成 绩：

江西科技师范大学

**毕业设计（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**

**院（系）： 元宇宙产业学院**

**专 业： 计算机科学与技术**

**学生姓名： 宁方昀**

**学 号： 20213640**

**指导教师： 李健宏**

**2024年 6 月 1日**

**目录**

[题目（中文）：基于Web客户端技术的个性化UI设计和实现 I](#_Toc28705)

[第一章 前言 1](#_Toc25499)

[1.1 项目概要 1](#_Toc9057)

[1.2 研学计划 2](#_Toc21283)

[1.3 研究方法 2](#_Toc32142)

[第二章 技术总结和文献综述 2](#_Toc31848)

[2.1 Web平台和客户端技术概述 2](#_Toc17397)

[2.2 项目的增量式迭代开发模式 4](#_Toc9650)

[第三章 内容设计概要 5](#_Toc2368)

[3.1 分析和设计 5](#_Toc28280)

[3.2 项目的实现和编程 6](#_Toc20635)

[CSS代码编写如代码块3-2： 6](#_Toc28798)

[3.3 项目的运行和测试 7](#_Toc2779)

[3.4 项目的代码提交和版本管理 8](#_Toc9960)

[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 9](#_Toc17865)

[4.1 分析和设计 9](#_Toc12988)

[4.2 项目的实现和编程 10](#_Toc26386)

[4.3 项目的运行和测试 12](#_Toc4272)

[4.4 项目的代码提交和版本管理 13](#_Toc22375)

[第五章 应用响应式设计技术开发可适配窄屏和宽屏的UI 14](#_Toc12858)

[5.1 分析与设计 14](#_Toc26569)

[5.2 项目的实现和编程 15](#_Toc15411)

[5.3 项目的运行和测试 18](#_Toc18974)

[5.4 项目的代码提交和版本管理 20](#_Toc32661)

[第六章 个性化UI设计中对鼠标交互的设计开发 21](#_Toc32635)

[6.1 分析与设计 21](#_Toc9493)

[6.2 项目的实现和编程 21](#_Toc29138)

[6.3 项目的运行和测试 25](#_Toc2732)

[6.4 项目的代码提交和版本管理 26](#_Toc23514)

[第七章 对触屏和鼠标的通用交互操作的设计开发 27](#_Toc2637)

[7.1 分析和设计 27](#_Toc31085)

[7.2 项目的实现和编程 27](#_Toc23380)

[7.3 项目的运行和测试 30](#_Toc2998)

[7.4 项目的代码提交和版本管理 32](#_Toc3195)

[第八章 UI的个性化键盘交互控制的设计开发 33](#_Toc22495)

[8.1 分析与设计 33](#_Toc16974)

[8.2 项目的实现和编程 33](#_Toc15392)

[8.3 项目的运行和测试 39](#_Toc1659)

[8.4 项目的代码提交和版本管理 40](#_Toc5731)

[第九章 谈谈本项目中的高质量代码 41](#_Toc5750)

[9.1编程的作用 41](#_Toc20687)

[9.2 项目的高质量代码 41](#_Toc21562)

[第十章 用gitBash工具管理项目的代码仓库和http服务器 43](#_Toc19459)

[10.1 经典Bash工具介绍 43](#_Toc21052)

[10.2 通过gitHub平台实现本项目的全球域名 43](#_Toc23491)

[10.2.1 创建一个空的远程代码仓库 43](#_Toc31688)

[10.3 设置本地仓库和远程代码仓库的链接 44](#_Toc29278)

[参考文献 46](#_Toc9560)

**摘要**：**近十年间，HTML5作为核心的Web标准技术，凭借其跨平台兼容性和开源的特性，在多个领域的软件开发中得到了广泛应用。通过对本次毕业设计任务的深入分析，我们决定采用HTML5的Web客户端技术作为项目的技术基础，深入研究和实践程序设计与软件开发。我们成功设计并开发了一款个性化用户界面（UI）的应用程序。在开发过程中，我们利用HTML进行内容的结构化建模，通过CSS精心打造UI的外观样式，并借助JavaScript编程技术实现UI的交互功能。从工程管理的视角出发，我们采用了增量式开发模式，通过A（分析）、D（设计）、I（实现）、T（测试）的循环迭代，共进行了六次代码增量式项目迭代，以逐步求精的方式高效推进项目的进展。最终，我们利用gitBash将项目代码上传至Github，实现了该UI在全球互联网上的部署。现在，用户可以通过URL和二维码轻松地跨平台访问并高效使用这款程序，这标志着本项目的设计开发和测试工作取得了圆满的成功。**

