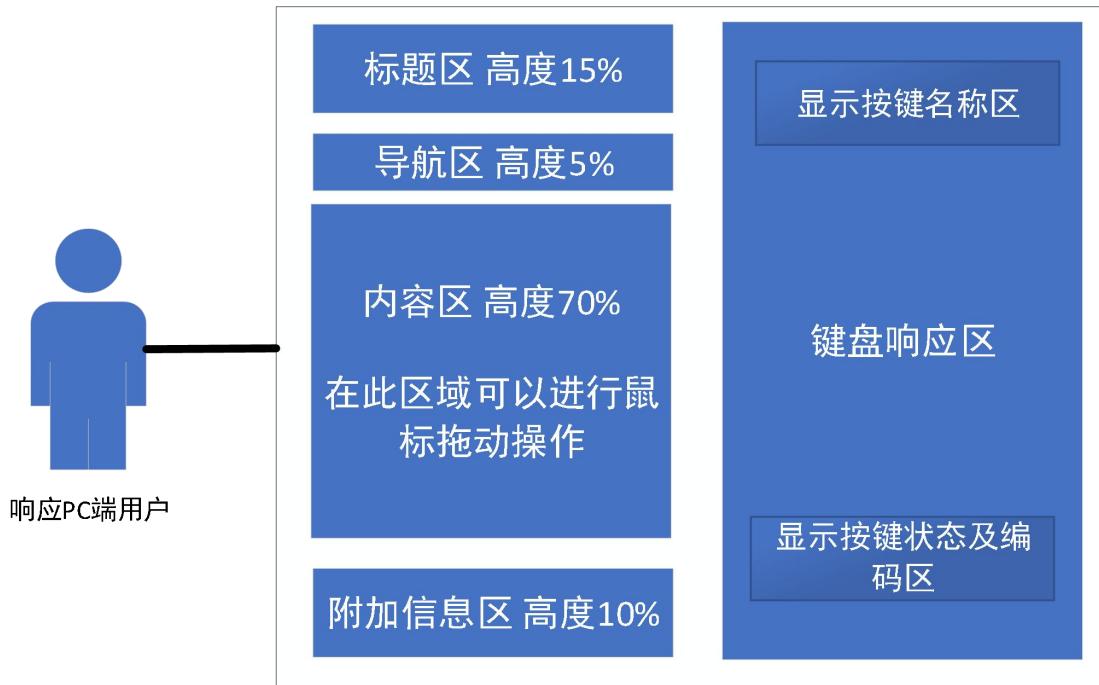


图 8.1 用例图

DOM 树如下图 8.2 所示：

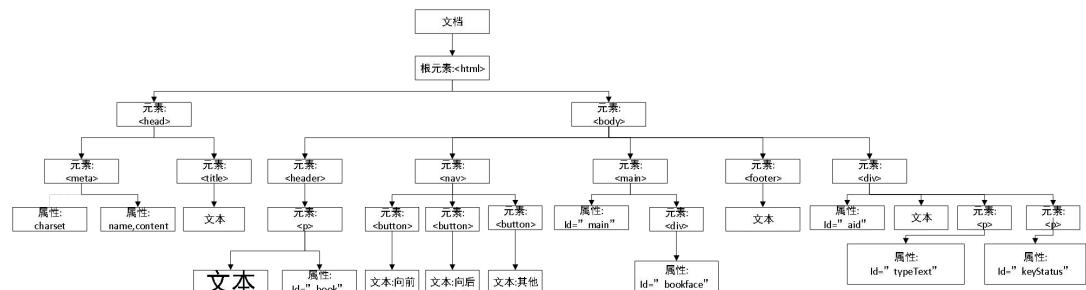


图 8.2 DOM 树 1.6

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。

HTML 代码编写如代码块 8-1 所示：

```
<nav>
<button id="prev">向前</button>
<button id="next">向后</button>
<button>其他</button>
</nav>
<main id="main">
```

```
<div id="bookface">  
本利代码的运行需要超过 1 千像素宽度的宽屏<br>  
建议使用有键盘的 PC 运行和调试代码<br>  
当然拖动/滑动超过 100 像素的 UI 互动依然有效!  
</div>  
</main>  
<footer>  
    CopyRight 刘盛江西科技师范大学 2024--2025  
</footer>  
<div id="aid">  
用户键盘响应区  
<p id="typeText"></p>  
<p id="keyStatus"></p>  
</div>
```

