

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 的设计和编程

（外文）：Customized UI design and Programming

based on Web client technology

院（系）：元宇宙产业学院

专 业：计算机科学与技术

学生姓名：刘强

学 号：20213606

指导教师：李健宏

2024 年 6 月 19 日

基于 Web 客户端技术的个性化 UI 的设计和编程

摘要: 本项目聚焦于 Web 客户端技术的学习与研究，通过广泛查阅技术资料和相关文献，特别是 MDN 社区的技术实践文章，深入探讨了 HTML、CSS 和 JavaScript 的基本技术和技巧。基于这些技术，成功构建了一个响应式用户界面，该界面能够自适应不同设备的屏幕尺寸，无论是 PC 端还是移动端设备。在功能实现方面，本项目采用了 DOM 技术和事件驱动编程，实现了对鼠标、触屏和键盘事件的响应与处理，并创新性地为鼠标和触屏设计了一个对象模型，通过代码模拟了这些指向性设备的行为。为了更好地管理项目的设计与开发，项目采用了软件工程的增强式开发模式，并进行了 6 次迭代开发，每次迭代都遵循了软件的 4 个经典开发过程（A:Analysis, D:Design, I: Implementation, T:Testing）。项目还使用了 Git 进行版本控制和开发日志记录，共进行了 12 次代码提交，详细记录了开发过程和代码优化。最终，项目通过 GitBash 上传到 GitHub，并设置为 HTTP 服务器，使得 UI 应用能够被全球用户便捷访问。

关键词: Web 客户端技术；DOM 技术；迭代开发；UI 应用

Personalized UI based on web client technology Design and programming

Abstract: This project focuses on the study and research of web client technologies, and delves into the basic techniques and techniques of HTML, CSS, and JavaScript through extensive review of technical materials and related literature, especially technical practice articles from the MDN community. Based on these technologies, a responsive user interface was successfully built that adapts to the screen size of different devices, whether PC or mobile. In terms of functional implementation, the project uses DOM technology and event-driven programming to realize the response and processing of mouse, touch screen and keyboard events, and innovatively designs an object model for mouse and touch screen, and simulates the behavior of these directional devices through code. In order to better manage the design and development of the project, the project adopted the enhanced development model of software engineering, and carried out 6 iterations of development, each iteration followed the 4 classic software development processes (A:Analysis, D:Design, I:Implementation, T:Testing). The project also used Git for version control and development logging, with a total of 12 code commits detailing the development process and code optimization. Eventually, the project was uploaded to GitHub via GitBash and set up as an HTTP server, making the UI application easily accessible to users around the world.

Keywords: Web client technology; DOM technology; iterative development; UI Apps

1. 前言

在当今信息化时代，Web 应用已经成为我们日常生活和工作的基本组成部分。随着移动互联网的兴起和用户需求的多样化，Web 技术也在不断进步。HTML5 作为新一代的 Web 标准，因其强大功能和跨平台特性而得到了广泛应用。它不仅提供了丰富的标签和 API，还通过 CSS3 和 JavaScript 实现了灵活和动态的用户界面设计，使得开发者能够创造出功能强大、美观且兼容性高的 Web 应用。

HTML5 的跨平台能力是其一大优势，它允许 Web 应用在不同的设备和操作系统上无缝运行，极大地降低了开发成本，提高了开发效率。CSS3 带来了更加丰富的样式设计和动画效果，使得 Web 应用在视觉上也能给予用户愉悦的体验。JavaScript 则负责实现交互逻辑，使得用户界面能够响应用户的操作，提供动态内容，增强用户体验。随着技术的不断发展，HTML5、CSS3 和 JavaScript 的组合已经成为现代 Web 开发的标准技术栈，为开发人员提供了构建现代化、高性能 Web 应用的强大工具。

1.1 项目概要

个性化 UI 设计是现代 Web 开发中提升用户体验的关键。它通过响应式设计适应不同设备和屏幕尺寸，根据用户偏好和行为调整界面，提供一致且流畅的交互体验。这包括用户研究、数据分析、交互设计、动态内容适配、主题定制、智能推荐、可访问性、性能优化和持续迭代等方面。个性化 UI 设计能够满足用户对个性化和定制化服务的需求，提升用户满意度和忠诚度。

