成 绩：

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 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：谢礼翔

学　　号：20203616

指导教师：李健宏

2024 年　6 月　19 日

目录

基于 Web 客户端技术的个性化 UI 的设计和编程

（Customized UI design and Programming based on Web client technology）

科师大元宇宙产业学院 2021 级 谢礼翔

摘要：在移动互联网时代，Web 技术因其跨平台的特性成为广泛采用的软件开发工具。本项目以 Web 客户端技术为研究重点，深入查阅了相关技术资料和文献，尤其参考了 mozilla 组织的 MDN 社区文章。主要探索了 HTML 内容建模、CSS 样式设计以及 JavaScript 功能编程的基础技术和技巧。结合本科课程学习成果，成功实现了一个个性化的用户界面（UI），该界面利用响应式设计技术，能够在不同设备上实现最佳的屏幕适配，包括 PC 端和移动端。在功能实现方面，项目采用 DOM 技术和事件驱动模式，实现了对鼠标、触屏和键盘事件的底层响应支持。特别地，针对鼠标和触屏设备，项目创新性地设计了对象模型，并通过代码模拟了这类指向性设备的行为，展示了项目的研究创新和亮点。为了有效管理设计和开发过程，项目采用了工程思想和软件工程的增量式开发模式。共进行了 6 次项目迭代开发，每次迭代包括经典的分析、设计、实现和测试阶段，以逐步完善应用程序的功能和性能。为了促进代码的分享和交流，项目积极与网络开发者合作，使用 git 工具记录了代码和开发过程的详细日志。总计进行了 12 次代码提交，展示了开发思路和代码优化的过程。最终，通过 gitbash 将项目上传至 GitHub，建立了自己的代码仓库，并将其设置为 HTTP 服务器，实现了全球范围内对本 UI 应用的便捷访问。

1. 前言

1.1. 学习计划

基于 web 客户端技术的个性化 UI 的设计与编程涵盖了前端开发的许多方面，包括 HTML、CSS 和 JavaScript 等技术。我将这个长时间的学习过程分为 3 个阶段：第一阶段，学习关于 HTML、CSS、JavaScript 的基础知识，理解 HTML 的

基本结构和标签，学习如何创建语义化的 HTML 文档；掌握 CSS 基本语法和选择器，学习 CSS 的布局技巧，了解响应式设计原理；学习 JavaScript 的基本语法和数据类型，理解 DOM 操作和事件处理。第二阶段，学习进阶知识，比如常见的前端框架，掌握组件化开发的概念和实践，掌握通过 Ajax 与后端 API 进行交互，学习处理异步请求和响应。第三阶段，实现个性化 UI 设计和实现，学习如何根据用户喜好和行为动态调整界面。第四阶段，项目实践，通过 GitHub 设计和完成个人的项目，与他人合作，学习并分享经验。

1.2. 项目的技术路线

首先要对项目进行需求分析。目标是确定项目的具体目标（例如，提高用户体验、增强用户参与度）。了解并定义项目的范围和限制条件。然后确定用户需求：收集用户偏好和行为数据，以了解用户的需求。确定需要个性化的 UI 元素（如推荐内容、动态布局）。了解用户对功能需求：列出所有功能，包括基础功能和个性化功能（如自定义主题、调整布局）。其次完成 UI 设计，遵守设计原则：以用户为中心设计，保证 UI 简单易用。完成响应式设计，确保在不同设备上表现一致。进行原型设计：使用工具（如 Figma、Sketch、Adobe XD）创建低保真到高保真的原型。设计交互原型，展示用户与系统的互动过程。最后需要进行用户测试：进行可用性测试，收集用户反馈并改进设计。

前端开发的技术选择：HTML/CSS/JavaScript 是前端开发的基础技术。根据项目需求选择 React、Vue 或 Angular 的前端框架。使用 Sass 或 Less 提高样式代码的可维护性。使用 Webpack、Babel 等工具优化代码和资源管理。

开发步骤：

(1) 项目初始化：使用脚手架工具（如 Create React App、Vue CLI）初始化项目。设置项目结构和目录。

(2) 组件开发：根据设计图开发 UI 组件，确保组件的可重用性。实现个性化功能，如主题切换、布局调整。

(3) 状态管理：使用 Redux、Vuex 或 Context API 等工具进行状态管理。管理用户偏好设置和个性化数据。

(4) API 集成：使用 AJAX 或 Fetch 与后端 API 通信。获取用户数据和个性

化内容。

(5) 响应式设计：使用媒体查询和弹性布局确保 UI 在不同设备上的良好体验。

1.3. 研究方法

在基于 Web 客户端技术的个性化 UI 设计和编程项目中，文献法可以帮助了解当前研究进展、技术趋势和最佳实践。首先，明确研究主题，即基于 Web 客户端技术的个性化 UI 设计和编程。然后，确定关键词，以便在文献搜索中使用。这些关键词可以包括：Personalized UI、Web client technology、User interface design、UX design、Frontend development、Personalization algorithms (如 collaborative filtering, content-based filtering)、Adaptive UI、Responsive design。

然后便是搜索文献，使用多个学术数据库和资源平台，进行系统的文献搜索。常用的数据库包括：Google Scholar：广泛覆盖各种学术资源。IEEE Xplore：主要针对工程和计算机科学领域。ACM Digital Library：计算机科学和信息技术方面的核心期刊和会议论文。SpringerLink：提供广泛的科学和技术书籍、期刊。PubMed：如果项目涉及用户心理学或行为研究，这里也有相关资源。根据标题和摘要，对检索到的文献进行初步筛选，选择与项目相关度较高的文献。然后，将这些文献分类整理，例如：理论基础、技术实现、应用案例、用户研究。深入阅读选定的文献，重点关注以下几个方面：

(1) 理论基础：了解个性化 UI 设计的基本概念、原理和发展历史。技术实现：研究现有的技术方案和框架，了解前端技术栈（如 React、Vue.js）以及个性化算法的实现方法。

(2) 应用案例：分析成功的案例，借鉴其设计思路和实现方法。

(3) 用户研究：了解用户需求分析的方法和工具，学习如何开展用户调研和测试。

在阅读过程中，做好详细的笔记，包括：重要观点和结论、实现方法和技术细节、关键数据和实验结果、文献的优点和不足。使用参考管理工具（如 EndNote、Zotero、Mendeley）来管理和组织文献资料，方便后续引用和查找。结合多篇文献，进行综合分析，提炼出对项目最有价值的信息，包括：当前研究的热点和趋势、常见的技术难点和解决方案、不同方法的优缺点和适用场景、基于文献分析

的结果，指导实际项目的设计和开发，包括：确定个性化 UI 的设计原则和策略、选择合适的前端技术和个性化算法、设计用户调研和测试方案。

最后，撰写一篇系统的文献综述，内容包括：研究背景和意义、文献的搜集和筛选方法、现有研究的主要内容和结论、研究空白和未来研究方向、对项目的启示和应用。文献法不仅能够为项目提供坚实的理论基础，还能帮助我们借鉴已有的研究成果，提高项目的科学性和可行性。

2. 技术总结和文献综述

2.1. Web 平台和客户端技术概述

Web 之父 Tim Berners-Lee 在发明 Web 的基本技术架构以后，就成立了 W3C 组织，该组织在 2010 年后推出的 HTML5 国际标准，结合欧洲 ECMA 组织维护的 ECMAScript 国际标准，几乎完美缔造了全球开发者实现开发平台统一的理想，直到今天，科学家与 Web 行业也还一直在致力于完善这个伟大而光荣的理想[1]。学习 Web 标准和 Web 技术，学习编写 Web 程序和应用有关工具，最终架构一套高质量代码的跨平台运行的应用，是我的毕设项目应用的技术路线。

