

# 江西科技师范大学

## 毕业论文（设计）

题目（中文）：基于 Web 客户端技术的个性化 UI 设计和实现

（外文）：Web client based customized UI design and Programming

院（系）：元宇宙产业学院

专    业：计算机科学与技术

学生姓名：郭理靖

学    号：20213599

指导教师：李健宏

2024 年 5 月 23 日

# 基于 Web 客户端技术的个性化 UI 设计和实现

## 目录

第 1 章 引言.....	1
1.1 学习计划.....	1
1.2 项目技术路线.....	1
1.3 参考资料.....	1
1.4 研究方法.....	1
第 2 章 Web 平台和客户端技术概述.....	2
第 3 章 软件开发的过程管理——增量式开发模式.....	3
第 4 章 移动互联时代的响应式设计和窄屏代码实现.....	4
4.1 项目分析和设计.....	4
4.2 项目的实现和编程.....	5
4.3 项目的运行和测试.....	7
4.4 项目代码的提交和版本管理.....	8
第 5 章 宽屏和窄屏通用的响应式设计和代码实现.....	10
5.1 项目设计与分析.....	10
5.2 项目的实现和编程.....	11
5.3 项目的运行和测试.....	14
第 6 章 个性化 UI 设计中鼠标模型.....	15
6.1 设计与分析.....	15
6.2 项目的实现和编程.....	16
6.3 项目的运行和测试.....	19
6.4 项目的代码提交和版本管理.....	21
第 7 章 通用的 UI 设计，用一套代码同时为触屏和鼠标建模.....	21
7.1 分析和设计.....	21
7.2 项目的实现和编程.....	22
7.3 项目的运行和测试.....	25
7.4 项目的代码提交和版本管理.....	26
第 8 章 UI 的个性化键盘控制——应用 keydown 和 keyup 键盘底层事件.....	27
8.1 分析与设计.....	27
8.2 项目的实现和编程.....	27
8.3 项目的运行和测试.....	33
8.4 项目的代码提交和版本管理.....	34
第 9 章 谈谈本项目中的高质量代码.....	34
9.1 编程的作用.....	34
9.2 项目的高质量代码.....	35

第 10 章 用 git 工具展开代码的版本管理和发布.....	37
参考文献.....	41

**摘要:** Web 平台以其跨操作系统平台的优势成为了广泛流行的软件开发技术, 本项目以 Web 客户端技术为研究学习对象, 通过查阅相关技术资料 and 文献, 特别是 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。实现了一个个性化的用户界面 (UI: user interface), 该用户界面可以动态适用于 PC 端和移动端设备, 以响应式技术为支撑做到了最佳适配用户屏幕, 以 DOM 技术和事件驱动模式的程序为支撑实现了对鼠标、触屏、键盘的底层事件响应和流畅支持, 为鼠标和触屏设计了一个对象模型, 用代码实现了对这类指向性设备的模拟, 这是本项目模型研究法的一次创新实践, 也是本项目的亮点。为了处理好设计和开发的关系, 用工程思想管理项目, 本项目使用了软件工程的增量式增强的开发模式, 一共做了 6 次迭代开发, 每次迭代都包含了软件的 4 个经典开发过程: ADIT (A: 分析 D: 设计 I: 实施 T: 测试), 逐步实现了整个 UI 应用编写。最后, 为了与互联网上的其他开发者合作, 本项目还使用了 git 工具进行代码和开发过程日志记录, 一共做了 6 次提交代码的操作, 详细记录和展现了开发思路和代码优化的路径, 通过 git bash 把项目上传到 github 上, 建立了自己的代码仓库, 并将该代码仓库设置成为了 HTTP 服务器, 实现了本 UI 应用的全球便捷访问。

**关键字:** Web 技术; git 工具; HTML 建模; ADIT 开发过程;

## Web client based customized UI design and Programming

**Abstract:** Web platform has become a widely popular software development technology because of its advantages across operating system platforms. This project takes Web client technology as the research and learning object, through consulting relevant technical materials and literature, especially the technical practice articles of the MDN community organized by Mozilla. Explore basic techniques and techniques for HTML content modeling, CSS style design, and JavaScript functional programming. Implement a personalized user interface (UI: The user interface can be dynamically applied to both PC and mobile devices, and is best adapted to the user screen with the support of responsive technology. The underlying event response and smooth support for mouse, touch screen and keyboard are realized with the support of DOM technology and event-driven program. An object model is designed for mouse and touch screen. The simulation of this kind of directional device is realized by code, which is an innovative practice of the model research method in this project and also the highlight of this project. In order to deal with the relationship between design and development and manage the project with engineering ideas, this project uses the incremental enhancement development mode of software engineering, and has done six iterations of development, each iteration includes four classic software development processes: ADIT (A: analysis D: design I: implementation T: test), and gradually realizes the entire UI application writing. Finally, in order to cooperate with other developers on the Internet, this project also used git tools to log the code and development process, and made 6 code submission operations, recorded and displayed the development ideas and code optimization paths in detail, uploaded the project to github through git bash, and established its own code warehouse. And set the code repository as an HTTP server to realize the global convenient access to this UI application.

**Keyword:** Web technology; git tools; HTML modeling; ADIT development process;

# 第 1 章 引言

## 1.1 学习计划

通过系统性学习计算机科学技术，尤其是程序设计和软件工程领域的学习方法，训练自己的代码能力，架构自己感兴趣的技术路线。设计论文时插入图片配合文字，演示软件效果展示真实性，提供网址和二维码让设计的软件变得透明。同时查阅大量的相关技术文献，探索 HTML 内容建模、CSS 样式设计和 JavaScript 功能编程的基本技术和技巧。

## 1.2 项目技术路线

设计一个能够适应 PC 端和移动端的展示界面，包括对于书籍的翻页和视频的播放，设计有三个按钮，分别是开始、上下翻页。采用纯粹的 HTML 语言和 CSS 样式表和 JavaScript。页面分为上中下结构，上为导航栏，中为功能实现，下为底部信息栏。使用 HTML 标签将项目结构搭建好，利用 CSS 样式合理分配页面结构以及背景、字体颜色、整体布局等。然后使用 JavaScript 实现主区域的翻页功能。经过不断的测试和代码优化，最终完成整个项目的构建。

## 1.3 参考资料

- (1) W3C 的官方 HTML 文档。
- (2) Mozilla 组织的 MDN 社区的技术实践文章。
- (3) 《FOUNDATIONS OF COMPUTER SCIENCE》教材

## 1.4 研究方法

- (1) 理论学习，除了 1.3 参考资料中提到的参考资料，在中国知网和 csdn 社区中查阅相关技术文章。
- (2) 实践操作，使用 VS Code 编写代码，通过浏览器的开发者面板寻找错误和实时调试代码。
- (3) 参与社区讨论，将代码编写中遇到的各类情况上传至社区，提出或找寻解决

方法。

## 第 2 章 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[0]。学习 Web 标准和 Web 技术,学习编写 Web 程序和应用有关工具,最终架构一套高质量代码的跨平台运行的应用,这也是我的毕设项目应用的技术路线。

让我们从对 Web 的简要描述开始,Web 是 World Wide Web 的缩写。大多数人使用“Web”而不是“World Wide Web”,我们将遵循这个惯例。Web 是由世界各地的计算机用户共享的一系列文档,称为网页。不同类型的网页有不同的功能,但最基本的是它们都在计算机屏幕上显示内容。通过“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。

Web 编程是一个庞大的领域,不同类型的 Web 编程使用不同的工具实现。所有这些工具都与核心语言 HTML 一起工作,所以几乎所有的 Web 编程书籍都会在一定程度上描述 HTML。本教材详细介绍了 HTML5、CSS 和 JavaScript。这三种技术被认为是客户端 Web 编程的支柱。在客户端 Web 编程中,所有的网页计算都在最终用户的计算机上执行(即客户端计算机)。