**关键词：html5；个性化UI；增量式开发模式；**

# 前言

## 项目概要

本项目以HTML5的Web客户端技术为核心，深入研究和实践了程序设计与软件开发。通过广泛参考技术书籍、开发者论坛和文献，特别是Mozilla组织的MDN社区中的技术实践文章，我们掌握了HTML内容建模、CSS样式设计和JavaScript功能编程的精髓。在此基础上，我们设计并开发了一个个性化用户界面（UI）的应用程序。在开发过程中，我们充分运用了HTML进行内容建模，CSS来打造UI的外观设计，以及JavaScript编程实现UI的交互功能。本项目所有代码均为手工编写，未导入任何外部代码（包括框架和库），确保了代码的原创性和纯净性。为了实现一个能够响应各种屏幕尺寸的用户界面，本项目采用了响应式设计编程技术，使得应用程序能够自动适配PC端和移动端设备。同时，借助DOM技术和事件驱动模式，程序能够灵敏地响应鼠标、触屏、键盘等底层事件，为用户提供流畅的操作体验。我们特别设计了一个对象模型来模拟鼠标和触屏设备，这既是本项目模型研究法的一次创新实践，也是其亮点之一。此外，本项目还大量运用了面向对象的程序设计思想，通过构建一个通用的pointer模型，实现了对鼠标和触屏的统一控制，从而提高了代码的质量和可维护性。从代码的开源和分享角度出发，本项目采用了git工具进行版本管理。在开发过程中，我们重构了六次代码，并正式提交了这些版本。在测试阶段，我们还进行了两次代码修改并提交。最终，我们利用git bash工具将本项目的代码仓库上传至GitHub，建立了自己的代码仓库，并将其设置为HTTP服务器，使本UI应用能够方便地被全球用户访问。

## 研学计划

本项目规划了六个阶段进行迭代开发，借助git bash工具进行项目提交，通过GitHub仓库进行代码的存储与管理。同时，我们以开源的方式与全球开发者共享此项目。这六个阶段从初步的内容设计到移动互联功能的实现，再到响应式UI的完善，逐步推进。

在第一阶段，我们主要运用HTML与CSS技术实现基础的内容展示。进入第二阶段，我们在前一阶段的基础上，利用更高级的CSS语法和JavaScript技术，使内容呈现更为丰富，并添加导航功能以便用户精确定位所需内容。

第三阶段则着重于实现移动互联的响应式UI设计。我们根据键盘和鼠标的响应来展示针对不同设备的个性化UI，确保内容在各种系统上都能得到恰当的展示。

随后的三个阶段，我们专注于个性化UI设计的交互响应以及适应移动互联网时代各种屏幕尺寸的优化。我们致力于完善与实现更多个性化的UI设计元素。

最终，我们通过HTTP服务使该UI应用得以全球访问，确保用户无论身处何地都能轻松体验我们的产品。

## 研究方法

首先，我们深入研究了与Web客户端技术相关的个性化UI设计和实现的文献资料，旨在明确用户在移动互联时代对个性化UI设计的具体需求和期望。在此基础上，我们详细分析了项目的可行性，并决定采用软件工程中广受欢迎的迭代增量式开发模型作为项目的设计框架。

迭代增量式开发模型将软件开发过程细分为多个小型的迭代周期，每个迭代周期都会生成一个可运行、可测试的软件增量。随着迭代的不断推进，这些增量会逐步累积和完善，最终形成完整的软件系统。

基于定性和定量研究收集的数据，我们进行了系统的统计分析和数据挖掘，以获取用户需求的深刻洞察。根据这些数据分析结果，我们利用迭代增量式开发模型，逐步推进个性化UI设计的实现过程，确保每个迭代周期都能满足用户的具体需求。

最终，我们总结了Web客户端技术中个性化UI设计和实现的关键问题和挑战，并对未来可能的发展趋势和创新方向进行了展望。这些研究成果不仅为相关领域的学术研究提供了有价值的参考，也为业界实践者提供了重要的启示和指导。