本次课程设计的目标是利用 HTML5、CSS3、JavaScript 等现代 Web 技术设计和实现一个具有个性化用户界面（UI）的应用程序。该界面将支持鼠标、键盘和触屏操作，为用户提供无缝的使用体验。通过动态调整 UI 布局和交互方式，实现更加个性化的用户体验。同时，该项目将为开发人员提供一个参考案例，展示如何利用先进 Web 技术实现高质量、自适应且个性化的用户界面。为确保代码质量和可维护性，我们采用了面向对象的设计思想和响应式设计方法，以适应不同设备屏幕的需求。在整个开发过程中，我们亲手编写每一行代码，深入理解 Web 客户端技术的核心原理。项目采用增量式开发模式，经过多次迭代和重构，

不断优化代码，实现了设计、开发和测试的有机结合。为了更好地进行版本控制和代码管理，我们使用 Git 工具，将项目托管在 GitHub 平台上，并通过 GitHub 的 HTTP 服务器实现全球互联网部署，使用户能够便捷地访问和体验该程序。

本论文将详细介绍本次课程设计的技术路线、开发过程、功能实现及代码管理方法。

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 git bash 工具进行项目提交，并上传到 git hub 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，通过研究相关的 Web 客户端技术和个性化 UI 设计的文献，确定用户在移动互联时代对个性化 UI 设计的需求和期望。接着，分析项目的可行性，并选择迭代增量式开发模型作为软件开发的方法。这种模型将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），并在后续迭代中逐步完善和扩展这个部分。

在开发过程中，基于定性和定量研究所得数据，进行统计分析和数据挖掘，根据分析结果，通过迭代增量式开发模型逐步实现移动互联时代的个性化 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。

第一章 技术总结和文献综述

(一) 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。

(二) 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。

The incremental model

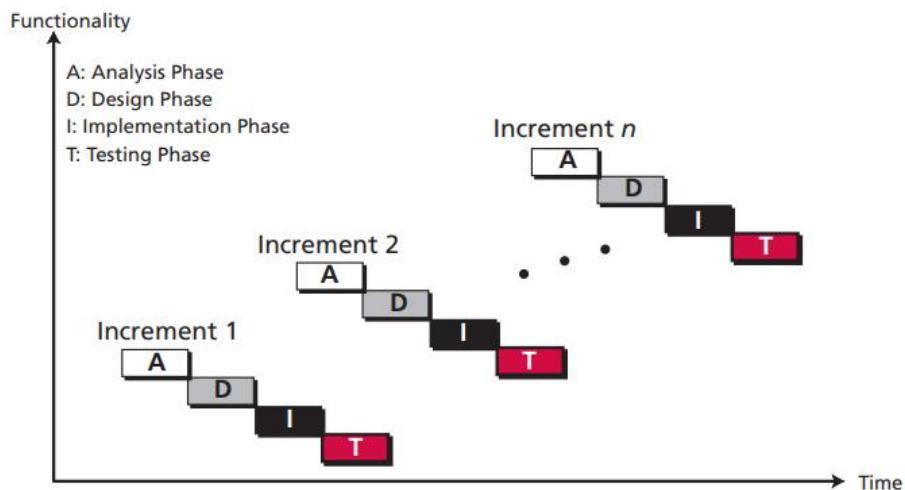


图 3-1 增量迭代开发模式

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。

第二章 内容设计概要

(三) 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。

(四) 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 代码编写如代码截图 3-1：

代码截图 3-1

CSS 代码编写如代码截图 3-2：

```
1  *{  
2      margin: 10px;  
3      text-align: center;  
4      font-size:30px ;  
5  }  
6  header{  
7      border: 2px solid blue;  
8      height: 200px;  
9  }  
10  
11 main{  
12     border: 2px solid blue;  
13     height: 400px;  
14  }  
15  
16 footer{  
17     border: 2px solid blue;  
18     height: 100px;  
19  }  
20 a{  
21     display: inline-block ;  
22     padding:10px ;  
23     color: white;  
24     background-color: blue;  
25     text-decoration: none ;  
26  }
```

代码截图 3-2

(五) 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 4-1 手机端项目结果

图 4-2 PC 端项目结果

二维码

(六) 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：如开发者的姓名和电子邮件。进入 gitBash 命令行后，按次序输入以下命令：

```
86182@LAPTOP-SOVINFHE MINGW64 /  
$ mkdir WebUI  
  
86182@LAPTOP-SOVINFHE MINGW64 /  
$ cd WebUI  
  
86182@LAPTOP-SOVINFHE MINGW64 /WebUI  
$ git init  
Initialized empty Git repository in D:/Git/WebUI/.git/  
  
86182@LAPTOP-SOVINFHE MINGW64 /WebUI (master)  
$ git config user.name 刘强@科师大  
  
86182@LAPTOP-SOVINFHE MINGW64 /WebUI (master)  
$ git config user.email 3252386238@qq.com
```