Web 应用的程序设计体系由三大语言有机组成: HTML, CSS, JavaScript。这三大语言的组合也体现了人类社会化大生产分工的智慧,可以看作用三套相对独立体系实现了对一个信息系统的描述和控制,可以总结为: HTML 用来描述结构(Structure)、CSS 用来描述外表(presentation)、Javascript 用来描述行为(Behavior) [3];这也可以用经典的 MVC 设计模式来理解 Web 平台架构的三大基石,Model 可以理解为 HTML 标记语言建模,View 可以理解为用 CSS 语言来实现外观,Controller 则可理解为用 JavaScript 结合前面二个层次,实现了在微观和功能层面的代码控制。

### 第 3 章 软件开发的过​​程管理——增量式开发模式

软件生命周期中的开发过程包括四个阶段：分析、设计、实现和测试。有几种开发过程模型。我们在这里讨论最常见的两种：瀑布模型和增量模型。

(1) 瀑布模型 软件开发过程中非常流行的一种模型被称为瀑布模型（图 2-1）。

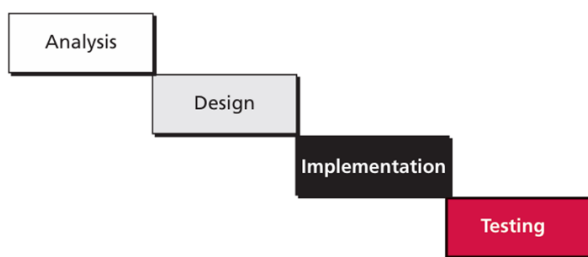


图 2-1 瀑布模型

在这个模型中，开发过程只向一个方向流动。这意味着在前一个阶段完成之前，不能开始下一个阶段。

例如，整个项目的分析阶段必须在设计阶段开始之前完成。整个设计阶段必须在实现阶段开始之前完成。

瀑布模型有优点和缺点。一个优点是每个阶段在下一个阶段开始之前都已完成。例如，设计阶段的团队知道自己要做什么，因为他们已经有了分析阶段的完整结果。测试阶段可以测试整个系统，因为整个正在开发的系统已经准备好。然而，瀑布模型的一个缺点是很难定位问题：如果在过程的某个部分出现问题，就必须检查整个过程。

(2) 增量模型 在增量模型中，软件按照一系列步骤进行开发。开发人员首先完成整个系统的简化版本。这个版本代表整个系统，但不包括细节。图 2-2 展示了增量模型的概念。

在第二个版本中，增加更多的细节，同时有些细节留下未完成的状态，并再次对系统进行测试。如果出现问题，开发人员知道问题是与新功能有关的。他们不会添加更多的功能，直到现有系统正常工作。这个过程一直持续到所有所需功能都被添加为止。

在软件工程的视角中的开发过程包括四个阶段：分析、设计、实施和测试，本文下面简称为 ADIT,此开发过程有两种经典模型：瀑布模型（The waterfall model）和增量模型(The incremental model)。

瀑布模型需要专业团队完美的配合，从分析、设计到实施，最后到测试，任何阶段

的开始必须基于上一阶段的完美结束。这对于我们大多数普通开发者是非常不方便的，比如在实施过程中，开发者发现设计段存在问题，开发者是不能纠正设计阶段的问题，只能按设计法案实施。我们大多数小微开发者，在软件的开发中总是在不断地优化设计，不断地重构代码，持续改进程序的功能和代码质量，

这种方式与增量开发模式非常匹配，因此在本项目的开发中我们采用了增量模型的开发模式。在本项目中我们一共做了六次项目的开发迭代，如下图 2-2 所示。

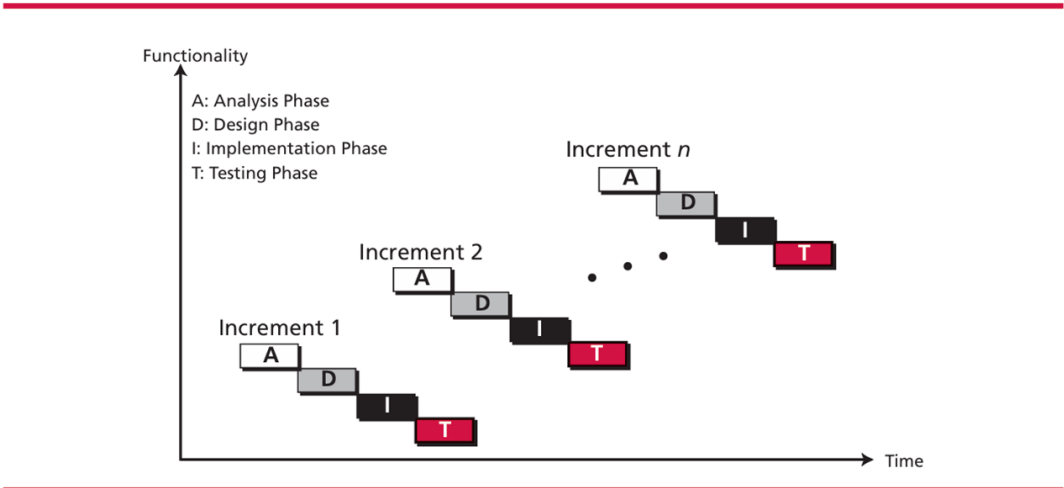


图 2-2 增量模型

## 第 4 章 移动互联时代的响应式设计和窄屏代码实现

### 4.1 项目分析和设计

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：

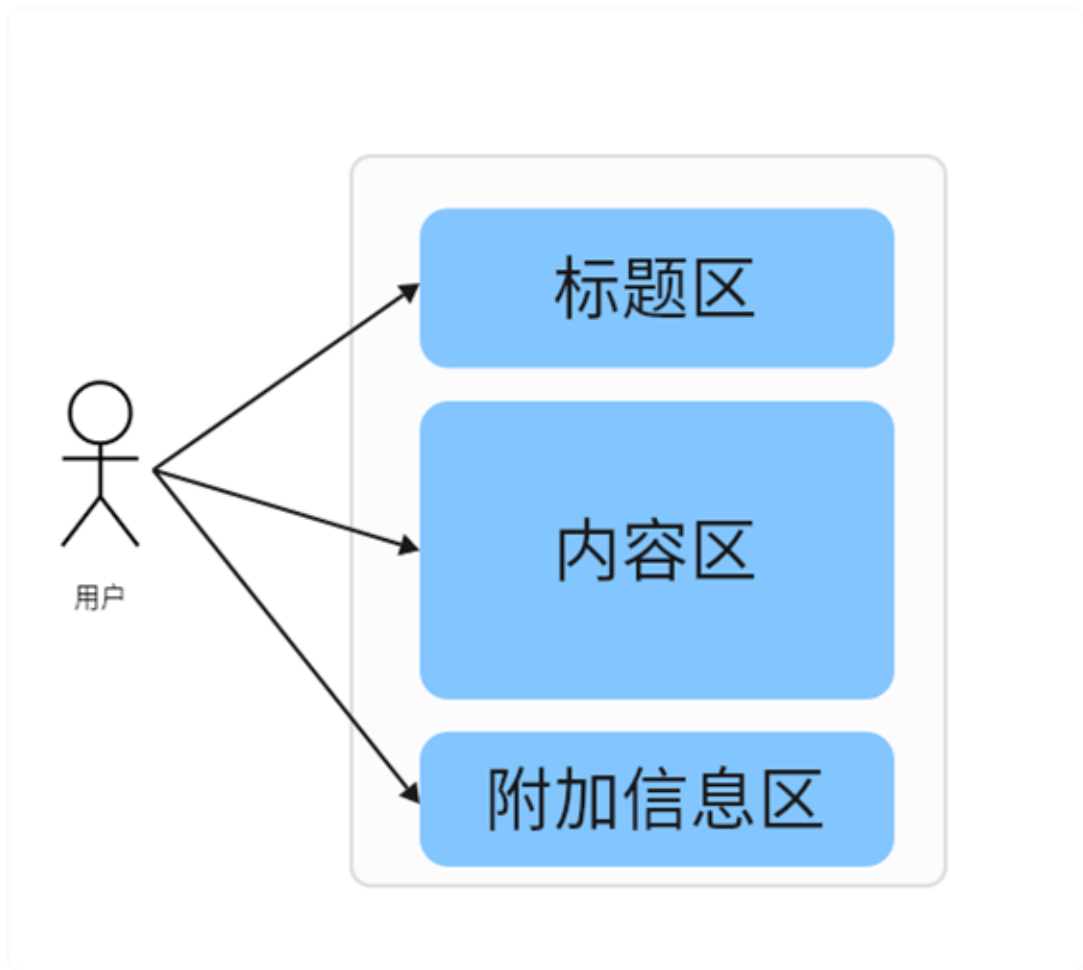


图 4-1 用例图

## 4.2 项目的实现和编程

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。

软件工程的增强式开发模式（A:分析 D:设计 I:实现 T:测试）阐述。

A: 首先需要分析的是如何实现窄屏响应式代码的原理，主要问题是需要克服长度和设备的兼容问题，那么我们就需要可以在页面加载代码时，就获取设备宽度，随后分配给对应的代码段进行等比例的放缩。