2.1.1. 研究历史

1989 年，蒂姆·伯纳斯-李爵士发明了万维网（见最初的提案）。他在 1990 年 10 月创造了“万维网”一词，并写下了第一个万维网服务器“httpd”和第一个客户端程序（一个浏览器和编辑器）“世界万维网”。

他编写了“超文本标记语言”（HTML）的第一个版本，这是一种具有超文本链接功能的文档格式化语言，后来成为了 Web 的主要发布格式。他对 uri、HTTP 和 HTML 的最初规范随着网络技术的传播，在更大的圈子中得到了改进和讨论。

2.1.2. 研究背景

1994 年，在许多公司的敦促下，决定成立万维网联盟。蒂姆·伯纳斯-李爵士开始领导网络联盟团队的基本工作，以培养一个一致的架构，以适应网络标准的快速发展，以构建网站、浏览器和设备，以体验网络所提供的一切。

在创立万维网联盟的过程中，蒂姆·伯纳斯-李爵士创建了一个同行社区。Web 技术已经如此之快，以至于组装一个组织来协调 Web 标准至关重要。蒂姆接受了麻省理工学院举办 W3C 课程的邀请。他从一开始就要求 W3C 拥有全球的足迹。

2.1.3. Web 平台和 Web 编程

让我们从对网络的简要描述开始，它是万维网的缩写。大多数人说“是网络”而不是“万维网”，我们会遵循这个惯例。Web 是一个文档的集合，被称为网页，它们由世界各地的计算机用户（在大部分时间内）共享。不同类型的网页可以做不同的事情，但至少，它们都能在电脑屏幕上显示内容。我们所说的“内容”是指文本、图片和用户输入机制，如文本框和按钮。[2]

Web 编程是一个很大的领域，通过不同的工具实现不同类型的 Web 编程。所有的工具都与核心语言 HTML 一起工作，所以几乎所有的 web 编程书籍都在某种程度上描述了 HTML。这本教科书涵盖了 HTML5，CSS，和 JavaScript，所有的深入。这三种技术被认为是客户端网络编程的支柱。使用客户端 web 编程，所有网页计算都在终端用户的计算机（客户端计算机）上执行。[3] Web 应用的程序设计体系由三大语言有机组成：HTML, CSS, JavaScript。这三大语言的组合也体现了人类社会化大生产分工的智慧，可以看作用三套相对独立体系实现了对一个信息系统的描述和控制，可以总结为：HTML 用来描述结构（Structure）、CSS 用来描述外表（presentation）、Javascript 用来描述行为（Behavior）[3]；这也可以用经典的 MVC 设计模式来理解 Web 平台架构的三大基石，Model 可以理解为 HTML 标记语言建模，View 可以理解为用 CSS 语言来实现外观，Controller 则可理解为用 JavaScript 结合前面二个层次，实现了在微观和功能层面的代码控制。

2.2. 项目的增量式迭代开发模式

本项目作为一个本科专业学生毕业设计的软件作品，与单一用途的程序相比较为复杂，本项目所涉及的手写代码量远超过简单一二个数量级以上，从分析问题的到初步尝试写代码也不是能在几天内能落实的，可以说本项目是一个系统工

程，因此需要从软件工程的管理视角来看待和规范项目的编写过程。

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

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

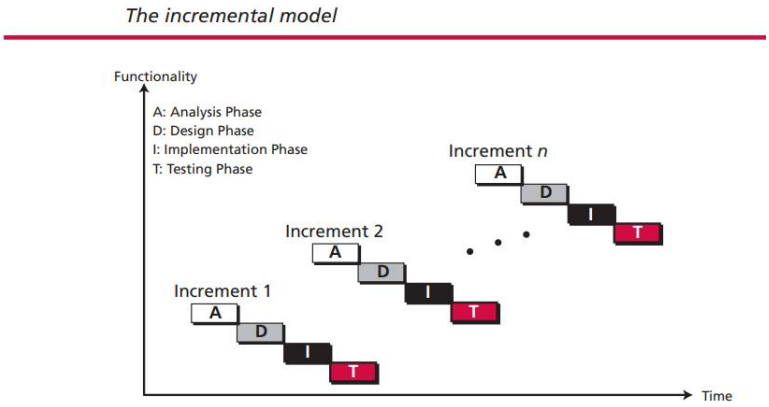


图 1. 增量开发模型

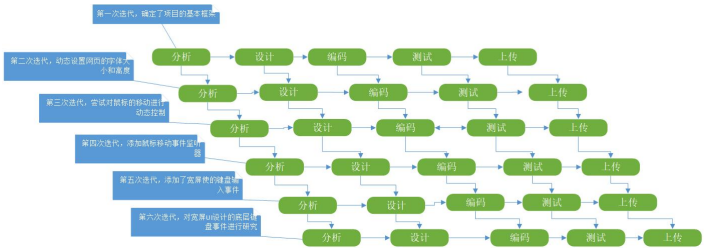


图 2. 本项目的增量开发模型

增量模型：在增量模型中，软件分一系列步骤进行开发。开发人员首先完成了整个系统的一个简化版本。这个版本表示整个系统，但不包括详细信息。图中显示了增量模型的概念。

在第二个版本中，添加了更多的细节，而一些没有完成，系统再次测试。如

果有问题，开发人员就知道问题在于新功能。在现有的系统正常工作之前，它们不会添加更多的功能。此过程，直到添加所有所需的功能[5]。

3. 内容设计概要

3.1. 分析和设计

这一步是项目的初次开发，本项目最初使用人们习惯的“三段论”式简洁方式开展内容设计，首先用一个标题性信息展示 logo 或文字标题，吸引用户的注意力，迅速表达主题；然后展现主要区域，也就是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的重点；最后则是足部的附加信息，用来显示一些用户可能关心的细节变化。用例图如图 3 所示，DOM 树如图 4 所示。