测试运行成功后，执行下面命令提交代码：

```
Reo1@Rrreol666 MINGW64 /d/nHQ_md_repository/NEW_repository (master)  
$ git commit -m 第一次提交，我们完成了软件的设计概要  
On branch master
```

成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m 第一次提交，我们完成了软件的设计概要  
[main 88dffdc] 第一次提交，我们完成了软件的设计概要  
 1 file changed, 52 insertions(+), 44 deletions(-)  
 rewrite index.html (82%)
```

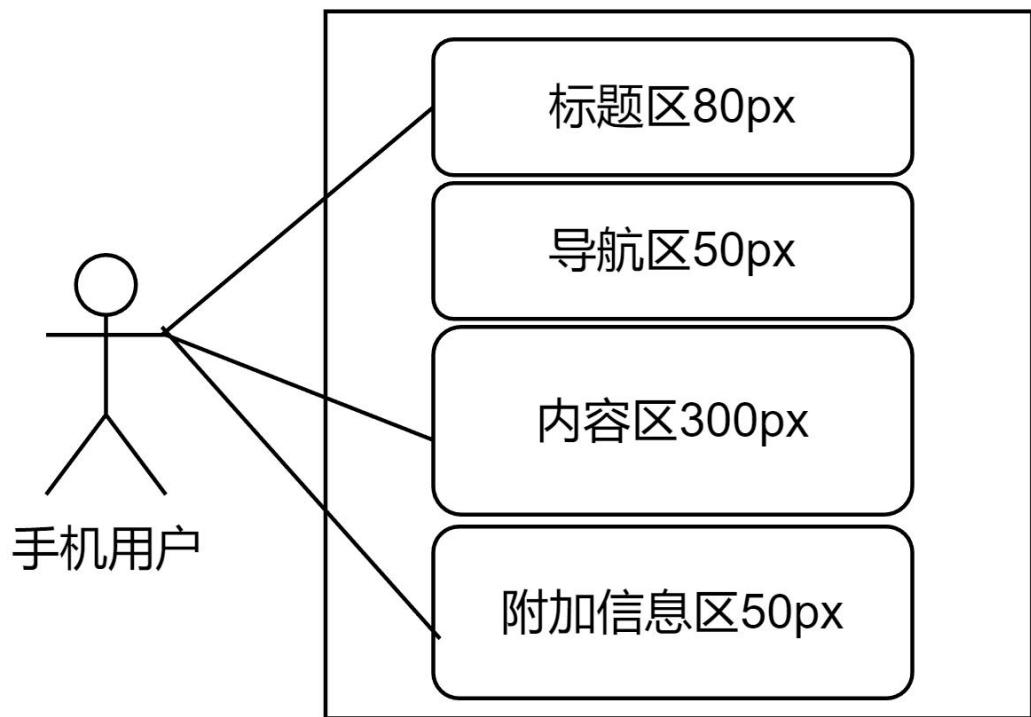
项目代码仓库自此也开启了严肃的历史记录，我们可以输入日志命令查看，

```
$ git log
```

第三章 窄屏终端的响应式设计

二、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。用例图如下：



(一) 4.2 项目的实现和编程

HTML 代码编写如代码截图 4-1:

代码截图 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 4-2 所示：

```
28 <style>
29  *{
30   margin: 10px;
31   text-align: center;
32 }
33
34 header{
35   border: 2px solid blue;
36   height: 15%;
37   font-size: 1.66em;
38 }
39
40 main{
41   border: 2px solid blue;
42   height: 70%;
43   font-size: 1.2em;
44 }
45
46 nav{
47   border: 2px solid blue;
48   height: 10%;
49   }
50   nav button{
51   font-size: 1.1em;
52   }
53 footer{
54   border: 2px solid blue;
55   height: 5%;
56   }
57 </style>
```

代码截图 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 代码截图 4-3 所示：