D: 首先需要在一个在 js 中设计一个类变量。该变量中包含至少两个功能，一个是获取显示设备的屏幕宽度的功能函数，二是将获取的宽度按照显示需要进行一个放缩的功能函数，根据放缩的函数得到的放缩大小，将得到的放缩大小数据可以响应式的设计到实际的显示代码中。



I: 代码实现如下:

```
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";  
T: 访问页面，在不同的设备终端上均能正常显示。
```



图 4-2 PC 端界面

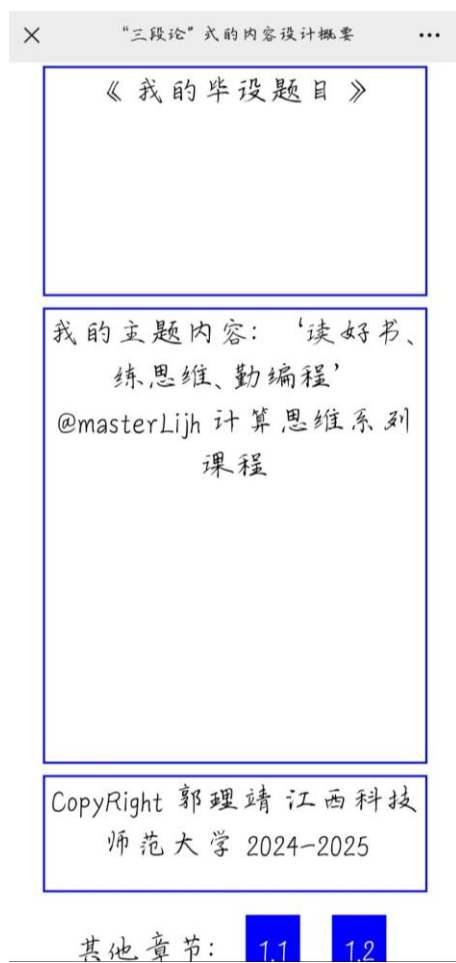


图 4-3 手机端界面

### 4.3 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 4-4 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 4-5 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 4-4 PC 端界面

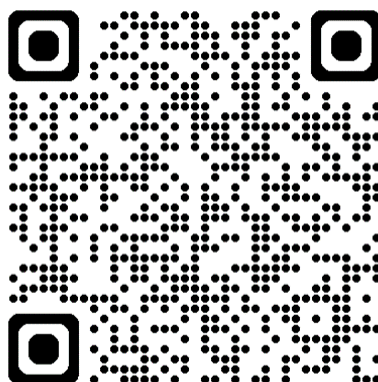


图 4-5 PC 端界面二维码

## 4.4 项目代码的提交和版本管理

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 江科师大李健宏  
$ git config user.email marsterlijh@jxstnu.edu.cn  
$ touch index.html myCss.css
```

编写好 index.html 和 myCss.css 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add index.html myCss.css  
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发  
成功提交代码后，gitbash 的反馈如下图 4-6 所示：
```

```
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发  
[master (root-commit) 32de024] 项目第一版：“三段论”式的内容设计概要开发  
2 files changed, 46 insertions(+)  
create mode 100644 index.html  
create mode 100644 myCss.css
```

图 4-6 git bash 反馈结果

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图 4-7 所示：

```
$ git log  
commit e7a2dd7859fd9601bd8634178ab640c1b011d22b (HEAD -> master)  
Author: Guo-lijing <1973810314@qq.com>  
Date: Sun Apr 28 10:54:53 2024 +0800  
  
本次代码提交完成了鼠标对UI的初步控制，提交了两个文件1-3.html和1-4.html。1-3h  
tml对屏幕超过1000px的用户屏幕增加了用于键盘反馈的UI界面；还对主区域的书的封面元  
素设定了鼠标响应事件，包括鼠标按下、移动、拖动和弹起。1-4在1-3的基础上增加了鼠标  
对UI界面主区域的拖动控制。  
  
commit 0331929af004b770fa7ff6b08966f4d4d2ef2e7c  
Author: Guo-lijing <1973810314@qq.com>  
Date: Thu Apr 25 09:36:07 2024 +0800  
  
本次代码提交初步完成响应式设计。代码实施完成了下面几步：1.为软件的四个区域分  
配了高度的比率，通过把设备的高度全部分配给body对象，结合前面的高度的分配，实现各  
区域响应式设计；2.为应用计算了基础字体的大小，并利用了body的遗传机制，结合CSS对  
字体的相对控制，实现了字体的响应式设计。  
  
commit 8405eb98a81424d4c3c2a421f1873b6ae299d5d3  
Author: Guo-lijing <1973810314@qq.com>  
Date: Mon Apr 22 09:42:28 2024 +0800  
  
本项目的第一次设计概要，完成了以下事情：1 用HTML语言实现了内容模型的建立，分  
...skipping...  
commit e7a2dd7859fd9601bd8634178ab640c1b011d22b (HEAD -> master)  
Author: Guo-lijing <1973810314@qq.com>  
Date: Sun Apr 28 10:54:53 2024 +0800  
  
本次代码提交完成了鼠标对UI的初步控制，提交了两个文件1-3.html和1-4.html。1-3html对屏幕超过1000px的用户屏幕增  
加了用于键盘反馈的UI界面；还对主区域的书的封面元素设定了鼠标响应事件，包括鼠标按下、移动、拖动和弹起。1-4在1-3  
的基础上增加了鼠标对UI界面主区域的拖动控制。  
  
commit 0331929af004b770fa7ff6b08966f4d4d2ef2e7c
```