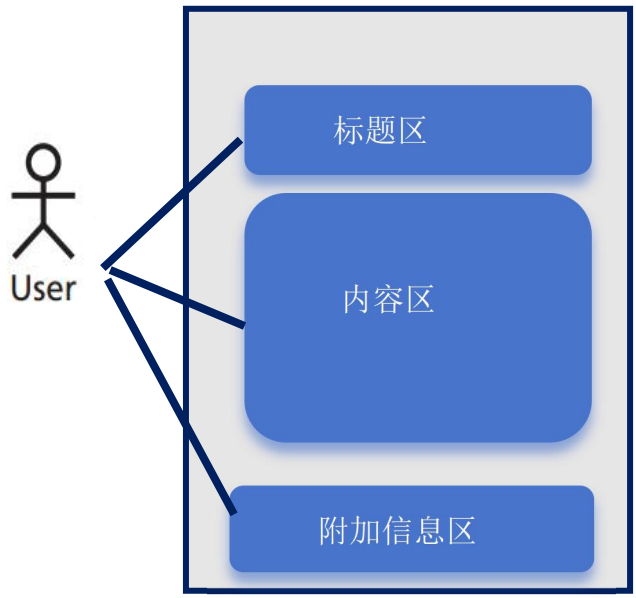


图 3 .用例图

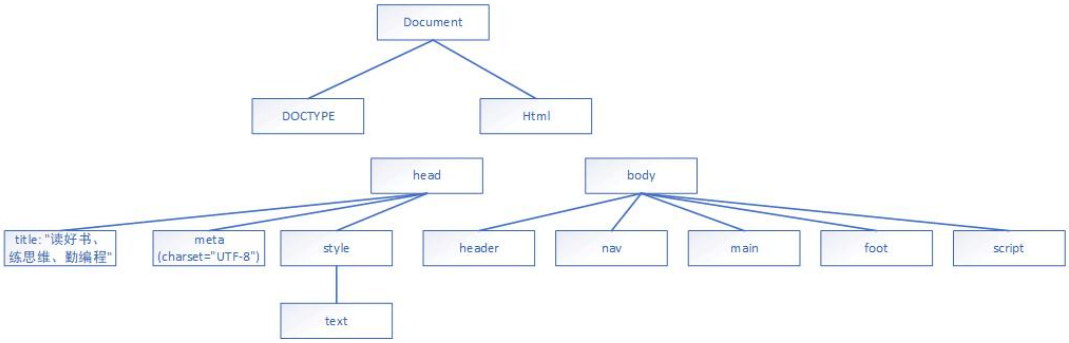


图 4. 初次设计的 DOM 树

3.2. 项目的实现和编程

(1) HTML 代码编写如下：

```
<!doctype html>
<html lang="en">
  <head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width , initial-scale=1">
  <!--
    增加这句后，可以看到浏览器的 DevTools 面板的模拟移动设备状态的变化
  -->
  <title>Welcome to myCTApp</title>
  <link rel="stylesheet" type="text/css" href="key.css">
</head>
<body >
  <header>
    <p id="book">
      现代开源软件开发环境搭建的研究
    </p>
  </header>
  <nav>
```

```
    <button>向前</button>
    <button>向后</button>
    <button>其他</button>
  </nav>
    <main id="main">
    <div id="bookface">
        <p>  这是书的封面图<br>
          在此对象范围拖动鼠标/滑动触屏<br>
                拖动/滑动超过 100 像素，视为有效 UI 互动！</p>
        <p>通过参考文献系统的阐述了现代开源软件的历史和现状，从软件开发的角度，总结了现代开源软件的开发环境，创造性的搭建了适合个人习惯的开发环境。<br>
          其中，主要用 Linux 命令行搭建软件操作系统环境，实现代码的版本控制，实现了代码编译器和系统操作命令行完美高效的运行，并通过修改操作系统的配置实现代码编译器，通过 Git 克隆获取 GitHub 的开源代码，实现在本地建立了个人的代码仓库以及对代码的快速修改和运行。<br></p>
    </div>
    </main>
  <footer>
    CopyRight xielx  江西科技师范大学  2024--2025
  </footer>
  <div id="aid">
    <p>用户键盘响应区</p>
    <div id="outputText"></div>
    <div id="keyStatus"></div>
  </div>
 <script    src="key.js"></script>
 </body>
</html>
```

<!doctype html>: 这是文档类型声明，指明文档类型为 HTML5。<html lang="en">: HTML 文档的根元素，指定了语言为英语。<head>: 包含了文档的元信息，比如字符编码、视口设置和引用的外部资源等。<meta charset="UTF-8">: 指定字符编码为 UTF-8，确保正确显示文本内容。

<meta name="viewport" content="width=device-width, initial-scale=1">: 设置视口的宽度等于设备的宽度，并且初始缩放为 1，适配移动设备的屏幕。<title>Welcome to myCTApp</title>: 设置网页标题为“Welcome to myCTApp”。<link rel="stylesheet" type="text/css" href="key.css">: 引入外部 CSS 文件，用于样式设置。<body>: 包含了网页的可见内容。<header>: 网页的页眉，包含一个段落元素 <p>，显示了书籍研究项目的标题。<nav>: 导航部分，包含了一组按钮，用于向前、向后或其他导航功能。<main id="main">: 主要内容区域，包含了一个 div 元素，用于展示书籍封面图和项目简介。 : 用于展示书籍封面图和项目简介的内容。<p>: 显示了书籍封面图的提示信息和项目简介的内容。<footer>: 网页的页脚，显示了版权信息和项目归属。 : 辅助区域，用于用户键盘响应。<p>: 显示了辅助区域的标题。 : 用于显示键盘响应的输出文本。 : 用于显示键盘状态信息。<script src="key.js"></script>: 引入外部 JavaScript 文件，用于处理键盘响应逻辑。

(2) CSS 代码编写如下：

```
*{margin:0 ;
   padding: 0;}
body{
        background-color: rgb(0,0,0) ;
        text-align: center ;
        /* max-width: 800px; */
          }
main>img#bookFace{
 max-width: 60%;
 position: absolute;
 left: 0 ;
```

```
  top: 0;
  transition: all 0.5s ease-out ;
  }

header{
  height: 15% ;
  background: rgb(250,100,0);
  color: rgb(250,250,0);
}
nav {
  height: 10% ;
}
footer{
  height: 10% ;
  background: rgb(30,20,10);
  color: white;
}
main{
  height: 45% ;
  position:relative;
  overflow: hidden ;
  background-image: url('lesson/back.jpg');
  /* lesson/back.jpg 是一张很小尺寸的图，用户选择书后会被 lesson 下的其他书封面遮住 */

  background-size:contain;
  background-position: center;
  background-repeat: no-repeat;
}
```

```
header>p#book{
    color: white;
    font-size: 1.2em;
    letter-spacing: 0.1em;
    text-shadow:    2px 2px black;
}
nav>p#chapter{
  color: white ;
  font-size: 0.9em;
  background-color: rgb(80,30,10) ;
}

footer>p#statusInfo{
    font-size: 0.75em;
    text-align:center;
  }
```

*{margin:0 ; padding: 0;}：将所有元素的外边距和内边距设置为 0，消除默认的间距。body：设置了 body 元素的样式属性：background-color：将背景颜色设置为黑色。text-align：设置文本内容居中对齐。main>img#bookFace：选择了 main 元素下 id 为 bookFace 的 img 元素，并设置了其样式属性：max-width：图片最大宽度为父元素的 60%。position：绝对定位，相对于最近的已定位父元素。left 和 top：图片左上角相对于父元素左上角的位置。transition：图片变化效果持续时间为 0.5 秒，采用 ease-out 缓动函数。header, nav, footer, main：分别设置了 header、nav、footer、main 元素的样式属性，包括高度、背景颜色、文字颜色等。header>p#book：选择了 header 元素下 id 为 book 的 p 元素，并设置了其样式属性：color：文字颜色为白色。font-size：字体大小为 1.2em。letter-spacing：字母间距为 0.1em。text-shadow：文字阴影效果，水平偏移 2px，垂直偏移 2px，颜色为黑色。nav>p#chapter：选择了 nav 元素下 id 为 chapter 的 p 元素，并设置了其样式属性：color：文字颜色为白色。font-size：字体大小为 0.9em。background-color：背景颜

色为深棕色。footer>p#statusInfo: 选择了 footer 元素下 id 为 statusInfo 的 p 元素，并设置了其样式属性：font-size: 字体大小为 0.75em。text-align: 文本内容居中对齐。这些样式设置将影响网页中各个元素的外观和布局，使得页面看起来更加美观和统一。

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 5 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 6 的二维码，运行测试本项目的第一次开发的阶段性效果。

图 5. PC 端运行页面

图 6. 移动端二维码

3.4. 项目的代码提交和版本管理

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

进入 gitBash 命令行后，按次序输入以下命令：

$ cd /

$ mkdir xielxText

$ cd xielxText/

$ git init

$ git config user.name ksd@谢礼翔

$ git config user.email 2803879657@qq.com

$ touch index.html myCss.css

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

$ git add index.html myCss.css

$ git commit -m 项目第一版：“三段论”式的内容设计概要开发

成功提交代码后，gitbash 的反馈如下所示：

```
28038@DESKTOP-ONU8144 MINGW64 /xielxText (master)
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
[master (root-commit) 91ed923] 项目第一版：“三段论”式的内容设计概要开发
 1 file changed, 38 insertions(+)
 create mode 100644 index.html
```