```
1 <script>
2     var UI = {};
3     UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;
4     UI.appHeight = window.innerHeight;
5     const LETTERS = 22 ;
6     const baseFont = UI.appWidth / LETTERS;
7
8 //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
9     document.body.style.fontSize = baseFont + "px";
10    //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。
11    //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。
12    document.body.style.width = UI.appWidth - 2*baseFont + "px" ;
13    document.body.style.height = UI.appHeight - 4*baseFont + "px";
14 </script>
```

代码截图 4-3

(二) 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

第四章 应用响应式设计技术开发可适配窄屏和宽屏的 UI

(三) 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：

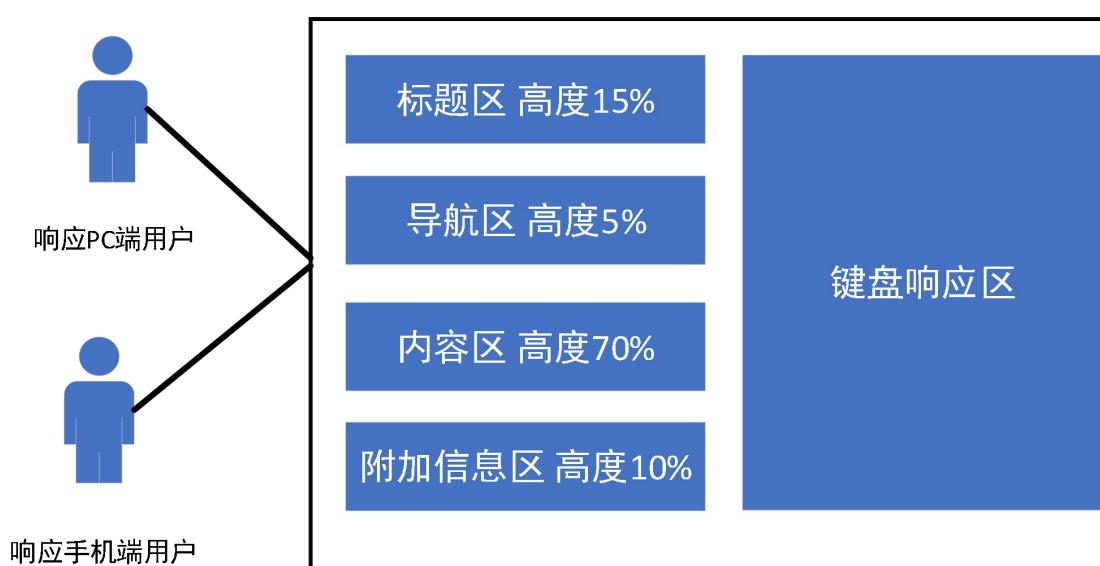


图 5.1 用例图

(四) 5.2 项目的实现和编程

HTNL 代码截图如下代码截图 5-1 所示:

代码截图 5-1

Css 代码截图下代码截图 5-2 所示：

```
1 <style>
2   *{
3     text-align: center;
4     box-sizing: border-box ;
5   }
6   header,main,div#bookface,nav,footer{
7     margin:1em;
8   }
9   header{
10     border: 2px solid blue;
11     height: 15%;
12     font-size: 1.66em;
13   }
14   main{
15     border: 2px solid blue;
16     height: 70%;
17     font-size: 1.2em;
18   }
19   }
20   nav{
21     border: 2px solid blue;
22     height: 10%;
23   }
24   }
25   nav button{
26     font-size: 1.1em;
27   }
28   footer{
29     border: 2px solid blue;
30     height: 5%;
31   }
32   body{
33     position:relative ;
34   }
35   #aid{
36     position: absolute;
37     border: 3px solid blue;
38     top: 0.5em;
39     left: 600px;
40   }
41   #bookface{
42     width: 80%;
43     height: 80%;
44     border:1px solid red;
45     background-color: blanchedalmond;
46     margin:auto;
47   }
48 </style>
49
```

代码截图 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

码截图 5-3 所示：