代码块 8-1

Css 代码块下代码块 8-2 所示:

```
*{  
margin:10px;  
text-align:center;  
}  
  
header{  
border:3px solid green;  
height:10%;  
font-size:1em;  
  
}  
  
nav{  
border:3px solid green;  
height:10%;  
}
```

```
main{
border:3pxsolidgreen;
height:70%;
font-size:0.8em;
position:relative;
}

footer{
border:3pxsolidgreen;
height:10%;
font-size:0.7em;
}
body{
position:relative;
}
button{
font-size:1em;
}

#bookface{
position:absolute;
width:80%;
height:80%;
border:1pxsolidred;
background-color:blanchedalmond;
left:7%;
top:7%;
background-image:url("../lesson/CS.jpg");
background-size:cover;
```

```

}

#aid{
position:absolute;
border:3pxsolidblue;
top:0px;
left:600px;
}

#typeText{
border:1pxsolidblue;
padding:0.2em;
color:gray;
}

#keyStatus{
position:absolute;
border:1pxsolidblue;
width:90%;
right:0;
bottom:0;
font-size:0.6em;
padding:0.5em;
}

```

代码块 8-2

添加两个 EventListener，keydown 显示按键是的键值和字符编码，keyup 显示松开按键时的键值和字符编码，添加功能 printLetter(k)确保只有一个按键。js 代码块如代码块 8-3 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

var text=$("#typeText");

$("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;
    printLetter(k);
})

```

```

let c=ev.keyCode;
if(c==13){
let pchild=document.createElement("p");
$("#aid").appendChild(pchild);
text=pchild;
} else text.textContent+=k;
$("#keyStatus").textContent="按下键: "+k+", "+"编码: "+c;
});
$("body").addEventListener("keyup",function(ev){
    ev.preventDefault();
let key=ev.key;
$("#keyStatus").textContent=key+"键已弹起";
if(printLetter(key)){
    text.textContent+=key;
}
function printLetter(k){
    if(k.length>1){ //学生须研究这个逻辑的作用
        return false;
    }
    let puncs=[ '~','`','!','@','#','$','%','^','&','*','(',')','-','_','+', '=',
',','.',';',';','<','>','?','/',' ','\\','\\' ];
    if((k>='a'&&k<='z')||(k>='A'&&k<='Z')||(k>='0'&&k<='9')){
        console.log("letters");
        return true;
    }
    for(let p of puncs){
        if(p==k){
            console.log("puncs");
            return true;
        }
    }
}

```

```
}

returnfalse;

//提出更高阶的问题，如何处理连续空格和制表键 tab？

} //functionprintLetter(k)

});

}//CodeBlockEnd
```

代码块 8-3

8.3 项目的运行和测试

本次项目测试主要就是测试 keydown 和 keyup 的功能，测试图如下图 8.3 和 8.4 所示。



图 8.3 keydown 功能测试图



图 8.4keyup 功能测试图

移动端用户可以通过扫描图 8.5 的二维码，运行测试本项目的第六次开发的阶段性效果。



图 8.5 手机端二维码

8.4 项目的代码提交和版本管理

再次编写好 index.html, myCss.cssm,myjs.js 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 myjs.js
```

```
$git commit -m 第六次提交，我完善了一下用户键盘响应区的功能，既能输出键盘按下的字符，也能输出对应的 ASCII 码，还能识别按钮是否弹起。
```

成功提交代码后，gitbash 的反馈如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git commit -m 第六次提交，我完善了一下用户键盘响应区的功能，既能输出键盘按下的字符，也能输出对应的ASCII码，还能识别按钮是否弹起
[master 8d5234b] 第六次提交，我完善了一下用户键盘响应区的功能，既能输出键盘按下的字符，也能输出对应的ASCII码，还能识别按钮是否弹起
 3 files changed, 3 insertions(+), 3 deletions(-)
```

我们可以输入日志命令查看，