项目代码仓库自此也开启了严肃的历史记录，我们可以输入日志命令查看，

$ git log

gitbash 反馈代码的仓库日志如下所示：

```
28038@DESKTOP-ONU8144 MINGW64 /xielxText (master)
$ git log
commit 91ed9233b866d7f641210c6a960dd727d4a6c992 (HEAD -> master)
Author: ksd@谢礼翔 <2803879657@qq.com>
Date:   Thu Jun 13 20:56:59 2024 +0800
```

4. 移动互联时代的响应式设计和宽屏代码实现

响应性设计—适应显示硬件

在计算机上使用的显示器硬件差别很大，显示器的大小和分辨率取决于成本。设计师并没有选择每个网页的版本，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。因此，一个网页并没有提供很多细节。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。[1]

允许一个浏览器选择显示细节有一个有趣的结果：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或不同的浏览器上显示时，可能会出现略有不同[1]。

用 JavaScript 开动态读取显示设备的信息，然后按设计，使用 js+css 来部署适配当前设备的显示的代码。图 7 为响应式设计的 UML 图

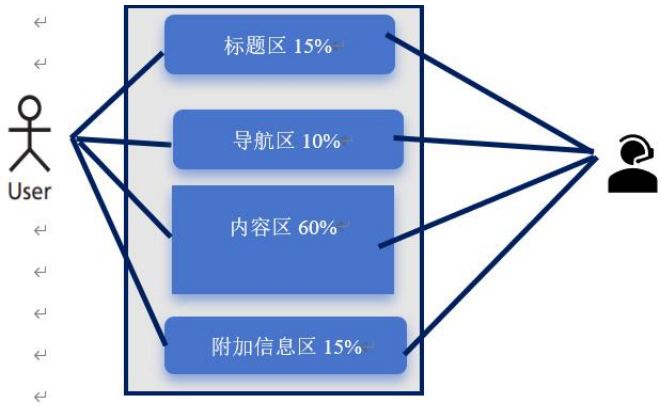


图 7. 响应式设计 UML

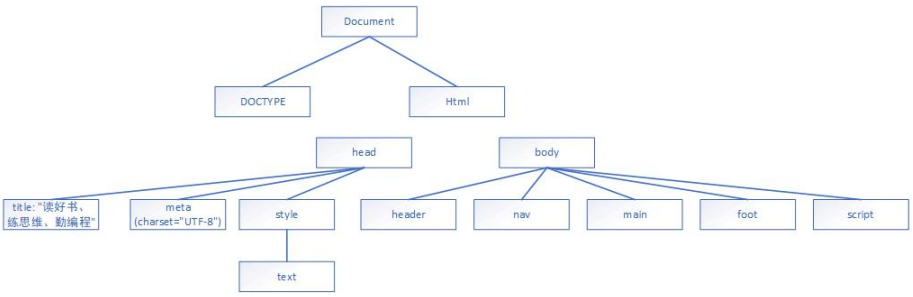


图 8. DOM 树

实现代码

用汉语言来描述我们是如何实现的，与上一阶段比较，本阶段初次引入了 em 和 % ，这是 CSS 语言中比较高阶的语法，可以有效地实现我们的响应式设计 。设计代码如下：

```
<style>
  *{
    margin: 10px;
    text-align: center;
  }
  header{
    border: 2px solid rgb(195, 27, 229);
    height: 15%;
    font-size: 1.0em;
  }
  main{
    border: 2px solid rgb(241, 92, 204);
    height: 70%;
    font-size: 0.6em;
    background-image: url(./CSS.jpg);
    background-size: cover;
  }
  nav{
    border: 2px solid rgb(246, 118, 203);
    height: 10%;
    font-size: 1.2em;
  }
  footer{
    border: 2px solid rgb(245, 131, 220);
    font-size:0.6em ;
    height: 5%;
```

```
}
```

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

```
<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>
```

4.1. 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，本文此处给出 PC 端用 Chrome 浏览器打开项目的结果以及移动端打开项目图的结果，如下图 9、图 10 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 11 的二维码，运行测试本项目的第二次开发的阶段性效果。

图 9. PC 端

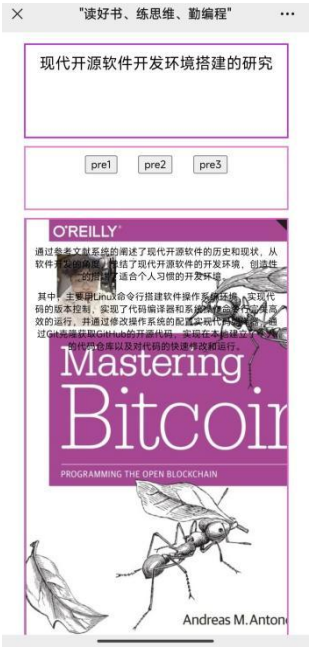


图 10. 手机端

图 11. 二维码

5. 适用移动互联时代的响应式设计

在移动互联时代，用户终端的多样性体现在设备类型、屏幕尺寸、分辨率和输入方式上。用户可能使用手机、平板电脑、笔记本电脑、台式电脑等各种设备

访问网站或应用程序。这意味着开发人员需要确保他们的网站或应用程序能够在不同的设备上提供一致的用户体验。

响应式设计是一种用于创建能够在不同设备上自适应布局和样式的设计方法。通过使用 CSS 和 JavaScript，可以实现响应式设计以应对移动互联时代用户终端的多样性。

使用 CSS 实现响应式设计媒体查询（Media Queries）：使用 CSS 的媒体查询可以根据设备的特性（如屏幕宽度、高度、方向等）来应用不同的样式。

```
css
@media screen and (max-width: 600px) {
  /* 在小屏幕上应用的样式 */
}
```

流式布局（Fluid Layout）：通过使用百分比或者其他相对单位而非固定单位（像素），可以创建能够根据屏幕大小自动调整的布局。

```
css
.container {
  width: 100%;
  max-width: 1200px; /* 限制最大宽度 */
}
```

弹性图片和媒体：使用 max-width: 100%来确保图片和视频在不同屏幕尺寸下能够自适应大小。

```
css
img, video {
  max-width: 100%;
  height: auto;
}
```

使用 JavaScript 实现响应式设计

获取设备信息：使用 JavaScript 可以获取设备的宽度、高度以及其他信息，从而根据这些信息做出相应的调整。

javascript

var screenWidth = window.innerWidth;

动态样式变化：通过 JavaScript 可以实时监测设备尺寸的变化，并在设备尺寸变化时对页面样式进行调整。

javascript

window.onresize = function(event) {

// 在这里执行样式的动态调整

};

条件加载：使用 JavaScript 可以根据设备类型或特性加载不同的内容或样式。

javascript

if (screenWidth < 600) {

// 加载针对小屏幕的内容或样式

}

结合 CSS 和 JavaScript，可以创建能够在不同设备上自适应的网页和应用程序，提供更好的用户体验，适应移动互联时代用户终端的多样性。

CSS 代码实现如下;

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

为所有元素设置一个 10 像素的外边距（margin）。

将所有文本水平居中。

```
header{
    border: 2px solid rgb(195, 27, 229);
    height: 15%;
    font-size: 1.4em;
}
```

设置 2 像素的紫色边框。高度为 15%（相对于包含它的元素）。

字体大小为 1.4 倍默认字体大小。

```
main{
  border: 2px solid rgb(241, 92, 204);
  height: 70%;
  font-size: 0.6em;
  background-image: url(./CSS.jpg);
  background-size: cover;
}
```