```
1 <script>
2     var UI = {};
3     UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;
4     UI.appHeight = window.innerHeight;
5     const LETTERS = 22 ;
6     const baseFont = UI.appWidth / LETTERS;
7
8     //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
9     document.body.style.fontSize = baseFont + "px";
10    //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。
11    //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。
12    document.body.style.width = UI.appWidth - 2*baseFont + "px" ;
13    document.body.style.height = UI.appHeight - 8*baseFont + "px";
14
15    if(window.innerWidth < 900){
16        $("aid").style.display='none';
17    }
18    $("aid").style.width=window.innerWidth - UI.appWidth - 2*baseFont + 'px';
19    $("aid").style.height= document.body.clientHeight + 'px';
20
21    //尝试对鼠标设计UI控制
22    var mouse={};
23    mouse.isDown= false;
24    mouse.x= 0;
25    mouse.deltaX=0;
26    $("bookface").addEventListener("mousedown",function(ev){
27        let x= ev.pageX;
28        let y= ev.pageY;
29
30        console.log("鼠标按下了， 坐标为: "+"("+x+","+y+")");
31        $("bookface").textContent= "鼠标按下了， 坐标为: "+"("+x+","+y+")";
32    });
33
34    $("bookface").addEventListener("mousemove",function(ev){
35        let x= ev.pageX;
36        let y= ev.pageY;
37
38        console.log("鼠标正在移动， 坐标为: "+"("+x+","+y+")");
39        $("bookface").textContent= "鼠标正在移动， 坐标为: "+"("+x+","+y+")";
40    });
41    $("bookface").addEventListener("mouseout",function(ev){
42        let x= ev.pageX;
43        let y= ev.pageY;
44
45        console.log("鼠标按下了， 坐标为: "+"("+x+","+y+")");
46        $("bookface").textContent= "鼠标离开了， 坐标为: "+"("+x+","+y+")";
47        $("bookface").textContent="鼠标已经离开";
48    });
49    $("body").addEventListener("keypress",function(ev){
50        let k = ev.key;
51        let c = ev.keyCode;
52        $("keyboard").textContent = "您的按键 : " + k + " , "+ "字符编码 : " + c;
53    });
54
55    function $(ele){
56        if (typeof ele !== 'string'){
57            throw("自定义的$函数参数的数据类型错误，实参必须是字符串！");
58            return
59        }
60        let dom = document.getElementById(ele) ;
61        if(dom){
62            return dom ;
63        }else{
64            dom = document.querySelector(ele) ;
65            if (dom) {
66                return dom ;
67            }else{
68                throw("执行$函数未能在页面上获取任何元素，请自查问题！");
69            }
70        }
71    }
72 } //end of $
```

代码截图 5-3

(五) 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：

图 5.2 PC 端键鼠功能测试图

图 5.3 手机端触碰功能测试图

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

第六章 个性化 UI 设计中对鼠标交互的设计开发

(六) 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！”。

(七) 6.2 项目的实现和编程

以下代码截图是对 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 代码编写如代码截图 6-1：

代码截图 6-1

Css 代码截图 6-2 所示：

```
1 <style>
2   *{
3     margin: 10px;
4     text-align: center;
5   }
6   header{
7     border: 3px solid green;
8     height: 10%;
9     font-size: 1em;
10  }
11 }
12 nav{
13   border: 3px solid green;
14   height: 10%;
15 }
16 main{
17   border: 3px solid green;
18   height: 70%;
19   font-size: 0.8em;
20   position: relative;
21 }
22
23 #box{
24   position: absolute;
25   right: 0;
26   width: 100px;
27 }
28
29 footer{
30   border: 3px solid green;
31   height:10%;
32   font-size: 0.7em;
33 }
```

```
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>
```

代码截图 6-2

js 代码截图 6-3 所示

```
1 <script>
2 var UI = {};
3 if(window.innerWidth>600){
4     UI.appWidth=600;
5 }else{
6     UI.appWidth = window.innerWidth;
7 }
8
9 UI.appHeight = window.innerHeight;
10
11 let baseFont = UI.appWidth /20;
12 //通过改变body对象的字体大小，这个属性可以影响其后代
13 document.body.style.fontSize = baseFont +"px";
14 //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
15 //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
16 document.body.style.width = UI.appWidth - baseFont + "px";
17 document.body.style.height = UI.appHeight - baseFont*4 + "px";
18 if(window.innerWidth<1000){
19     $("aid").style.display='none';
20 }
21 $("aid").style.width=window.innerWidth-UI.appWidth - baseFont*3 +'px';
22 $("aid").style.height= UI.appHeight - baseFont*3 +'px';
23
24 //尝试对鼠标设计UI控制
25 var mouse={};
26 mouse.isDown= false;
27 mouse.x= 0;
28 mouse.y= 0;
29 mouse.deltaX=0;
30 $("bookface").addEventListener("mousedown",function(ev){
31     mouse.isDown=true;
32     mouse.x= ev.pageX;
33     mouse.y= ev.pageY;
34     console.log("mouseDown at x: "+("+"+mouse.x +"," +mouse.y +"") );
35     $("bookface").textContent= "鼠标按下，坐标: "+("+"+mouse.x+"," +mouse.y+"");
36 });

});
```

```

});`  

$("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' ;  

    }  

});`  

70});`  

71`  

72`  

73 function $(ele){  

74     if (typeof ele !== 'string'){  

75         throw("自定义的$函数参数的数据类型错误，实参必须是字符串！");  

76         return  

77     }  

78     let dom = document.getElementById(ele) ;  

79     if(dom){  

80         return dom ;  

81     }else{  

82         dom = document.querySelector(ele) ;  

83         if (dom) {  

84             return dom ;  