图 4-7 仓库日志

# 第 5 章 宽屏和窄屏通用的响应式设计和代码实现

## 5.1 项目设计与分析

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。

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 所示。

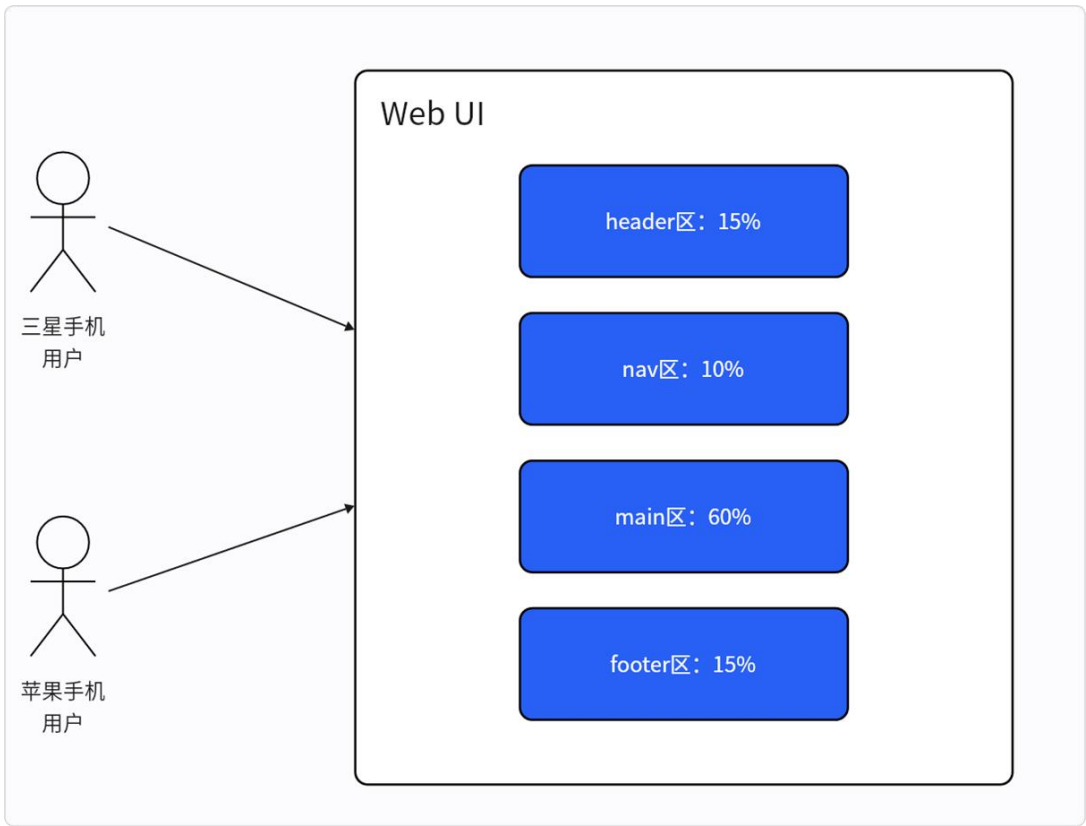


图 5-1 窄屏用户用例图

## 5.2 项目的实现和编程

项目代码如下：

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width , initial-scale=1">
    <style>
    </style>
    <title> "读好书、练思维、爱编程" </title>
  </head>
  <body>
    <header>
      <p id="book">
        现代电影赏析
      </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
      <div id="bookface">书的封面</div>
    </main>

    <footer>
      <p id="statusInfo">
        郭理靖 江西科技师范大学 2025
      </p>
    </footer>
    <div id="aid">
      <p>键盘响应区</p>
    </div>
    <script>
  </script>
  </body>
</html>
```

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 代码如下：

```
<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 对象的高度设置为设备/屏幕的高度，实现纵向全屏。
    //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.height = UI.appHeight - 70 + "px";
    document.body.style.width = UI.appWidth + "px";
    if(window.innerWidth < 1000 ){
        $("aid").style.display = "none";
    }
    $("aid").style.width = window.innerWidth - UI.appWidth - 30 + "px";
    $("aid").style.height = UI.appHeight - 62 + "px";
    //尝试对鼠标设计 UI 设计
    var mouse = {};
    mouse.isDown = false;
    mouse.x = 0;
    mouse.y = 0;
    mouse.deltaX = 0;
    $("bookface").addEventListener("mousedown",function(e){
        mouse.isDown = true;
        mouse.x = e.pageX;
        mouse.y = e.pageY;
        $("bookface").textContent = "鼠标按下，坐标为: (" + mouse.x + ","
+ mouse.y + ")"
    })
    $("bookface").addEventListener("mouseup",function(e){
        mouse.isDown = false;
        $("bookface").textContent = "鼠标弹起"
        if(mouse.deltaX > 50){
            $("bookface").textContent += ",拖动有效"
        }
    })

```

```

        $(".bookface").style.left = '7%' ;
        mouse.deltaX = 0;
        mouse.x = 0;
        mouse.y = 0;

    })

    $(".bookface").addEventListener("mousemove",function(e){
        e.preventDefault();
        if(mouse.isDown){
            let x = e.pageX;
            let y = e.pageY;
            mouse.deltaX = x - mouse.x;
            console.log("鼠标移动距离为: " + mouse.deltaX);
            $(".bookface").textContent = "鼠标拖动, 坐标为: (" + x + "," + y
+ ")";

            $(".bookface").style.left = mouse.deltaX + 'px' ;
        }
    })

    $(".bookface").addEventListener("mouseout",function(e){
        e.preventDefault();
        mouse.isDown=false;
        // mouse.x= 0 ;
        //mouse.y= 0 ;
        $(".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%' ;
        }
    })

})

```

```

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>

```

### 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：

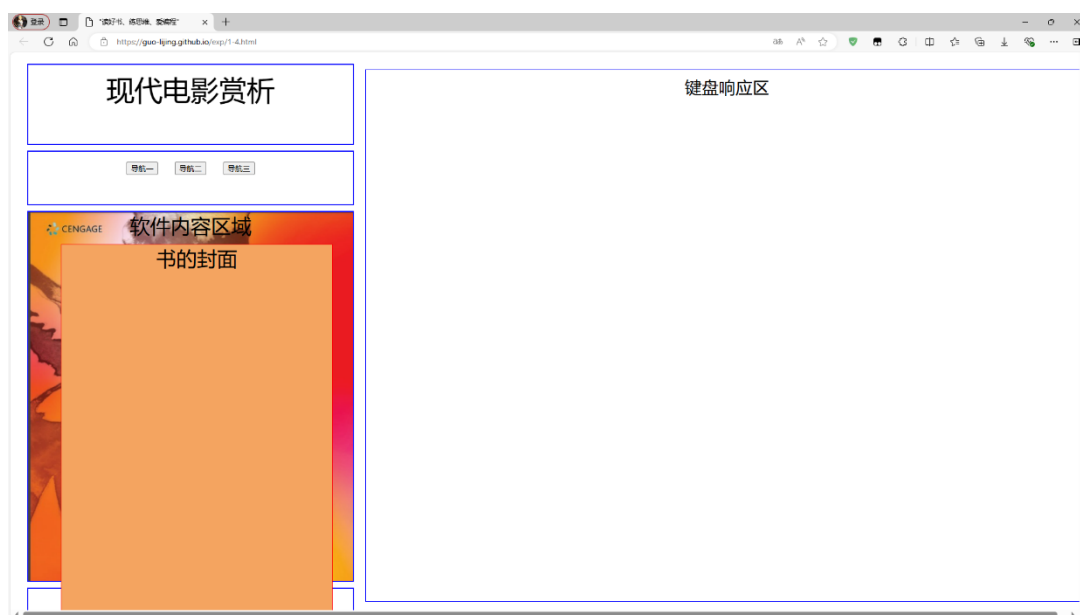


图 5-2 PC 端



图 5-3 手机端

## 第 6 章 个性化 UI 设计中鼠标模型

### 6.1 设计与分析

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 代码如下：

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width , initial-scale=1">
    <style>
    </style>
    <title> "读好书、练思维、爱编程" </title>
  </head>
  <body>
    <header>
      <p id="book">
        现代电影赏析
      </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
      <div id="bookface">书的封面</div>
    </main>
    <footer>
      <p id="statusInfo">
        郭理靖 江西科技师范大学 2025
      </p>
    </footer>
    <div id="aid">
      <p>键盘响应区</p>
    </div>
    <script>
    </script>
  </body>
</html>
```

JavaScript 代码如下：

```

<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 对象的高度设置为设备/屏幕的高度，实现纵向全屏。
    //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.height = UI.appHeight - 70 + "px";
    document.body.style.width = UI.appWidth + "px";
    if(window.innerWidth < 1000){
        $("aid").style.display = "none";
    }
    $("aid").style.width = window.innerWidth - UI.appWidth - 30 + "px";
    $("aid").style.height = UI.appHeight - 62 + "px";
    //尝试对鼠标设计 UI 设计
    var Pointer = {};
    Pointer.isDown= false;
    Pointer.x = 0;
    Pointer.deltaX = 0;
    { //Code Block begin
        let handleBegin = function(ev){
            Pointer.isDown=true;
            //console.log(ev.touches);
            if(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;
  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%';
  Pointer.deltaX = 0;
  Pointer.x = 0;
  }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%';
  Pointer.deltaX = 0;
  Pointer.x = 0;
  }
};

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);
}
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>