# 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。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。

1989年，为了帮助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”的想法，这样当一篇论文引用另一篇论文时，读者可以点击链接并快速跳转到另一篇论文。从1989年到1991年，伯纳斯-李相当多产:(1)他设计了HTML，超文本链接作为关键特性;(2)他设计了万维网背后的概念，包括HTTP协议;(3)他创建了一个使用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)提交了第一个关于HTML的正式提案。1994年，蒂姆·伯纳斯-李在麻省理工学院创立了万维网联盟(W3C)，并接管了HTML标准的管理工作。他编写了“超文本标记语言”(HTML)的第一个版本，这种文档格式语言具有超文本链接的功能，成为Web的主要发布格式。随着Web技术的传播，他对uri、HTTP和HTML的最初规范进行了改进和讨论。

万维网联盟（World Wide Web Consortium，简称W3C）是由万维网的发明者蒂姆·伯纳斯-李在1994年10月创立的，其成立地点位于美国麻省理工学院计算机科学实验室（MIT/LCS），并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的成立标志着万维网的正式规范化发展阶段，旨在促进网络技术的统一和标准化，确保网络信息的普遍可访问性和兼容性。

初期，W3C主要关注HTML等基础网页技术的标准制定，随着互联网的迅速发展，它的工作范围逐渐扩展到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）、Web Accessibility（网页无障碍）、Web Services（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，极大地推动了互联网技术的进步和应用的普及。

随着时间的推移，W3C的成员组成也日益国际化，从最初的几家创始机构扩展到覆盖全球40多个国家的400多个会员组织，并在全球多个地区设立了办事处，体现了其作为国际性标准组织的广泛影响力和中立性。W3C通过其开放的标准制定过程，鼓励多方参与，确保网络技术的持续创新和健康发展。

让我们先简单介绍一下Web，也就是万维网的缩写。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(大部分)。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。所谓“内容”，我们指的是文本、图片和用户输入机制，如文本框和按钮，我们可以用不同的技术来处理对应的“内容”，比如：HTML（超文本标记语言）：用于定义网页内容的结构和意义；CSS（层叠样式表）：用于描述网页的外观和布局；JavaScript：一种编程语言，用于实现网页的动态功能和交互性；DOM（文档对象模型）：一个编程接口，使得JavaScript可以操作网页的内容、结构和样式；Web APIs：一系列由浏览器提供的API，如Fetch API用于数据获取、Web Storage用于客户端存储、Web Workers用于后台处理等，它们极大地丰富了Web应用的功能【2】。

Web编程是一个很大的领域，不同类型的Web编程由不同的工具实现。所有的工具都使用核心语言HTML，所以几乎所有的web编程书籍都在某种程度上描述了HTML【3】。每本教科书涵盖了HTML5, CSS和Java Script，所有的深度。这三种技术被认为是客户端web编程的支柱。使用客户端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。

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)【4】。任何开发模式则都必须经历以下四个阶段：分析（Analysis）、设计（Design）、实施（Implementation）、测试（test）。

瀑布模型为项目提供了按阶段划分的检查点，强调开发的阶段性，并且每个阶段只执行一次，因此需要专业团队的完美配合，对于普通开发者是不太现实的，作为小微开发者无法一次完美地完成任一阶段的工作，比如在实施过程中发现设计阶段存在问题，则必须在下一次迭代项目时改良设计【5】。而在增量模型中，如下图3-1所示。

|  |
| --- |
|  |
| 图3-1 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。

# 内容设计概要

## 3.1 分析和设计

这一步是项目的初次开发，本项目最初使用人们习惯的“三段论”式简洁方式开展内容设计，首先用一个标题性信息展示logo或文字标题，吸引用户的注意力，迅速表达主题；然后展现主要区域，也就是内容区，“内容为王”是项目必须坚守的理念，也是整个UI应用的重点；最后则是足部的附加信息，用来显示一些用户可能关心的细节变化。如图4-1用例图所示：

|  |
| --- |
|  |
| 图4.1 用例图 |

## 3.2 项目的实现和编程

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>

《 宁方昀的毕设题目 》

</header>

<main>

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

</main>

<footer>

CopyRight 宁方昀 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4-2 PC端项目结果



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 /d

$ mkdir Git

$ cd Git

$ cd abc

$ git init

$ git config user.name 科师大宁方昀

$ git config user.email 912384213@qq.com

$ touch index.html myCss.css

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

成功提交代码后，gitbash的反馈如下所示：

|  |
| --- |
|  |

项目代码仓库自此也开启了严肃的历史记录，我们可以输入日志命令查看，

$ git log

gitbash反馈代码的仓库日志如下所示:



# 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。

|  |
| --- |
|  |
| 图5.1 手机用户的用例图 |

## 4.2 项目的实现和编程

HTML代码编写如代码块4-1：

<header>

  <p id="book">

      我的毕设题目

    </p>

  </header>

  <nav>

    <button>导航一</button>

    <button>导航二</button>

    <button>导航三</button>

  </nav>

  <main id = 'main'>

    软件内容区域

  </main>

  <footer>

   <p id="statusInfo">

    宁方昀 @ 江西科技师范大学 2025

   </p>

  </footer>

代码块4-1

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：

<style>

   \*{

    margin: 10px;

    text-align: center;

   }

    header{

      border: 2px solid blue;

      height: 15%;

      font-size: 1.66em;

    }

    main{

      border: 2px solid blue;

      height: 70%;

      font-size: 1.2em;

    }

    nav{

      border: 2px solid blue;

      height: 10%;

         }

    nav button{

   font-size: 1.1em;

  }

    footer{

      border: 2px solid blue;

      height: 5%;

    }

  </style>

代码块 4-2

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>

    var UI = {};

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

    UI.appHeight = window.innerHeight;

  const LETTERS = 22 ;

  const baseFont = UI.appWidth / LETTERS;

  //通过更改body对象的字体大小，这个属性能够遗传其子子孙孙

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

    //通过把body对象的宽度和高度设置为设备/屏幕的宽度和高度，实现全屏。

    //通过CSS对子对象百分比（纵向）的配合，从而实现响应式设计的目标。

    document.body.style.width = UI.appWidth - 2\*baseFont + "px" ;

    document.body.style.height = UI.appHeight - 4\*baseFont + "px";

 </script>

代码块4-3

## 4.3 项目的运行和测试

本次阶段的项目运行和测试是测试在移动端设备的显示，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4-2移动端展示图如下图所示：

****

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二次开发的阶段性效果。

|  |
| --- |
|  |
| 图4-3 二维码 |

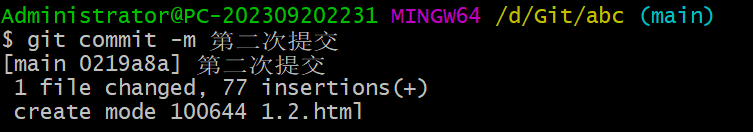
## 4.4 项目的代码提交和版本管理

编写好1.2.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 1.2.html myCss.css myjs.js

$ git commit -m 第二次提交

成功提交代码后，gitbash的反馈如下所示：



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：



# 应用响应式设计技术开发可适配窄屏和宽屏的UI

## 5.1 分析与设计

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 |

## 5.2 项目的实现和编程

HTNL代码块如下代码块5-1所示：

<body >

  <header>

    <p id="book">

     《我的毕设题目》

    </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  宁方昀 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：

<style>

  \*{

    text-align: center;

  box-sizing: border-box ;

  }

    header,main,div#bookface,nav,footer{

   margin:1em;

  }

    header{

      border: 2px solid blue;

      height: 15%;

      font-size: 1.66em;

    }

    main{

      border: 2px solid blue;

      height: 70%;

      font-size: 1.2em;

    }

    nav{

      border: 2px solid blue;

      height: 10%;

         }

    nav button{

   font-size: 1.1em;

  }

    footer{

      border: 2px solid blue;

      height: 5%;

    }

    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;

  }

</style>

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：

 <script>

    var UI = {};

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

    UI.appHeight = window.innerHeight;

  const LETTERS = 22 ;

  const baseFont = UI.appWidth / LETTERS;