85         }else{  

86             throw("执行$函数未能在页面上获取任何元素，请自查问题！");  

87             return ;  

88         }  

89     }  

90 } //end of $  

91`  

92`</script>

```

代码截图 6-3

(八) 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 所示：

图 6.1 鼠标模型拖动示例图

移动端用户示例如图 6.2 所示

图 6.2

可见，由于该次项目针对 pc 端，可见在移动用户端并不能有效拖动。

二维码如图 6.3 所示：

图 6.3 项目二维码

第七章 对触屏和鼠标的通用交互操作的设计开发

(九) 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。

(十) 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 代码如代码截图 7-1 所示：

```

1 <script>
2   var UI = {};
3   if(window.innerWidth>600){
4     UI.appWidth=600;
5   }else{
6     UI.appWidth = window.innerWidth;
7   }
8
9   UI.appHeight = window.innerHeight;
0
1. let baseFont = UI.appWidth /20;
2 //通过改变body对象的字体大小，这个属性可以影响其后代
3 document.body.style.fontSize = baseFont +"px";
4 //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
5 //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
6 document.body.style.width = UI.appWidth - baseFont + "px";
7 document.body.style.height = UI.appHeight - baseFont*4 + "px";
8 if(window.innerWidth<1000){
9   $("aid").style.display='none';
10 }
11 $("aid").style.width=window.innerWidth-UI.appWidth - baseFont*3 +'px';
12 $("aid").style.height= UI.appHeight - baseFont*3 +'px';
13
14 //尝试对鼠标和触屏设计一套代码实现UI控制
15 var Pointer = {};
16 Pointer.isDown= false;
17 Pointer.x = 0;
18 Pointer.deltaX =0;
19 { //Code Block begin
20   let handleBegin = function(ev){
21     Pointer.isDown=true;
22
23     if(ev.touches){console.log("touches1"+ev.touches);
24       Pointer.x = ev.touches[0].pageX ;
25       Pointer.y = ev.touches[0].pageY ;
26       console.log("Touch begin : "+("."+Pointer.x +"," +Pointer.y +""));
27       $("bookface").textContent= "触屏事件开始，坐标: "+("."+Pointer.x+"," +Pointer.y+"");
28     }else{
29
30     Pointer.x= ev.pageX;
31     Pointer.y= ev.pageY;
32     console.log("PointerDown at x: "+("."+Pointer.x +"," +Pointer.y +""));
33     $("bookface").textContent= "鼠标按下，坐标: "+("."+Pointer.x+"," +Pointer.y+"");
34   }
35 };
36 let handleEnd = function(ev){
37   Pointer.isDown=false;
38   ev.preventDefault();
39   //console.log(ev.touches)
40   if(ev.touches){
41     $("bookface").textContent= "触屏事件结束!";
42     if(Math.abs(Pointer.deltaX) > 100){
43       $("bookface").textContent += "，这是有效触屏滑动！" ;
44     }else{
45       $("bookface").textContent += " 本次算无效触屏滑动！" ;
46     }
47   }else{
48     $("bookface").textContent= "鼠标松开!";
49     if(Math.abs(Pointer.deltaX) > 100){
50       $("bookface").textContent += "，这是有效拖动！" ;
51     }else{
52       $("bookface").textContent += " 本次算无效拖动！" ;
53     }
54   }
55 }
56
57 let handleMoving = function(ev){
58   ev.preventDefault();
59   if (ev.touches){
60     if (Pointer.isDown){
61       console.log("Touch is moving");
62       Pointer.deltaX = parseInt( ev.touches[0].pageX - Pointer.x );
63       $("bookface").textContent= "正在滑动触屏，滑动距离: " + Pointer.deltaX +"px 。";
64       $('bookface').style.left =  Pointer.deltaX + 'px' ;
65     }
66   }
67 }
68
69
70
71
72
73
74
75

```