设置 2 像素的粉红色边框。高度为 70%。字体大小为默认字体大小的 0.6 倍。背景图片为 CSS.jpg，且覆盖整个背景区域。

```
nav{
  border: 2px solid rgb(246, 118, 203);
  height: 10%;
  font-size: 1.2em;
}
```

设置 2 像素的浅粉色边框。高度为 10%。字体大小为 1.2 倍默认字体大小。

```
footer{
  border: 2px solid rgb(245, 131, 220);
  height: 5%;
}
```

设置 2 像素的粉紫色边框。高度为 5%。

```
body{
    position: relative;
}
```

设置定位方式为相对定位。这通常用于使内部的绝对定位子元素相对于它进行定位。

```
#aid{
    position: absolute;
    border: 3px solid blue;
    top: 0px;
```

```
        left: 600px;
    }
```

绝对定位，边框为 3 像素的蓝色。定位在距离上方 0 像素，距离左边 600 像素的位置。

```
    #bookface{
        width: 80%;
        height: 80%;
        border: 2px solid yellowgreen;
        background-color: aquamarine;
        margin: auto;
    }
```

宽度和高度均为 80%。设置 2 像素的黄绿色边框。背景颜色为碧绿色。自动外边距（水平居中）。

CSS 代码展示了如何使用不同的选择器和属性来定义网页元素的样式。通过设置边框、尺寸、字体大小、背景图片、定位等属性，可以控制网页布局和视觉效果。此外，通过结合相对和绝对定位，可以实现更加复杂的布局需求。

用语言来描述我们是如何实现的：与上一阶段比较，本阶段首次使用了 JavaScript ，首先创建了一个 UI 对象，用于存储页面的一些属性。根据窗口的宽度设置 UI.appWidth 的值，如果窗口宽度大于 600 像素，则 UI.appWidth 为 600，否则为窗口的实际宽度。获取窗口的高度，并存储在 UI.appHeight 中。计算基准字体大小 baseFont，取页面宽度的 1/20。设置页面的基准字体大小为 baseFont 像素。设置页面的宽度和高度为窗口的宽度和高度。如果窗口宽度小于 1000 像素，隐藏 ID 为 "aid" 的元素。设置 ID 为 "aid" 的元素的宽度为窗口宽度减去 UI.appWidth 减 30 像素。设置 ID 为 "aid" 的元素的高度为窗口高度减去 62 像素。代码如下;

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

```
}else{
    UI.appWidth = window.innerWidth;
}
UI.appHeigth = window.innerHeight;
let baseFont = parseInt(UI.appWidth/20);
//字体大小、颜色可继承
document.body.style.fontSize = baseFont + "px";
//body 高度设置为屏幕高度
document.body.style.width = UI.appWidth + "px";
document.body.style.height = UI.appHeigth + "px";
if(window.innerWidth<1000){
    $("aid").style.display="none";
}
$("aid").style.width = window.innerWidth-UI.appWidth-30 + "px";
$("aid").style.height = UI.appHeigth-62+ "px";
```

5.1. 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，本文此处给出 PC 端用 Chrome 浏览器打开项目的结果以及移动端打开项目图的结果，如下图 9、图 10 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 11 的二维码，运行测试本项目的第三次开发的阶段性效果。

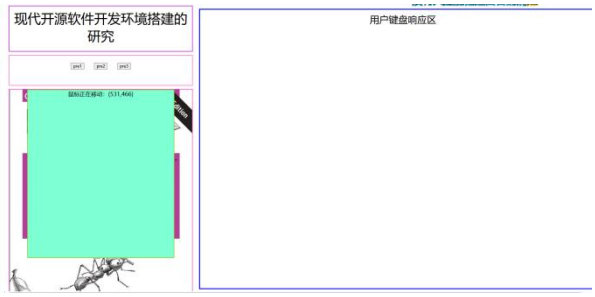


图 12. PC 端页面

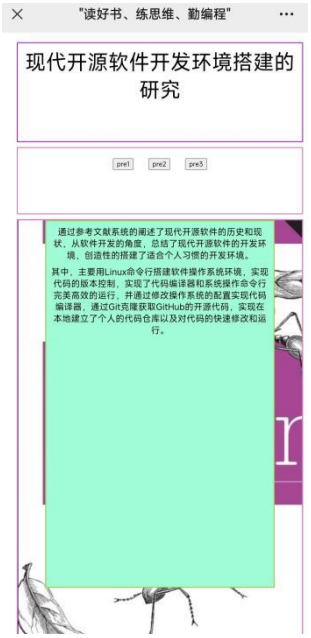


图 13. 移动端页面

图 14. 二维码

6. 个性化 UI 设计中鼠标模型

在网页开发中，可以使用一套代码逻辑同时为触屏和鼠标建立对象模型。这样可以实现响应触屏和鼠标事件的交互体验。以下是一种常见的方法：添加事件监听器：为触屏设备添加触屏事件监听器：touchstart、touchmove、touchend 等。为鼠标设备添加鼠标事件监听器：mousedown、mousemove、mouseup 等。编写事件处理函数：创建处理触屏事件的函数，例如 handleTouchStart、handleTouchMove、handleTouchEnd 等。创建处理鼠标事件的函数，例如 handleMouseDown、handleMouseMove、handleMouseUp 等。这些函数将根据具体的交互需求进行编写，处理相应的事件操作。根据设备类型动态绑定事件处理

函数：在页面加载时，使用 JavaScript 检测设备类型（触屏或鼠标）。根据设备类型，动态绑定相应的事件处理函数。如果是触屏设备，则将触屏事件监听器与对应的处理函数进行关联。如果是鼠标设备，则将鼠标事件监听器与对应的处理函数进行关联。Js 代码如下：

```
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("鼠标按下了："+"("+mouse.x+","+mouse.y+")");
    $("bookface").textContent= "鼠标按下了，坐标为："
"+"("+mouse.x+","+mouse.y+")";
});
$("bookface").addEventListener("mouseup",function(ev)
{
    mouse.isDown=false;
    mouse.x=0;
    mouse.y=0;
    $("bookface").textContent="鼠标松开！";
    if(Math.abs(mouse.deltaX)>50){
        $("bookface").textContent +=",这是有效拖动！";
    }
});
$("bookface").addEventListener("mousemove",function(ev)
```

```
{
    ev.preventDefault();
    if (mouse.isDown && Math.abs($('bookface').offsetLeft) < UI.appWidth /
5){
        console.log("认可的 mouse 事件： down and moving");
        mouse.deltaX = ev.pageX - mouse.x ;
        $('bookface').style.left = $('bookface').offsetLeft + mouse.deltaX + 'px' ;
        mouse.deltaX = 0 ;
    }
});
```

实现了一个简单的鼠标拖动功能。它跟踪鼠标按下、移动和松开的事件，并在特定条件下对元素进行拖动。当鼠标按下时触发。将 mouse.isDown 设为 true，表示鼠标已按下。记录按下时的坐标到 mouse.x 和 mouse.y。输出按下位置的坐标到控制台并更新页面内容。当鼠标松开时触发。将 mouse.isDown 设为 false，表示鼠标已松开。重置 mouse.x 和 mouse.y。更新页面内容，显示“鼠标松开！”。如果 mouse.deltaX 的绝对值大于 50，则认为进行了有效拖动，并在页面内容中显示。当鼠标移动时触发。调用 ev.preventDefault() 阻止默认行为（例如防止文本选择）。检查 mouse.isDown 是否为 true，即鼠标是否按下，同时确保元素的 offsetLeft 小于 UI.appWidth 的五分之一。如果条件满足，计算鼠标移动的水平距离 mouse.deltaX。更新元素的 left 样式属性，使其随鼠标移动。重置 mouse.deltaX 为 0。实现了一个基本的拖动功能。当用户在 bookface 元素上按下鼠标并移动时，元素会跟随鼠标移动。如果拖动距离超过一定阈值（50 像素），则会显示“这是有效拖动！”。展示了如何使用 JavaScript 来处理鼠标事件，并动态更新页面内容。