```
$gitlog
```

gitbash 反馈代码的仓库日志如下所示：

```
R7000@Lenovo MINGW64 /d/桌面/webUI (master)
$ git log
commit 8d5234b279bae62d279fbc98b7f5a02ab98fc878 (HEAD -> master)
Author: 江科大周志能 <2031552294@qq.com>
Date:   Tue Jun 18 21:43:24 2024 +0800
```

```
第六次提交，我完善了一下用户键盘响应区的功能，既能输出键盘按下的字符，也能输出对应的ASCII码，还能识别按钮是否弹起
```

第九章 谈谈本项目中的高质量代码

9.1 编程的作用

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一样工作，那么你可以打开对应的应用程序，让其开始工作。但是对于唯一的或开放式的任务，可能没有与之对应的应用程序。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，编程从根本上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的思考，那么编程可能是值得的。它能让你在几秒钟内完成一些手工需要很久才能完成的事情。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。

9.2 项目的高质量代码

创建一个 Pointer 对象，践行 MVC 设计模式，设计一套代码同时对鼠标和触屏实现控制。
面向对象思想，封装，抽象，局部变量，函数式编程，逻辑。（围绕着抽象定义函数、代码块、模型设计以及降低全局变量的使用来写）

以下是实现代码：

```

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+"));
    }
};


```

图 9.1 用 Pointer 对象实现鼠标模型代码详情图

```

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%' ;
        }
    }
};


```

图 9.2 鼠标拖动以及触屏操作代码详情图

```

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' ;
        }
    }
};

```

图 9.3 鼠标拖动或触屏滑动距离计算代码详情图

第十章用 gitBash 工具管理项目的代码仓库和 http 服务器

10.1 经典 Bash 工具介绍

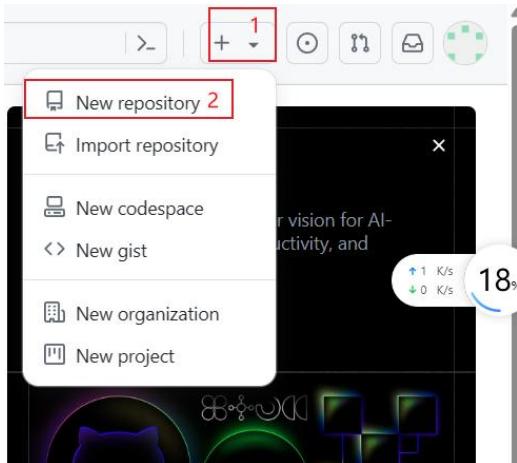
当我们谈到命令行时，我们实际上指的是 shell。shell 是一个接受键盘命令并将其传递给操作系统执行的程序。几乎所有的 Linux 发行版都提供了一个来自 GNU 项目的 shell 程序，名为 bash。这个名字是 Bourne-againshell 的首字母缩略词，bash 是 sh 的增强替代品，sh 是 Steve Bourne 编写的原始 Unix shell 程序。

和 Windows 一样，像 Linux 这样的类 unix 操作系统用所谓的分层目录结构来组织文件。这意味着它们被组织成树状的目录模式(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，子目录包含更多的文件和子目录，以此类推。

10.2 通过 GitHub 平台实现本项目的全球域名

10.2.1 创建一个空的远程代码仓库

注册并登录 github 网站后，创建仓库，步骤如下图所示。



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 *
 ykzhouzhineng	/ yk.github.io
仓库名设置为如下格式: 账户名.github.io	
<input checked="" type="checkbox"/> yk.github.io is available.	
<p>Great repository names are short and memorable. Need inspiration? How about didactic-invention ?</p> <p>Description (optional)</p> <hr/>	
<input type="button" value="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。

10.3 设置本地仓库和远程代码仓库的链接

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

```
$echo"WebUI 应用的远程 http 服务器设置">>>README.md
```

创建内容为 WebUI 应用的远程 http 服务器设置的名称为 README 的 markdown 文件