```

76     }
77 }else{
78     if (Pointer.isDown){
79         console.log("Pointer isDown and moving");
80         Pointer.deltaX = parseInt( ev.pageX - Pointer.x );
81         $("bookface").textContent= "正在拖动鼠标, 距离: " + Pointer.deltaX +"px 。";
82         $('bookface').style.left =  Pointer.deltaX + 'px' ;
83     }
84 }
85 };
86
87 $("bookface").addEventListener("mousedown",handleBegin );
88 $("bookface").addEventListener("touchstart",handleBegin );
89 $("bookface").addEventListener("mouseup", handleEnd );
90 $("bookface").addEventListener("touchend",handleEnd );
91 $("bookface").addEventListener("mouseout", handleEnd );
92 $("bookface").addEventListener("mousemove", handleMoving);
93 $("bookface").addEventListener("touchmove", handleMoving);
94 $("body").addEventListener("keypress", function(ev){
95     $(".aid").textContent += ev.key ;
96 });
97 } //Code Block end
98 function $(ele){
99     if (typeof ele !== 'string'){
100         throw("自定义$函数参数的数据类型错误, 实参必须是字符串! ");
101         return
102     }
103     let dom = document.getElementById(ele) ;
104     if(dom){
105         return dom ;
106     }else{
107         dom = document.querySelector(ele) ;
108         if (dom) {
109             return dom ;
110         }else{
111             throw("执行$函数未能在页面上获取任何元素, 请自查问题! ");
112             return ;
113         }
114     }
115 } //end of $
116
117 </script>

```

代码截图 7-1

(十一) 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 7.1 7.2 所示：

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 7.3 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

第八章 UI 的个性化键盘交互控制的设计开发

(十二) 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。

(十三) 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 代码编写如代码截图 8-1 所示：

代码截图 8-1

Css 代码截图下代码截图 8-2 所示：

```
1 <style>
2   *{
3     margin: 10px;
4     text-align: center;
5   }
6   header{
7     border: 3px solid green;
8     height: 10%;
9     font-size: 1em;
10  }
11 }
12 nav{
13   border: 3px solid green;
14   height: 10%;
15 }
16 main{
17   border: 3px solid green;
18   height: 70%;
19   font-size: 0.8em;
20   position: relative;
21 }
22
23 #box{
24   position: absolute;
25   right: 0;
26   width: 100px;
27 }
28
29 footer{
30   border: 3px solid green;
31   height:10%;
32   font-size: 0.7em;
33 }
```

```
--  
34   body{  
35     position: relative;  
36   }  
37   button{  
38     font-size:1em;  
39   }  
40   #aid{  
41     position: absolute;  
42     border: 3px solid blue;  
43     top:0px;  
44     left:600px;  
45   }  
46   #bookface{  
47     position: absolute;  
48     width: 80%;  
49     height: 80%;  
50     border:1px solid red;  
51     background-color: blanchedalmond;  
52     left:7% ;  
53     top: 7% ;  
54   }  
55 }  
56 }
```

代码截图 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 代码截图如代码截图 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);

$("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)){
        $("typeText").textContent += key ;
    }
    function printLetter(k){
if (k.length > 1){
    return false ;
}
let puncs = ['~','`','!','@','#','$','%','^','&','*','(',')','-','_','+','=','`','.',','<','>','?','/','`'];
    if ( (k >= 'a' && k <= 'z')|| (k >= 'A' && k <= 'Z')|| (k >= '0' && k <= '9')) {
        console.log("letters") ;
        return true ;
    }
for (let p of puncs ){
    if (p === k) {
        console.log("puncs") ;
        return true ;
    }
}
return false ;
}
});
}

```

```
        }
    }
function $(ele){
    if (typeof ele !== 'string'){
        throw("自定义的$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 ;
        }
    }
}

</script>
```

代码截图 8-3

(十四) 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

第九章 谈谈本项目中的高质量代码

(十五) 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。

(十六) 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：

```
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

```
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

(十七) 10.1 经典 Bash 工具介绍

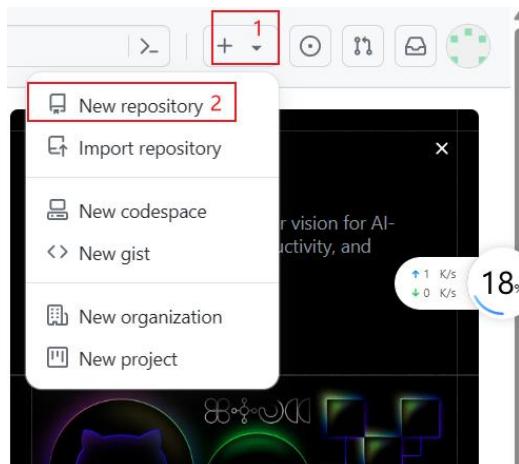
当我们谈到命令行时，我们实际上指的是 shell。shell 是一个接受键盘命令并将其传递给操作系统执行的程序。几乎所有的 Linux 发行版都提供了一个来自 GNU 项目的 shell 程序，名为 bash。这个名字是 Bourne -again shell 的首字母缩略词，bash 是 sh 的增强替代品，sh 是 Steve Bourne 编写的原始 Unix shell 程序。

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。

(十八) 10.2 通过 GitHub 平台实现本项目的全球域名

1. 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 *

 Rrreal-love /

Great repository names are short and memorable. Need inspiration? How about **bug-free-octo-disco** ?