  //通过更改body对象的字体大小，这个属性能够遗传其子子孙孙

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

    //通过把body对象的宽度和高度设置为设备/屏幕的宽度和高度，实现全屏。

    //通过CSS对子对象百分比（纵向）的配合，从而实现响应式设计的目标。

    document.body.style.width = UI.appWidth - 2\*baseFont + "px" ;

    document.body.style.height = UI.appHeight - 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){

    let x= ev.pageX;

    let y= ev.pageY;

    console.log("鼠标按下了，坐标为："+"("+x+","+y+")");

    $("bookface").textContent= "鼠标离开了，坐标为："+"("+x+","+y+")";

    $("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>

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5.2所示，手机端如下图5.3所示：

|  |  |
| --- | --- |
|  | 566c460245986f56912eb127faaa176 |
| 图5.2 PC端键鼠功能测试图 | 图5.3 手机端键鼠功能测试图 |

移动端用户可以通过扫描图5.4的二维码，运行测试本项目的第三次开发的阶段性效果。

|  |
| --- |
|  |
| 图5.4 移动端二维码 |

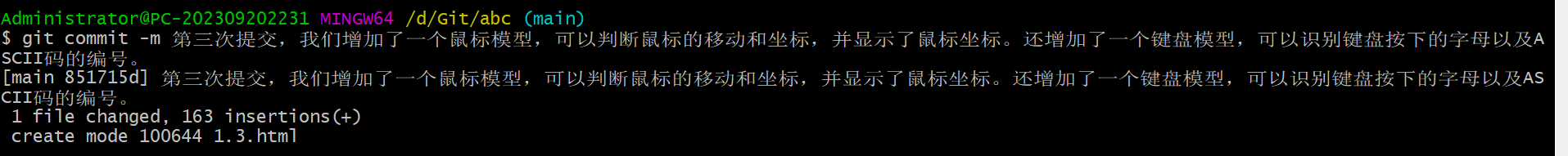
## 5.4 项目的代码提交和版本管理

编写好1.3.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 1.3.html myCss.css 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。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。

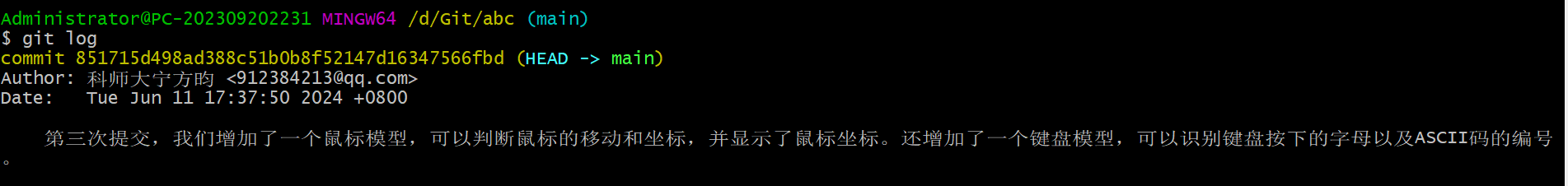
成功提交代码后，gitbash的反馈如下所示：



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：



# 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！”。

## 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：

<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">

        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 宁方昀 江西科技师范大学 2024--2025

  </footer>

  <div id="aid">

    <p>用户键盘响应区</p>

    <p id="keyboard"></p>

  </div>

代码块6-1

Css代码块下代码块6-2所示：

<style>

  \*{

    margin: 10px;

    text-align: center;

  }

  header{

    border: 3px solid green;

    height: 10%;

    font-size: 1em;

  }

  nav{

    border: 3px solid green;

    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;

    background-color: blanchedalmond;

    left:7% ;

    top: 7% ;

  }

</style>

代码块6-2

js代码块如6-3所示：

 <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>

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6.1，6.2所示：

|  |
| --- |
|  |
| 图6.1 鼠标模型拖动示例图 |

移动端用户可以通过扫描图6.2的二维码，运行测试本项目的第四次开发的阶段性效果。

|  |
| --- |
|  |
| 图6.2 移动端二维码 |

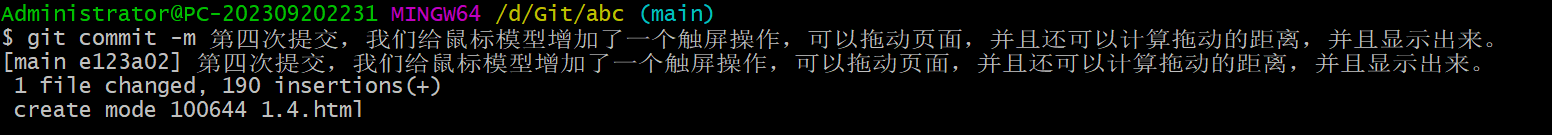
## 6.4 项目的代码提交和版本管理

编写好1.4.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 1.4.html myCss.css myjs.js