```

### 6.3 项目的运行和测试

本次测试主要测试 PC 端和移动端 main 结构中书的封面移动，如下图 6-1 和 6-2 所示

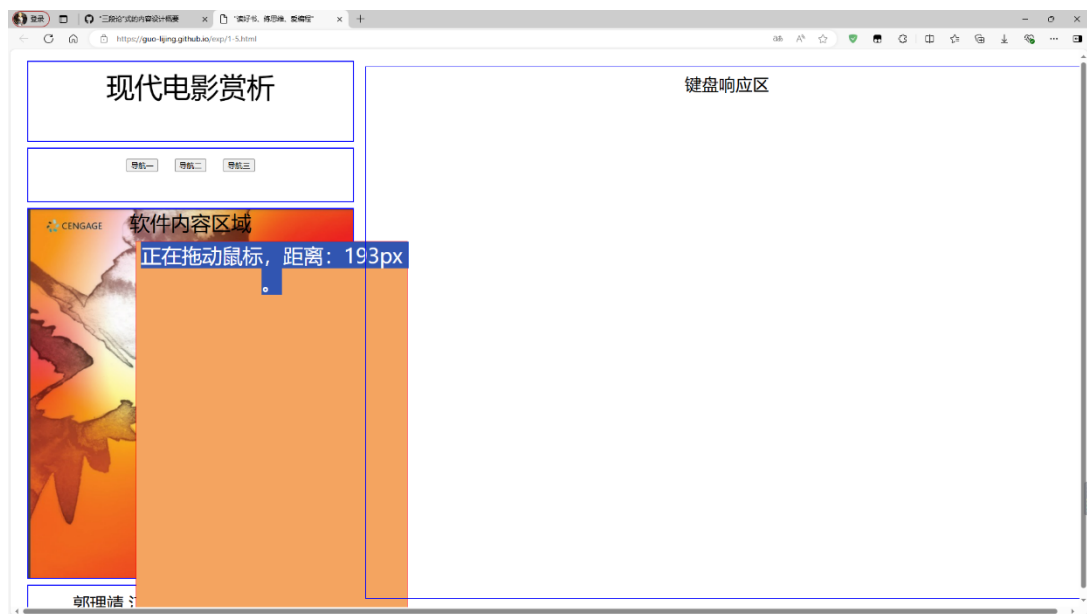


图 6-1 PC 端界面

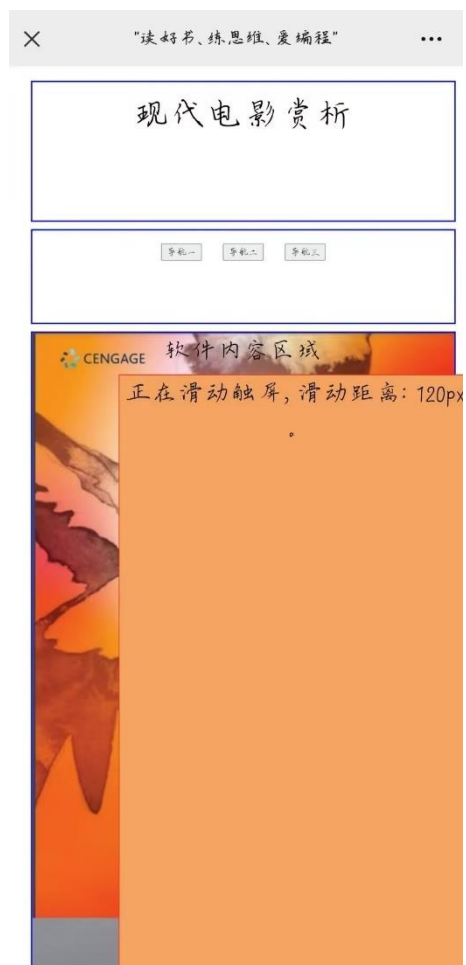


图 6-2 移动端界面

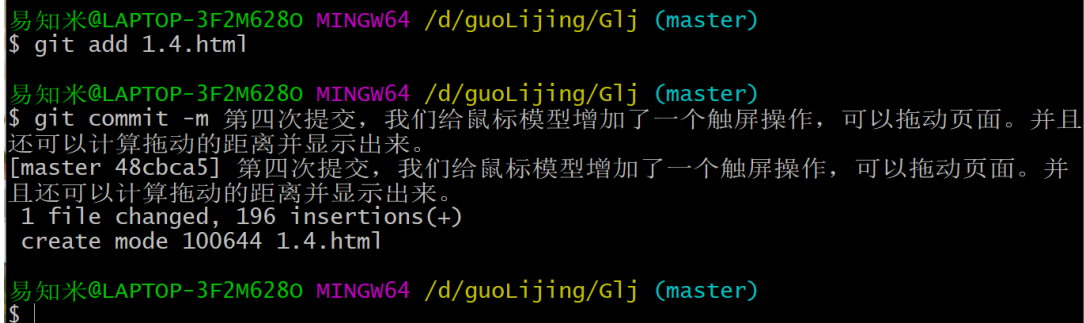
## 6.4 项目的代码提交和版本管理

编写好 1.4.html, myCss.css, 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 的反馈如下图 6-3 所示:



```
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$ git add 1.4.html

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$ 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显示出来。
[master 48cbca5] 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显示出来。
1 file changed, 196 insertions(+)
create mode 100644 1.4.html

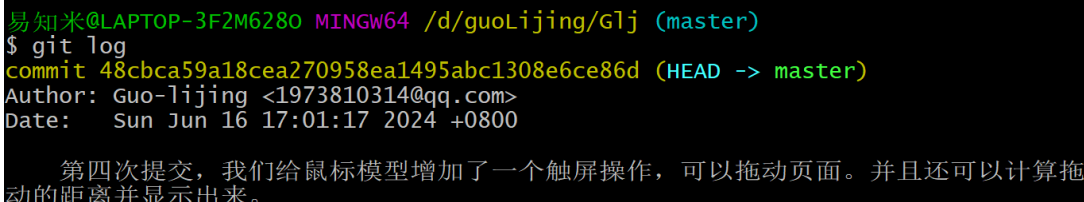
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$
```

图 6-3 git bash 反馈结果

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图 6-4 所示:



```
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$ git log
commit 48cbca59a18cea270958ea1495abc1308e6ce86d (HEAD -> master)
Author: Guo-lijing <1973810314@qq.com>
Date: Sun Jun 16 17:01:17 2024 +0800

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
动的距离并显示出来。
```

图 6-4 仓库日志

## 第 7 章 通用的 UI 设计, 用一套代码同时为触屏和鼠标建模

### 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 代码如下所示：

```
<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>

### 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 所示：



图 7-1 手机端界面

移动端用户可以通过扫描图 7-2 的二维码，运行测试本项目的第五次开发的阶段性效果。

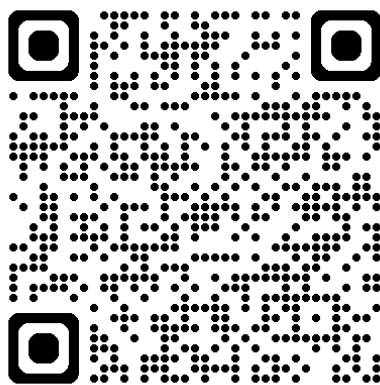


图 7-2 第五次项目二维码

## 7.4 项目的代码提交和版本管理

编写好 1.5.html, myCss.css, 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 的反馈如下图 7-3 所示:

```
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$ git add myJs.js

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (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 2ed4ec9] 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。
1 file changed, 351 insertions(+)
create mode 100644 myJs.js

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$
```

图 7-3 git bash 反馈结果

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图 7-4 所示:

```
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$ git log
commit 2ed4ec93e7f57daa39542564a681aa20728992d2 (HEAD -> master)
Author: Guo-Lijing <1973810314@qq.com>
Date: Sun Jun 16 17:25:56 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。

图 7-4 仓库日志

## 第 8 章 UI 的个性化键盘控制——应用 keydown 和 keyup 键盘底层事件

### 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 代码编写如下所示：

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width , initial-scale=1 ,user-scalable=no">
  <style>
    *{
      margin: 10px;
      text-align: center;
    }
    body{
```

```

position: relative;
}
header{
    border: 2px solid blue;
    height: 15%;
    font-size: 1.66em;