6.1. 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，本文此处给出 PC 端用 Chrome 浏览器打开项目的结果以及移动端打开项目图的结果，如下图 15、图 16 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 17 的

二维码，运行测试本项目的第四次开发的阶段性效果。

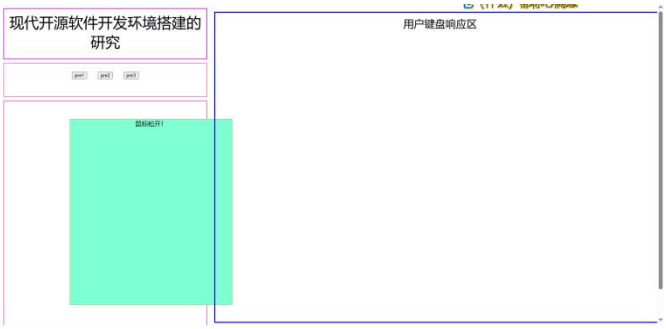


图 15. PC 端页面

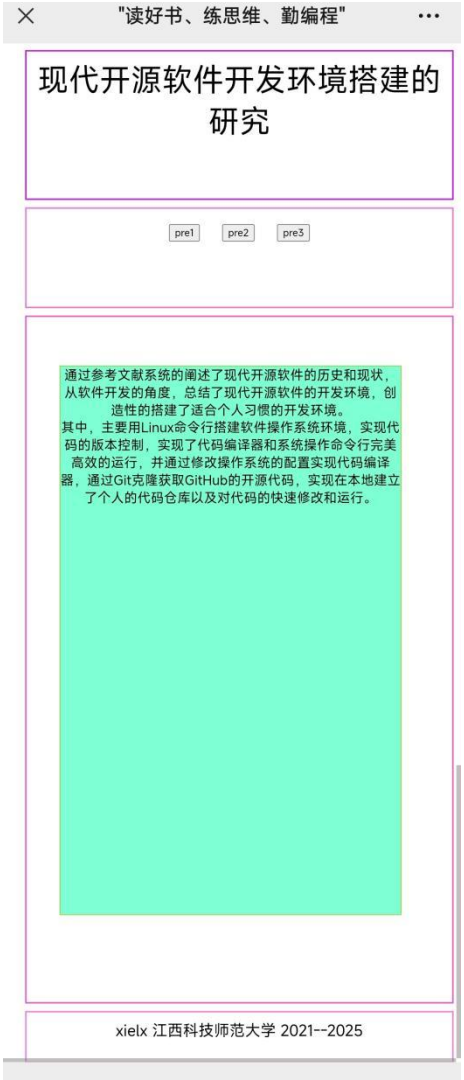


图 16. 移动端页面

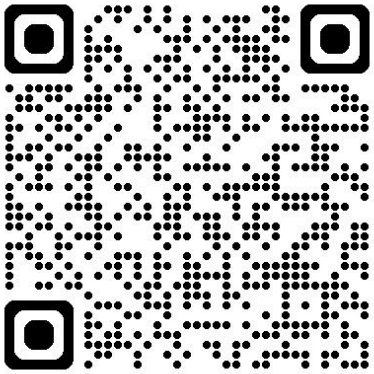


图 17. 二维码

7. 通用的 UI 设计

通过一套代码逻辑同时支持触屏和鼠标操作，关键在于建立一个统一的输入对象模型，并根据设备类型（触屏或鼠标）来处理不同的事件。这可以通过以下步骤实现：首先，我们需要检测设备是否支持触摸功能。可以通过检查 window 对象中是否存在 ontouchstart 属性来确定设备是否为触屏设备。创建一个 input 对象来存储输入设备的状态和位置信息。这个对象包含以下属性：isTouch：布尔值，表示设备是否支持触屏。isDown：布尔值，表示输入设备按钮（触摸或点击）是否被按下。x 和 y：数字，存储输入设备按下时的坐标。deltaX：数字，存储输入设备移动的水平距离。根据设备类型选择合适的事件类型进行监听。如果是触屏设备，使用 touchstart、touchmove 和 touchend 事件；如果是鼠标设备，使用 mousedown、mousemove 和 mouseup 事件。监听按下事件（touchstart 或 mousedown），并在事件处理函数中：将 isDown 设为 true，表示输入设备按钮被按下。获取按下时的坐标，并存储在 x 和 y 属性中。阻止默认行为（如文本选择或滚动）。监听移动事件（touchmove 或 mousemove），并在事件处理函数中：检查 isDown 是否为 true，即输入设备按钮是否被按下。获取当前移动的位置，并计算水平移动的距离 deltaX。更新目标元素的位置（如通过改变 style.left 属性）。重置 deltaX 为 0，以便下一次移动时重新计算。处理松开事件监听松开事件（touchend 或 mouseup），并在事件处理函数中：将 isDown 设为 false，表示输入设备按钮已松开。重置 x 和 y 属性为 0。根据移动距离 deltaX 判断是否是有效拖动（例如超过一定阈值），并执行相应的操作。通过

上述步骤，可以实现一套代码逻辑同时支持触屏和鼠标操作。具体的实现要点包括：使用统一的输入对象模型来存储输入设备的状态和位置信息。根据设备类型选择合适的事件类型进行监听。在事件处理函数中，根据当前设备类型获取相应的坐标信息，并更新目标元素的位置。这种方法不仅提高了代码的可读性和维护性，还增强了应用的用户体验，使其能够在不同类型的设备上都能正常工作。

Js 代码如下;

```
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;
        ev.preventDefault();
        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);
        $("bookface").addEventListener("click", function(ev) {
          console.log ("");
        });
        $("body").addEventListener("keypress",function(ev){
          let key = ev.key;
          $("outputText").textContent += key;
        });
        $("body").addEventListener("keydown",function(ev){
          ev.preventDefault();
          let key = ev.key;
```

```
    let c = ev.keyCode;
    $("keyStatus").textContent = '按下键：'+key + '的编码为' + c;
});
$("body").addEventListener("keyup",function(ev){
    ev.preventDefault();
    let key = ev.key;
    let c = ev.keyCode;
    $("keyStatus").textContent = '松开：'+key + '的编码为' + c;
    $("outputText").textContent += key;

});
```

它通过监听不同的事件来实现拖动和滑动的功能。handleBegin 在触屏或鼠标按下时触发。更新 Pointer 对象，并根据是触屏还是鼠标操作，记录初始位置并更新显示文本。handleEnd 在触屏或鼠标松开时触发。重置 Pointer.isDown 并判断滑动或拖动是否足够大（超过 100 px），以确定是否有效操作。handleMoving 在触屏或鼠标移动时触发。如果 Pointer.isDown 为 true，则计算并更新 Pointer.deltaX，同时更新 bookface 元素的位置和显示文本。分别为 mousedown、touchstart、mouseup、touchend、mouseout、mousemove 和 touchmove 添加事件监听器，绑定对应的处理函数。为 keypress、keydown 和 keyup 事件添加监听器，更新页面上显示的按键信息。通过监听各种触屏和鼠标事件，实现了对页面元素的拖动和滑动功能，同时还监听了键盘事件并更新页面显示内容。通过合理的事件处理函数，代码逻辑清晰且有效地处理了不同类型的用户交互。