Description (optional)

 **Public**
Anyone on the internet can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

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。

(十九) 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

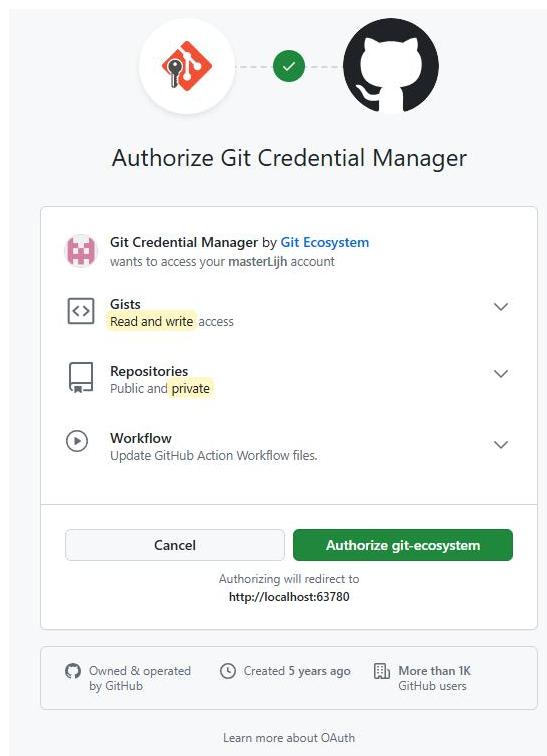
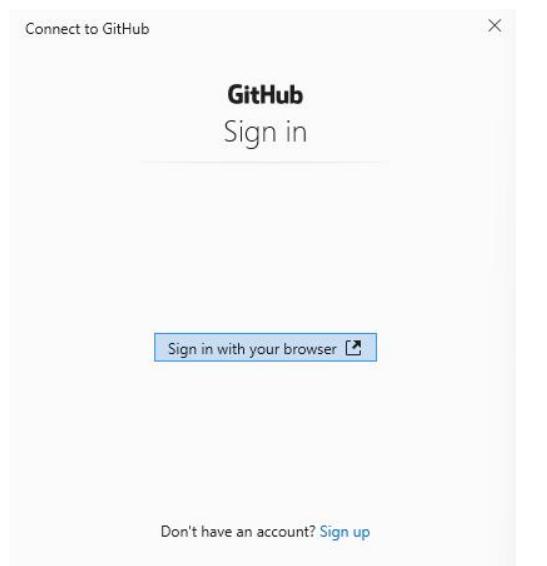
```
$ git remote add origin
```

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

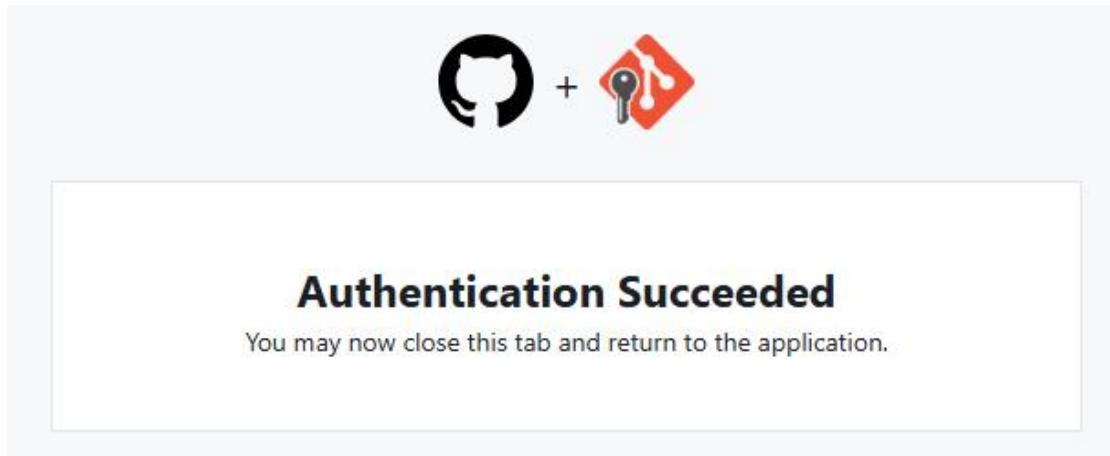
```
$ git push -u origin main
```

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。

三、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