```
$git init
```

初始化一个空的 git 本地仓库

```
$git add README.md
```

将 README.md 文件添加暂存区

```
$git commit -m"这是我第一次把代码仓库上传至 GitHub 平台"
```

将暂存区内容添加到本地仓库

```
$gitbranch-Mmain
```

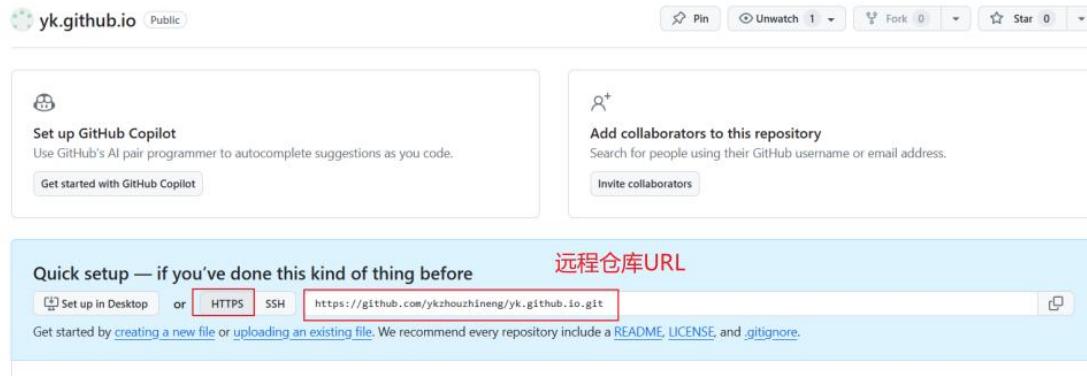
将当前分支重命名为 main

```
$gitremoteaddoriginhttps://github.com/ykzhouzhineng/WebUI.git
```

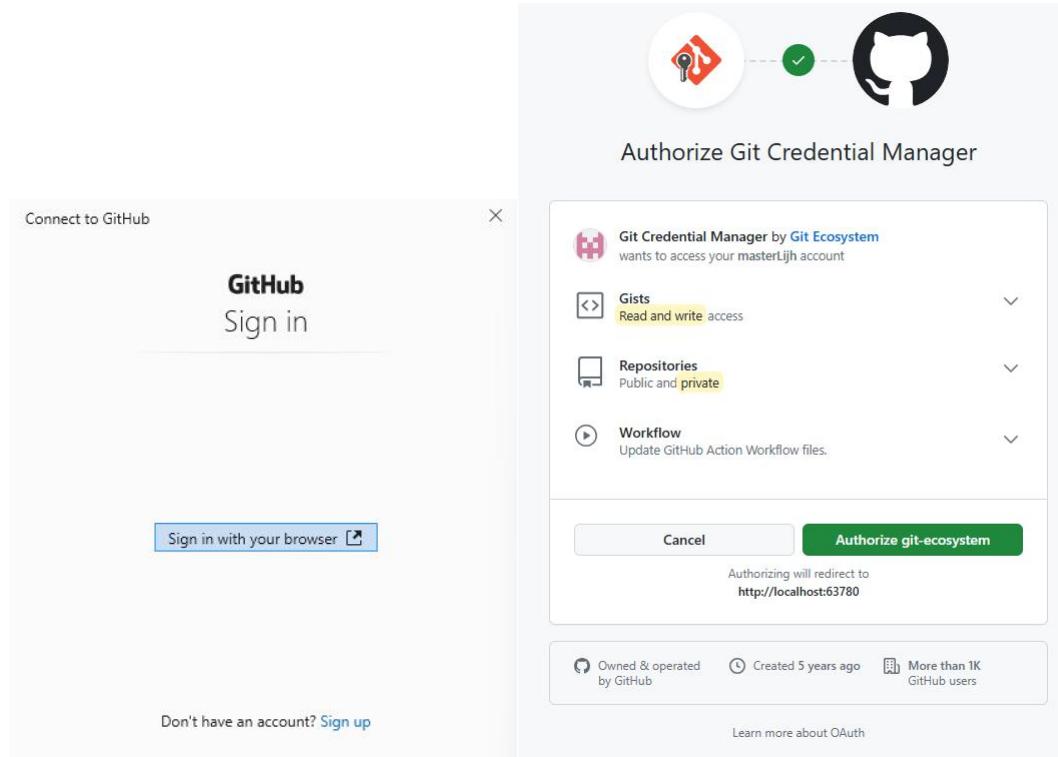
将本地 git 仓库和远程 github 仓库链接起来，并将其命名为 origin