7.1. 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，本文此处给出 PC 端用 Chrome 浏览器打开项目的结果以及移动端打开项目图的结果，如下图 18、图 19 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 20 的二维码，运行测试本项目的第五次开发的阶段性效果。

图 18. PC 端页面

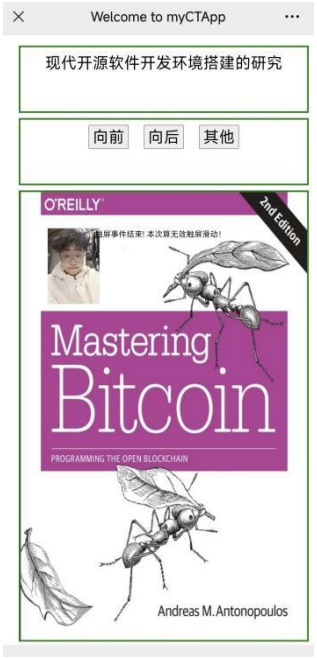


图 19. 移动端页面

图 20. 二维码

8. UI 的个性化键盘控制

8.1. 设计与分析

探索和利用 keydown 和 keyup 键盘底层事件，为未来 UI 的键盘功能提供底层强大的潜力。

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。代码如下所示：

```
$("body").addEventListener("keydown",function(ev){
ev.preventDefault() ; //增加“阻止事件对象的默认事件后”，不仅 keypress 事件将不再响应，而且系统的热键，如“F5 刷新页面/Ctrl+R ”、“F12 打开开发者面板”等也不再被响应
let k = ev.key;
let c = ev.keyCode;
$("keyStatus").textContent = "按下键 ： " + k + " ， "+ "编码 ： " + c;
});
$("body").addEventListener("keyup",function(ev){
ev.preventDefault() ;
let key = ev.key;
$("keyStatus").textContent =  key + " 键已弹起" ;
if (printLetter(key)){
$("typeText").textContent += key ;
}
function printLetter(k){
if (k.length > 1){
return false ;
}
let puncs =
['~','`','!','@','#','$','%','^','&','*','(',')','-','_','+','=',',',',','.',';',';',';','<','>','?','/',' ','\',''] ;
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 printLetter(k)
  });
```

通过一套代码逻辑同时支持触屏和鼠标操作，关键在于建立一个统一的输入对象模型，并根据设备类型（触屏或鼠标）来处理不同的事件。这可以通过以下步骤实现：首先，我们需要检测设备是否支持触摸功能。可以通过检查 window 对象中是否存在 ontouchstart 属性来确定设备是否为触屏设备。创建一个 input 对象来存储输入设备的状态和位置信息。这个对象包含以下属性：isTouch：布尔值，表示设备是否支持触屏。isDown：布尔值，表示输入设备按钮（触摸或点击）是否被按下。x 和 y：数字，存储输入设备按下时的坐标。deltaX：数字，存储输入设备移动的水平距离。根据设备类型选择合适的事件类型进行监听。如果是触屏设备，使用 touchstart、touchmove 和 touchend 事件；如果是鼠标设备，使用 mousedown、mousemove 和 mouseup 事件。监听按下事件（touchstart 或 mousedown），并在事件处理函数中：将 isDown 设为 true，表示输入设备按钮被按下。获取按下时的坐标，并存储在 x 和 y 属性中。阻止默认行为（如文本选择或滚动）。监听移动事件（touchmove 或 mousemove），并在事件处理函数中：检查 isDown 是否为 true，即输入设备按钮是否被按下。获取当前移动的位置，并计算水平移动的距离 deltaX。更新目标元素的位置（如通过改变 style.left 属性）。重置 deltaX 为 0，以便下一次移动时重新计算。处理松开事件监听松开事件（touchend 或 mouseup），并在事件处理函数中：将 isDown 设

为 false，表示输入设备按钮已松开。重置 x 和 y 属性为 0。根据移动距离 deltaX 判断是否是有效拖动（例如超过一定阈值），并执行相应的操作。通过上述步骤，可以实现一套代码逻辑同时支持触屏和鼠标操作。具体的实现要点包括：使用统一的输入对象模型来存储输入设备的状态和位置信息。根据设备类型选择合适的事件类型进行监听。在事件处理函数中，根据当前设备类型获取相应的坐标信息，并更新目标元素的位置。这种方法不仅提高了代码的可读性和维护性，还增强了应用的用户体验，使其能够在不同类型的设备上都能正常工作。系统用例图如下图 21。DOM 树如下图 22。

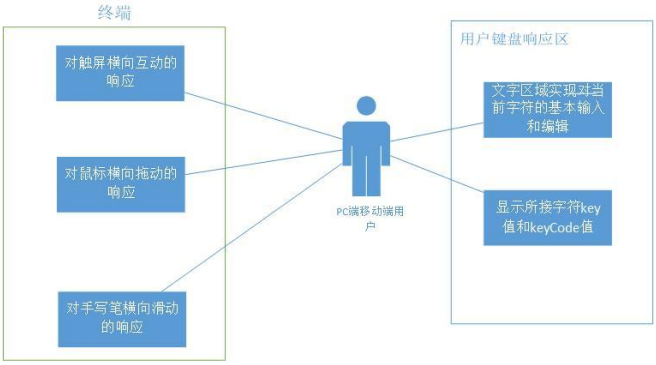


图 21. UI 的个性化键盘应用例图

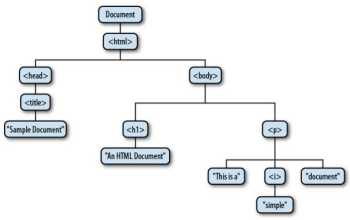


图 22. UI 的个性化键盘响应 DOM 树

8.2. 编程与实现

Js 代码如下;

```
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;
    ev.preventDefault();
    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);
$("bookface").addEventListener("click", function(ev) {
   console.log ("");
});
$("body").addEventListener("keypress",function(ev){
   let key = ev.key;
   $("outputText").textContent += key;
});
$("body").addEventListener("keydown",function(ev){
   ev.preventDefault();
   let key = ev.key;
   let c = ev.keyCode;
   $("keyStatus").textContent = '按下键：'+key + '的编码为' + c;
});
$("body").addEventListener("keyup",function(ev){
   ev.preventDefault();
   let key = ev.key;
   let c = ev.keyCode;
   $("keyStatus").textContent = '松开：'+key + '的编码为' + c;
   $("outputText").textContent += key;
});
```

它通过监听不同的事件来实现拖动和滑动的功能。handleBegin 在触屏或鼠标按下时触发。更新 Pointer 对象，并根据是触屏还是鼠标操作，记录初始位置并更新显示文本。handleEnd 在触屏或鼠标松开时触发。重置 Pointer.isDown 并判断滑动或拖动是否足够大（超过 100 px），以确定是否有效操作。handleMoving 在触屏或鼠标移动时触发。如果 Pointer.isDown 为 true，则计算并更新 Pointer.deltaX，同时更新 bookface 元素的位置和显示文本。分别为 mousedown、touchstart、mouseup、touchend、mouseout、mousemove 和 touchmove 添加事件监听器，绑定对应的处理函数。为 keypress、keydown 和 keyup 事件添加监听器，更新页面上显示的按键信息。通过监听各种触屏和鼠标事件，实现了对页面元素的拖动和滑动功能，同时还监听了键盘事件并更新页面显示内容。通过合理的事件处理函数，代码逻辑清晰且有效地处理了不同类型的用户交互。