$ git commit -m 第四次提交，我们给鼠标模型增加了一个触屏操作，可以拖动页面，并且还可以计算拖动的距离，并且显示出来。

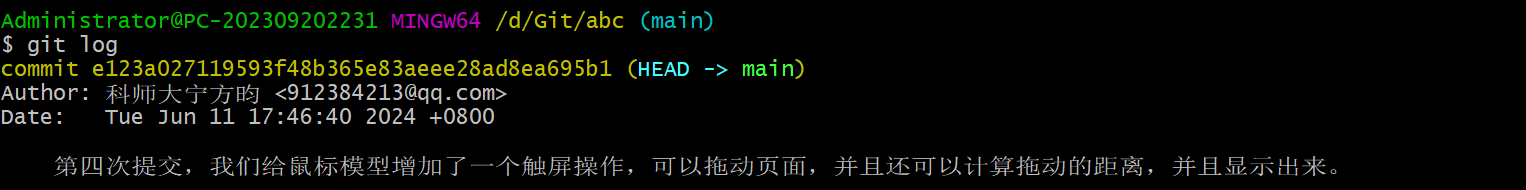
成功提交代码后，gitbash的反馈如下所示：



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：



# 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是显示拖动是否有效，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。

## 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：

<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>

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8.5所示：

|  |  |
| --- | --- |
| f6a096eeffb96307283555458ee0943 | f325c5f206023eb22385384d9da722e |
| 图7.1 显示触屏坐标测试图 | 图7.2 显示触屏拖动距离测试图 |

移动端用户可以通过扫描图6.3的二维码，运行测试本项目的第五次开发的阶段性效果。

|  |
| --- |
|  |
| 图7.3 移动端二维码 |

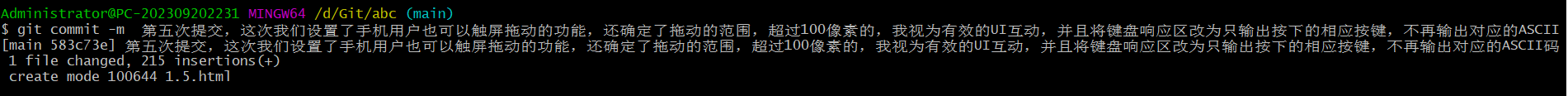
## 7.4 项目的代码提交和版本管理

编写好1.5.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

$ 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。

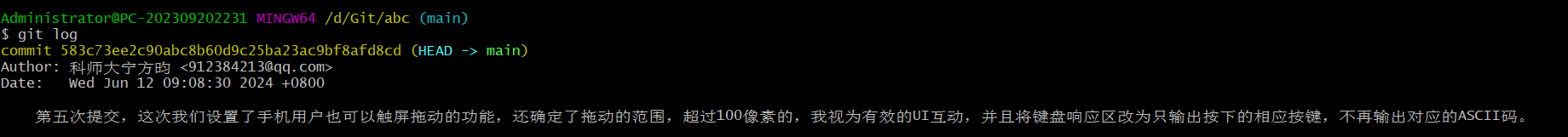
成功提交代码后，gitbash的反馈如下所示：



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：



# 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。

## 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：

<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">

        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 宁方昀 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>

   <hr>

   <p id="keyboard"></p>

  </div>

代码块8-1

Css代码块下代码块8-2所示：

<style>

  \*{

    margin: 10px;

    text-align: center;

  }

  header{

    border: 3px solid green;

    height: 10%;

    font-size: 1em;

  }

  nav{

    border: 3px solid green;

    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;

    background-color: blanchedalmond;

    left:7% ;

    top: 7% ;

  }

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：

<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\*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';

//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);