```
$gitpush-uoriginmain
```

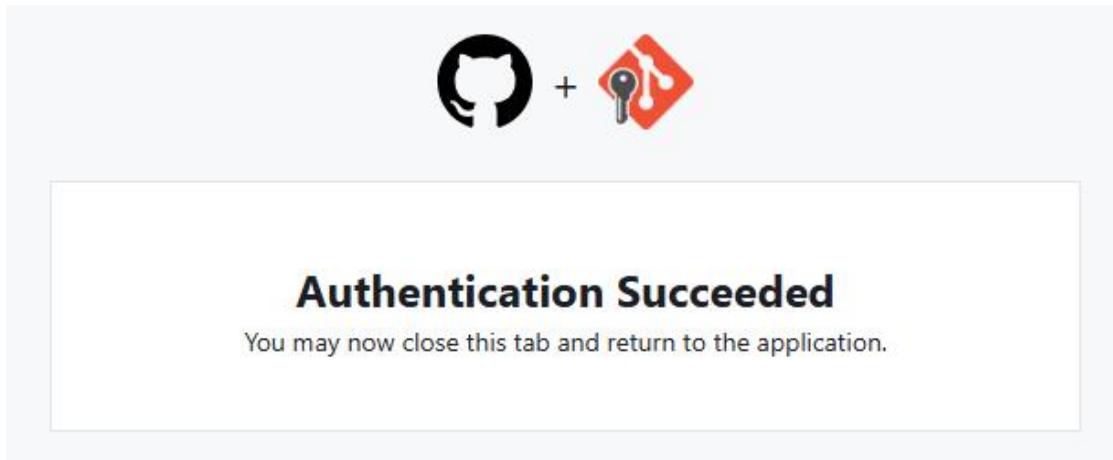
将本地的提交推送至远程仓库中



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，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。



自此以后，无论在本地修改了多少次代码，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至 `github` 平台，而远程上传命令则可简化为一条：`gitpush`，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 端的微软 Edge 浏览器打开，本文截取操作中间的效果图，如下所示：

A screenshot of a Microsoft Edge browser window. On the left, a green-bordered tab titled "《我的毕设题目》" is open. It contains a navigation bar with three buttons: "向前" (Forward), "向后" (Back), and "其他" (Other). Below the navigation bar is a warning message: "本例代码的运行需要超过1千像素宽度的宽屏" (The example code runs on a screen width of more than 1000 pixels), followed by "建议使用有键盘的PC运行和调试代码" (It is recommended to run and debug the code on a PC with a keyboard) and "当然拖动/滑动超过100像素的UI互动依然有效!" (Of course, dragging/sliding UI interactions over 100 pixels are still effective!). Below this message is a large "CSS IN DEPTH" logo featuring a woman's portrait. At the bottom of the tab, it says "CopyRight 周志能 江西科技师范大学 2024--2025". On the right, a blue-bordered tab titled "用户键盘响应区" (User Keyboard Response Area) is open. It contains a text input field with the placeholder "sdasdasdsadssdssd" and a status message at the bottom: "d 键已弹起" (Key d has been pressed).

参考文献

- [1]. W3C.W3C'shistory.W3CCommunity.[EB/OL].[38](https://www.w3.org/about/.https://www.w3.org/about/history/>.2023.12.20[2]. DouglasE.Comer.TheInternetBook[M](FifthEdition).CRCPressTaylor&FrancisGroup,2019:217-218[3]. JohnDean,PhD.WebprogrammingwithHTML5,CSS, andJavaScript[M].Jones&BartlettLearning, LLC.2019:2[4]. JohnDean,PhD.WebprogrammingwithHTML5,CSS, andJavaScript[M].Jones&BartlettLearning, LLC.2019:xi[5]. BehrouzForouzan.FoundationsofComputerScience[M](4thEdition).CengageLearningEMEA,2018:274--275[6]. MarijnHaverbeke.EloquentJavaScript3rdedition.NoStarchPress,Inc,2019.[7]. WilliamShotts.TheLinuxCommandLine,2ndEdition[M].NoStarchPress,Inc,2458thStreet,SanFrancisco,CA94103,2019: 3-7</div><div data-bbox=)