}
main{
    border: 2px solid blue;
    height: 70%;
    font-size: 1.2em;
    background-image: url(../lesson/CS.jpg);
    background-size: cover;
}
nav{
    border: 2px solid blue;
    height: 10%;
    font-size: 1.1em;
}
footer{
    border: 2px solid blue;
    height: 5%;
}
#aid{
    position: absolute;
    border: 1px solid blue;
    top: 0;
    left: 600px;
}
#bookface{
    position: absolute;
    width: 80%;
    height: 70%;
    border: 1px solid red;
    background-color: sandybrown;
    left: 10%;
}
#outputText{

    color: blue;
    border: 1px solid blue;
    height: 10%;
    width: 90%;

```

```

        padding: 0.2em;
    }
    #keyStatus{
        color: blue;
        border: 1px solid blue;
        height: 10%;
        width: 90%;
        padding: 0.5em;
    }
</style>
<title> "读好书、练思维、爱编程" </title>
</head>
<body>
    <header>
        <p id="book">
            现代电影赏析
        </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
        <div id="bookface">书的封面</div>
    </main>

    <footer>
        <p id="statusInfo">
            郭理靖 江西科技师范大学 2025
        </p>
    </footer>
    <div id="aid">
        <p>键盘响应区</p>
        <div id="outputText"></div>
        <div id="keyStatus"></div>
    </div>
    <script>
</script>
</body>
</html>

```

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 代码块如下所示：

```
<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 对象的高度设置为设备/屏幕的高度，实现纵向全屏。
  //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.height = UI.appHeight - 70 + "px";
  document.body.style.width = UI.appWidth + "px";
  if(window.innerWidth < 1000){
    $("aid").style.display = "none";
  }
  $("aid").style.width = window.innerWidth - UI.appWidth - 30 + "px";
  $("aid").style.height = UI.appHeight - 62 + "px";
  //尝试对鼠标设计 UI 设计
  var Pointer = {};
  Pointer.isDown = false;
  Pointer.x = 0;
  Pointer.deltaX = 0;
  { //Code Block begin
    let handleBegin = function(ev){
      Pointer.isDown = true;
      //console.log(ev.touches);
      if(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;
    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%';
        Pointer.deltaX = 0;
        Pointer.x = 0;
    }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%';
        Pointer.deltaX = 0;
        Pointer.x = 0;
    }
};
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 (e) {
//     let key = e.key;
//     $("outputText").textContent += key;
// });
$("body").addEventListener("keydown",function (e) {
    e.preventDefault();
    let key = e.key;
    $("outputText").textContent += key;
    $("keyStatus").textContent = "键" + key + "按下";
});
$("body").addEventListener("keyup",function (e) {
    e.preventDefault();
    let key = e.key;
    $("keyStatus").textContent = "键" + key + "弹起";
});
}
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>
```

### 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。

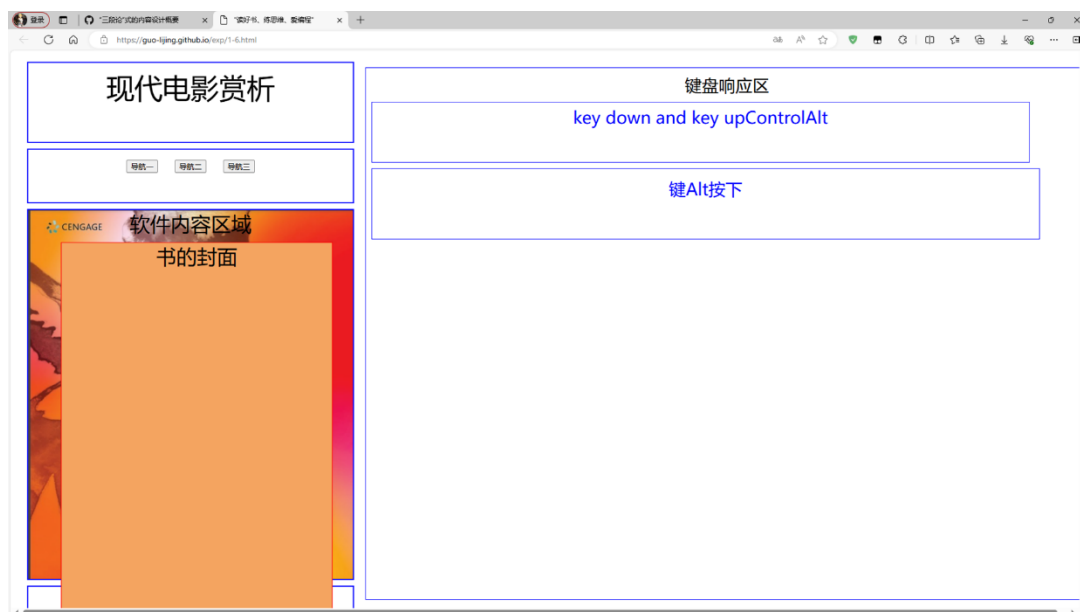


图 8-1 PC 界面展示

本代码 URL 地址

<https://guo-lijing.github.io/exp/1-6.html>

可扫描图 8-2 所示二维码查看

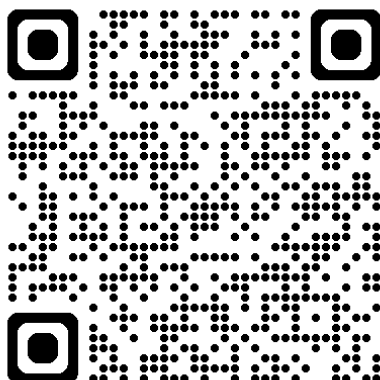


图 8-2 URL 地址二维码

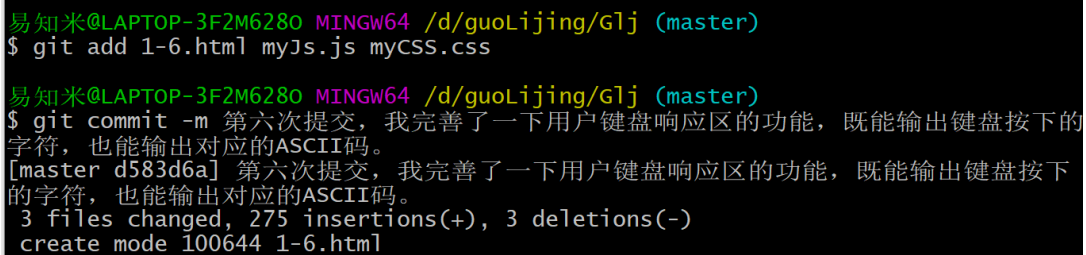
## 8.4 项目的代码提交和版本管理

编写好 1.6.html, myCss.css, 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 的反馈如下图 8-3 所示:



```
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$ git add 1-6.html myJs.js myCSS.css

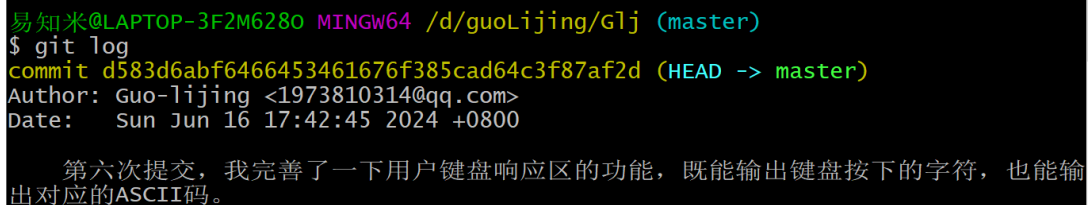
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$ git commit -m 第六次提交, 我完善了一下用户键盘响应区的功能, 既能输出键盘按下的字符, 也能输出对应的ASCII码。
[master d583d6a] 第六次提交, 我完善了一下用户键盘响应区的功能, 既能输出键盘按下的字符, 也能输出对应的ASCII码。
3 files changed, 275 insertions(+), 3 deletions(-)
create mode 100644 1-6.html
```

图 8-3 git bash 反馈结果

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图 8-4 所示:



```
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/Glj (master)
$ git log
commit d583d6abf6466453461676f385cad64c3f87af2d (HEAD -> master)
Author: Guo-lijing <1973810314@qq.com>
Date: Sun Jun 16 17:42:45 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。
```

图 8-4 仓库日志

## 第 9 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。

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;  
            //console.log(ev.touches);  
            if(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+")";  
            }  
        };  
    }
```

以上代码是用 Pointer 对象实现鼠标模型。

```
let handleEnd = function(ev){
  Pointer.isDown=false;
  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%';
    Pointer.deltaX = 0;
    Pointer.x = 0;
  }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%';
    Pointer.deltaX = 0;
    Pointer.x = 0;
  }
};
```

以上代码是响应鼠标行为或者触屏行为，当移动距离大于 100 像素时，规定为有效的行为。

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

```
    }  
  }  
};
```

以上代码是计算鼠标行为或者触屏行为的移动距离并且显示出计算的结果。

## 第 10 章 用 git 工具展开代码的版本管理和发布

跨世纪的经典 **Bash** 工具。

当我们谈论命令行时，实际上是指 **shell**。**Shell** 是一个接收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，称为 **bash**。这个名称是 **bourne-again shell** 的首字母缩写，它指的是 **bash** 是 **sh** 的增强替代品，而 **sh** 是由 **Steve Bourne** 编写的原始 **Unix shell** 程序[6]。

与 **Windows** 类似，类 **Unix** 操作系统如 **Linux** 将其文件组织在所谓的分层目录结构中。这意味着它们以树状模式的目录（在其他系统中有时称为文件夹）进行组织，这些目录可能包含文件和其他目录。文件系统中的第一个目录被称为根目录。根目录包含文件和子目录，子目录又包含更多的文件和子目录，以此类推[6]。

（围绕着 `git add html css javascript` 和 `git commit -m '过程标题'`，简述本项目在 **github** 上的 **http** 服务器的建立，提供 6 个开发阶段成果的访问的网址和二维码）

使用 **git** 连接远程 **GitHub**，并上传本地项目至 **GitHub** 以及在 **github** 上部署本地项目步骤如下：

上传：

（1）接入互联网，在 **github** 网页端创建 **github** 仓库（**moonandwindy.Github.io**）如下图所示 10-1 和图 10-2。



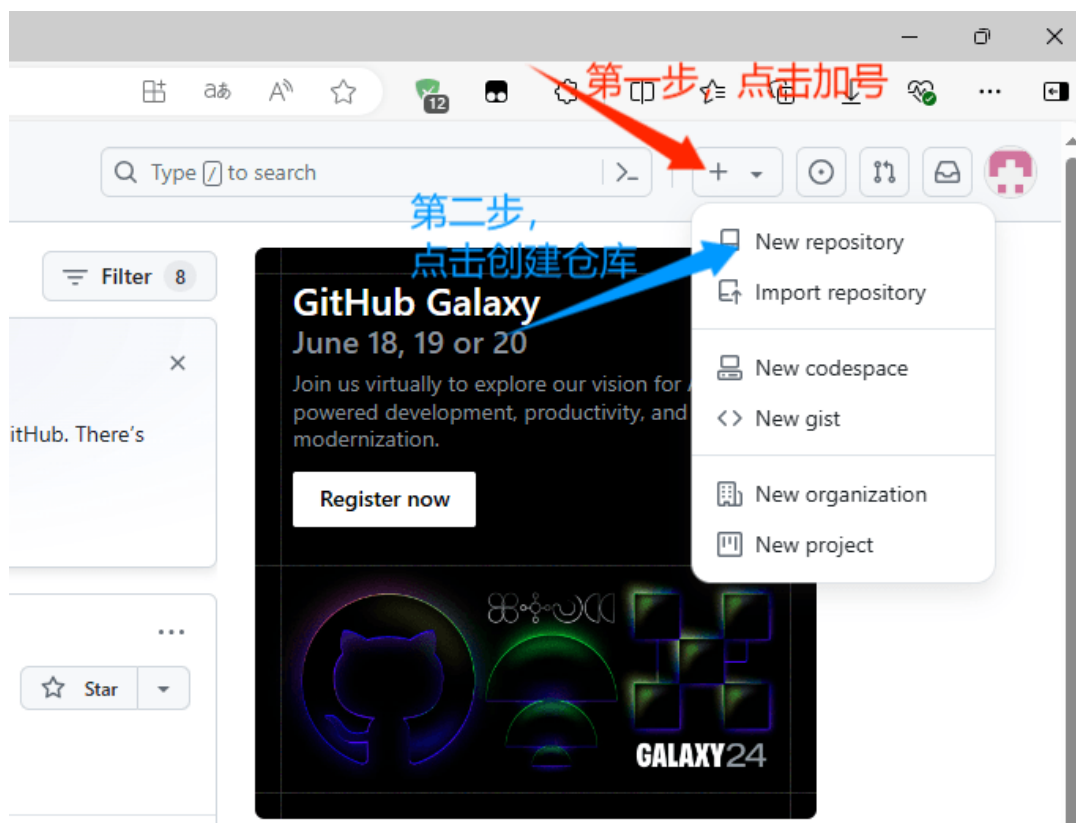


图 10-1 创建仓库

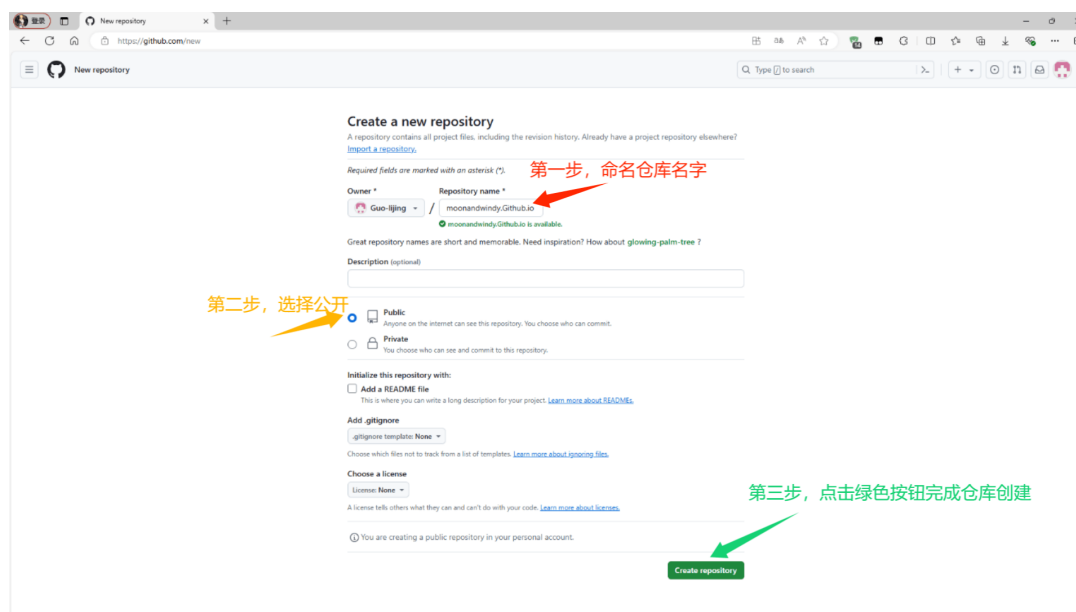


图 10-2 仓库命名

- (2) 运行 git 客户端, 使用 cd 命令进入本地项目所在文件夹, 运行 git init 指令。
- (3) 使用 git remote add origin

git@github.com:MoonAndWindy/moonandwindy.github.io.git

添加用于远程连接时用于指向对应仓库并含有密钥的的 ssl 至.git 文件夹中。

(4) 使用 `git add .` 指令将项目中的文件添加到准备上传的暂存区。

(5) 再使用 `git commit -m`”这是 1.1 文件提交”指令给文件添加信息。

(6) 最后使用 `git push origin main` 指令提交代码至远程仓库。如下图 10-3。

```
易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/moonandwindy/abc
$ echo "# moonandwindy.github.io" >> README.md

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/moonandwindy/abc
$ git init
Initialized empty Git repository in D:/guoLijing/moonandwindy/abc/.git/

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/moonandwindy/abc (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/moonandwindy/abc (master)
$ git commit -m "first commit"
[master (root-commit) 459d3c4] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/moonandwindy/abc (master)
$ git branch -M main

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/moonandwindy/abc (main)
$ git remote add origin https://github.com/Guo-lijing/moonandwindy.github.io.git

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/moonandwindy/abc (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 237 bytes | 237.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Guo-lijing/moonandwindy.github.io.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

易知米@LAPTOP-3F2M6280 MINGW64 /d/guoLijing/moonandwindy/abc (main)
$
```