 /\*\*\*\* 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>

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8.1所示。

|  |
| --- |
|  |
| 图8.1 keydown功能测试图 |

移动端用户可以通过扫描图8.2的二维码，运行测试本项目的第六次开发的阶段性效果。

|  |
| --- |
|  |
| 图8.2 手机端二维码 |

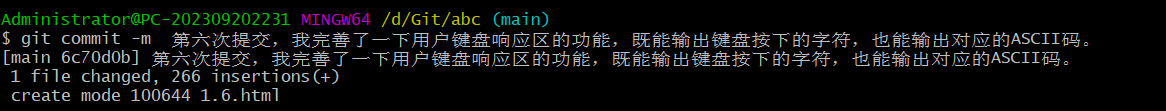
## 8.4 项目的代码提交和版本管理

编写好1.6.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 1.6.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。

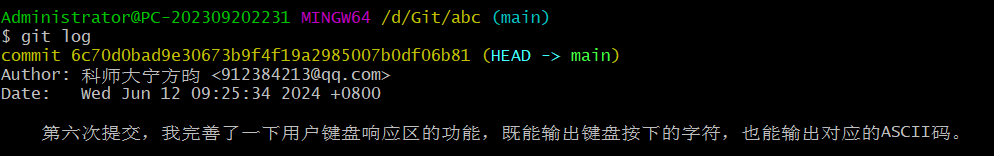
成功提交代码后，gitbash的反馈如下所示：



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：



# 谈谈本项目中的高质量代码

## 9.1编程的作用

如今，电脑虽已如螺丝刀般普及，但它们的内部机制却相当复杂，要让它们执行特定任务并非总是轻而易举。对于日常且易于理解的任务，如查看电子邮件或使用计算器功能，我们只需打开相应的应用程序即可轻松完成。然而，当面对独特或开放式任务时，可能会发现没有现成的应用程序可以直接使用。这时，编程便派上了用场。

编程，简而言之，就是构建程序的过程——这个程序由一系列精确的指令组成，旨在告诉计算机应该如何执行特定任务。由于计算机本质上是机械且死板的，编程工作往往显得单调乏味且令人沮丧。但若能克服这一挑战，并享受其中蕴含的严谨思维逻辑，那么编程的回报将是丰厚的。编程能极大地提升工作效率，使一些原本需要长时间手工完成的任务在短短几秒内即可完成。它为你提供了一种途径，让你的电脑工具能够执行以前无法实现的功能。同时，编程也是锻炼抽象思维能力的一种绝佳方式。

## 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。

以下是实现代码：

|  |
| --- |
|  |
| 图9.1 用Pointer对象实现鼠标模型 |

|  |
| --- |
|  |
| 图9.2 鼠标拖动以及触屏操作代码详情图 |

|  |
| --- |
|  |
| 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**

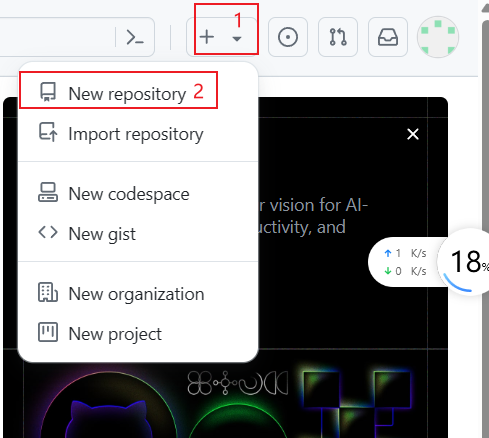
当我们提及命令行界面时，我们实质上是在谈论shell。shell是一个程序，它负责接收来自键盘输入的命令，并将其传递给操作系统进行执行。在Linux的众多发行版中，一个常见的shell程序来自GNU项目，名为bash。bash这个名字其实是Bourne-again shell的缩写，它是对sh（即Steve Bourne编写的原始Unix shell）的增强版。

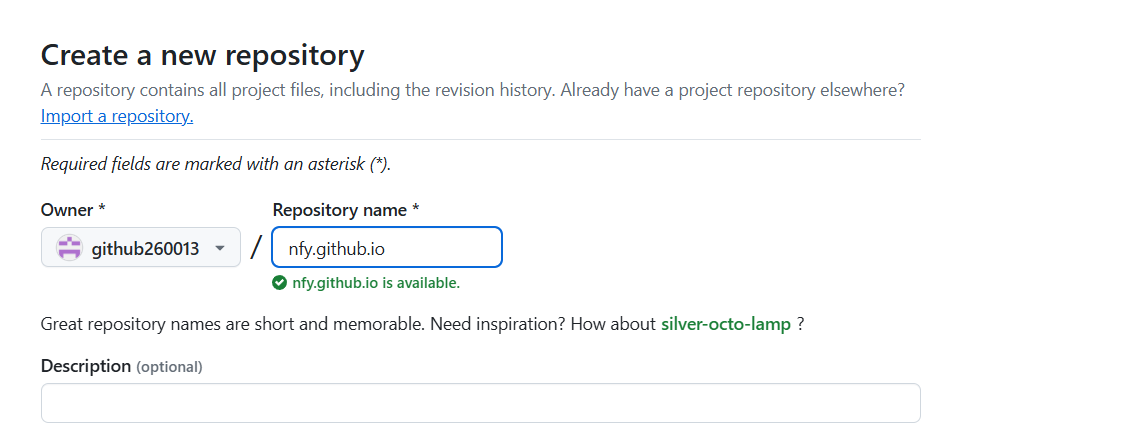
与Windows系统类似，类Unix操作系统如Linux，采用了一种层次化的目录结构来组织文件。这种结构呈现为树状模式（在其他系统中可能称为文件夹），每个节点都可能包含文件或其他子目录。文件系统的起点被称为根目录，它包含文件和子目录，而这些子目录又可能包含更多的文件和子目录，以此类推，形成了一种层级分明的文件组织方式。

## 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。







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

$ git branch -M main

将当前分支重命名为main

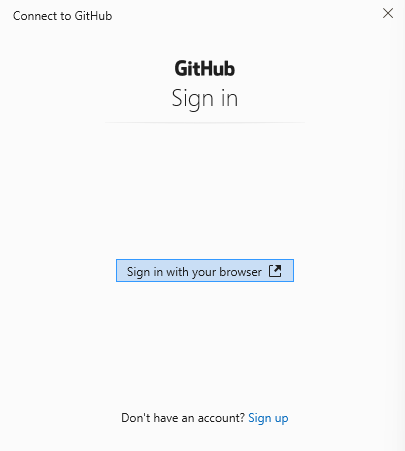
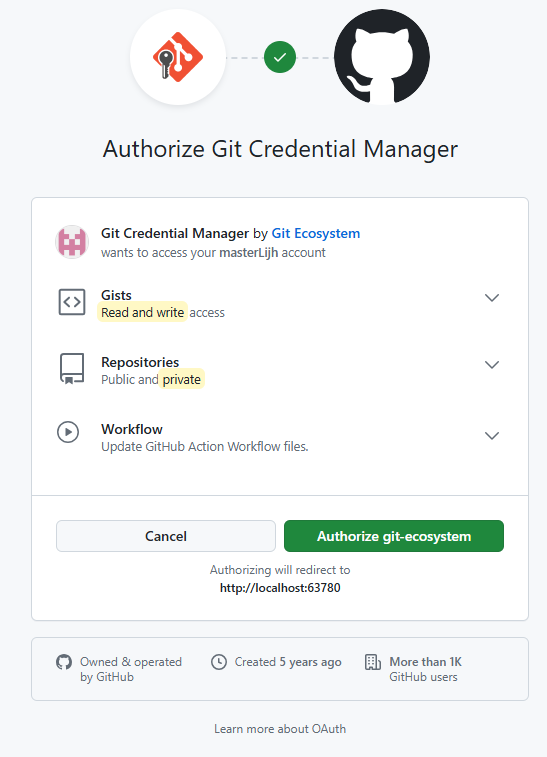
$ git remote add origin

将本地git仓库和远程github仓库链接起来,并将其命名为origin

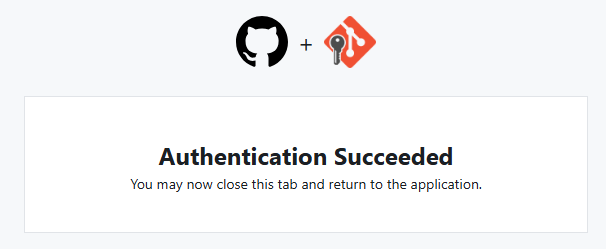
$ git push -u origin main

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：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.

# 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