图 10-3 git bash 结果

部署:

(1) 进入 github 网页端, 点击 setting

(2) 进入 setting 页面, 点击左侧 pages 功能选项

(3) 进入 pages 功能选项, 点击 build and deployment 下面的 save, 页面部署完毕。

如下图 10-4 和 10-5。

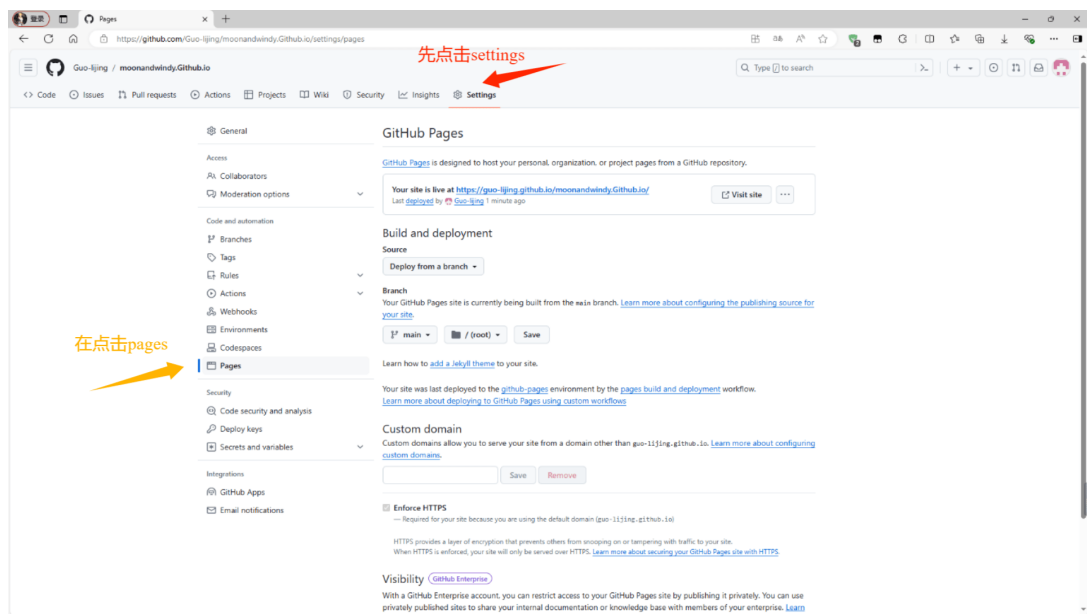


图 10-4 设置部署

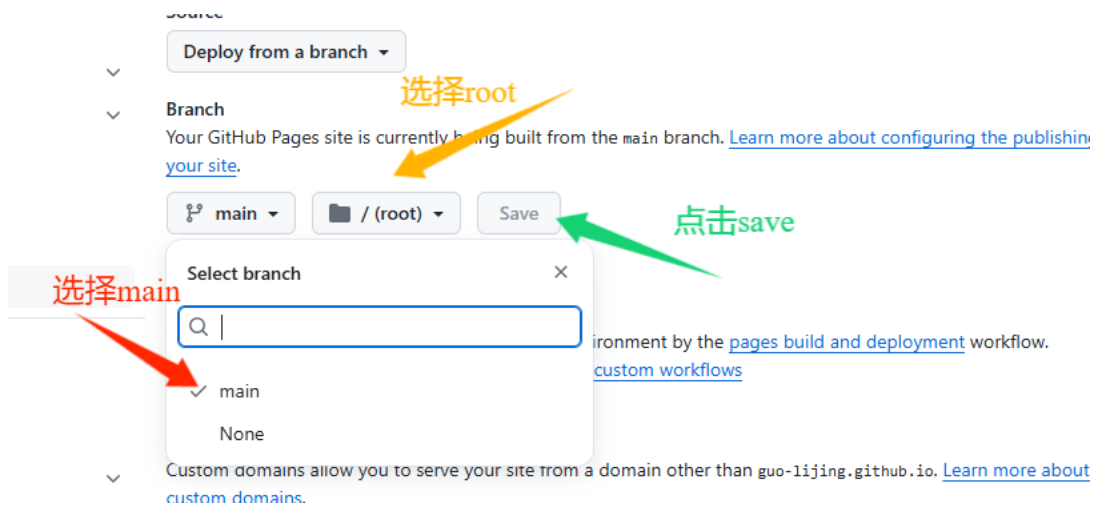


图 10-5 部署页面

页面二维码生成:

- (1) 使用 edge 浏览器，访问对应的域名。
- (2) 右键，选择生成二维码的功能选项。

六个网址以及对应的网页二维码如下:

图 10-1 至 10-6 分别是按顺序的展示页面



图 10-1 第一次项目二维码

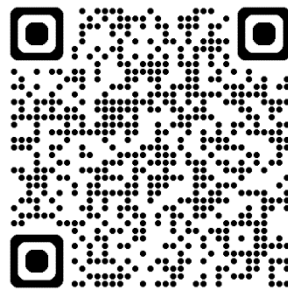


图 10-2 第二次项目二维码

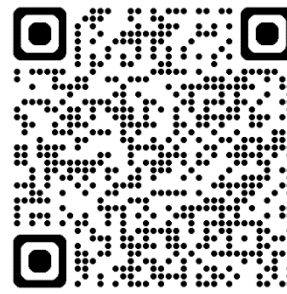


图 10-3 第三次项目二维码



图 10-4 第四次项目二维码

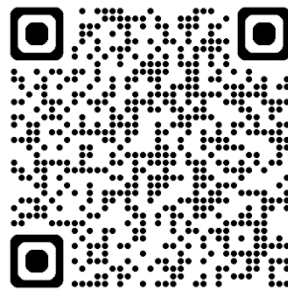


图 10-5 第五次项目二维码

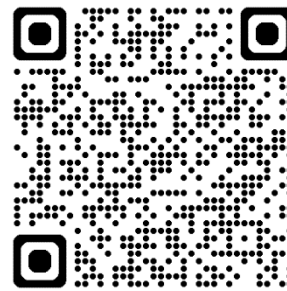


图 10-6 第六次项目二维码

## 参考文献

- [0] W3C. W3C's history. W3C Community. [EB/OL]. <https://www.w3.org/about/>. 2023.12.20
- [1] Douglas E. Comer. The Internet Book [M] (Fifth Edition). CRC Press Taylor & Francis Group, 2019: 217-218
- [2] John Dean, PhD. Web programming with HTML5, CSS, and JavaScript [M]. Jones & Bartlett Learning, LLC. 2019: 2
- [3] John Dean, PhD. Web programming with HTML5, CSS, and JavaScript [M]. Jones & Bartlett Learning, LLC. 2019: xi
- [4] Behrouz Forouzan. Foundations of Computer Science [M] (4th Edition). Cengage Learning EMEA, 2018: 274--275
- [5] Marijn Haverbeke. Eloquent JavaScript 3rd edition. No Starch Press, Inc, 2019.
- [6] William Shotts. The Linux Command Line, 2nd Edition [M]. No Starch Press, Inc, 245 8th Street, San Francisco, CA 94103, 2019: 3-7
- [7] Martina Seidl, Marion Scholz, et al. UML @ Classroom An Introduction to Object-Oriented Modeling [M]. Springer International Publishing Switzerland, 2015
- [8] Matti Tedre, Peter J. Denning. Computational Thinking, A Professional and Historical Perspective. Computational Thinking in Education A Pedagogical Perspective [C]. Routledge Taylor & Francis Group, 2022: 1-17