8.3. 测试与运行

项目的运行和测试至少要通过二类终端，本文此处给出 PC 端用 Chrome 浏览器打开项目的结果以及移动端打开项目图的结果，如下图 23、图 24 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 27 的二维码，运行测试本项目的第六次开发的阶段性效果。

图 23. PC 端页面

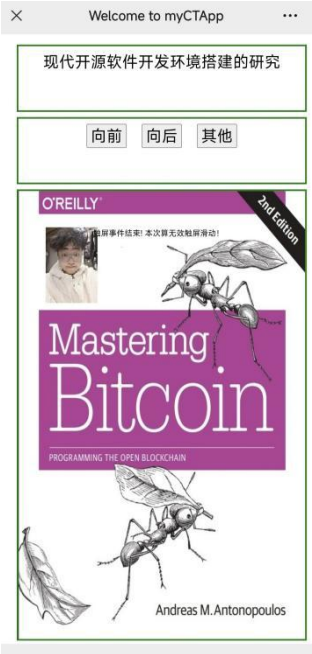


图 24. 移动端页面

8.4. 代码提交和版本管理

```
28038@DESKTOP-ONU8144 MINGW64 /xielxText (master)
$ git log
commit 96f0073495dc5216483d77ed0823882111d92b1e (HEAD -> master)
Author: ksd@谢礼翔 <2803879657@qq.com>
Date:   Wed Jun 19 13:41:49 2024 +0800

    xlx第六次提交，修改了键盘控制
```

图 25. 项目代码第六次提交

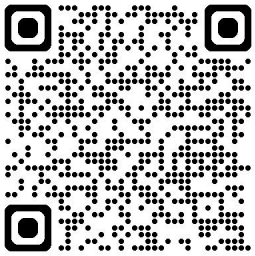


图 27. 二维码

9. 本项目中的高质量代码

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

如果你的计算机任务是一个常见的，很容易理解的任务，比如显示你的电子邮件或像一个计算器，你可以打开适当的应用程序，并开始工作。但是对于唯一的或开放式的任务，可能没有应用程序。

这就是编程可能会出现的地方。编程是构建一个程序的行为——一组精确的指令，告诉计算机要做什么。因为计算机是愚蠢的，迂腐的野兽，编程从根本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思考，那么编程可能是值得的。它允许你在几秒钟内完成一些永远需要手工完成的事情。这是一种让你的电脑工具做一些它以前不能做的事情的方法。它提供了一个很好的抽象思维的练习[6]。创建一个 Pointer 对象，践行 MVC 设计模式，设计一套代码同时对鼠标和触屏实现控制。

面向对象思想，封装，抽象，局部变量，函数式编程，逻辑。

（围绕着抽象定义函数、代码块、模型设计以及降低全局变量的使用来写）

10. 用 gitBash 工具管理本项目的 http 服务器

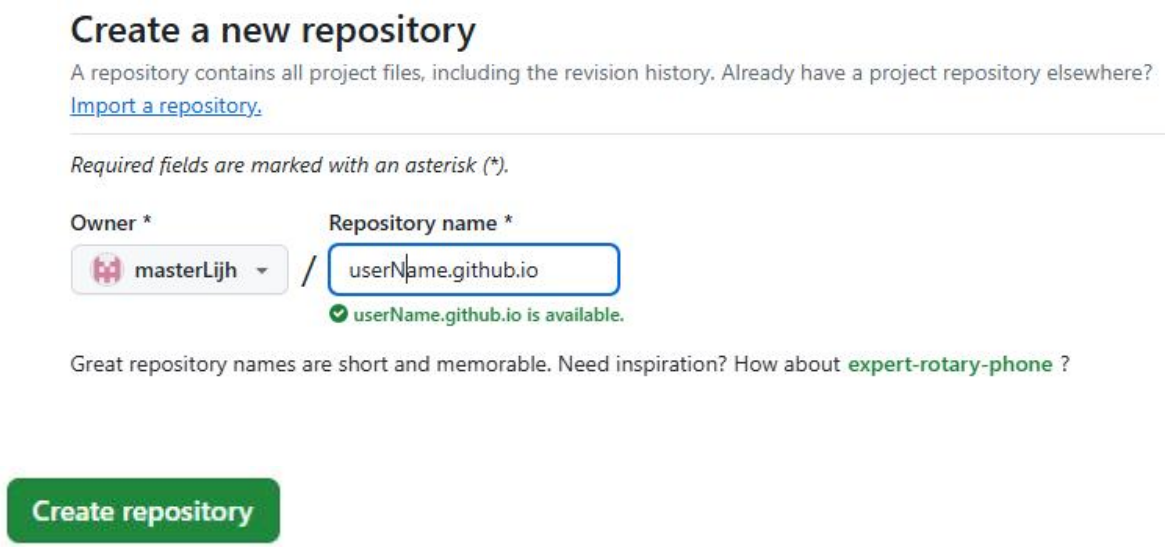
10.1. 经典 Bash 工具介绍

当我们谈到命令行时，我们实际上指的是 shell。shell 是一个程序，并将它们传递给操作系统执行。几乎所有的 Linux 发行版都提供了一个来自 GNU 项目的 shell 程序。这个名字是伯恩-再次外壳的首字母缩写，指的是 bash 是 sh 的增强替代品，最初的 Unix 外壳程序由史蒂夫·伯恩写。[7]

和 Windows 一样，类似 inix 的 Linux 操作系统以分层目录结构组织文件。这意味着它们被组织成一个树状的目录模式（在其他系统中有时称为文件夹），其中可能包含文件和其他目录。文件系统中的第一个目录称为根目录。根目录包含文件和子目录，其中包含更多的文件和子目录，等等。[7]

10.2. 通过 gitHub 平台实现本项目的全球域名

10.3. 创建一个空的远程代码仓库



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

10.4. 设置本地仓库和远程代码仓库的链接

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

$ git init

$ git add README.md

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

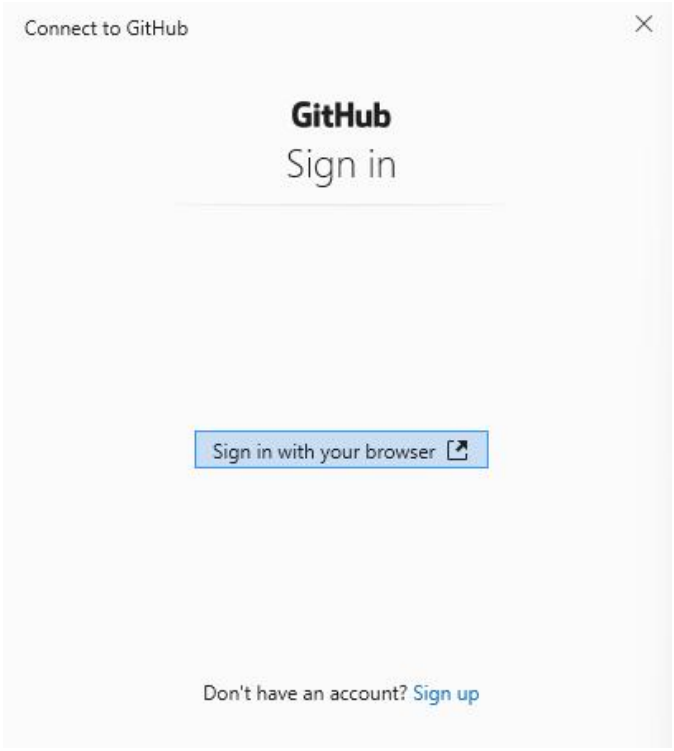
$ git branch -M main

$ git remote add origin

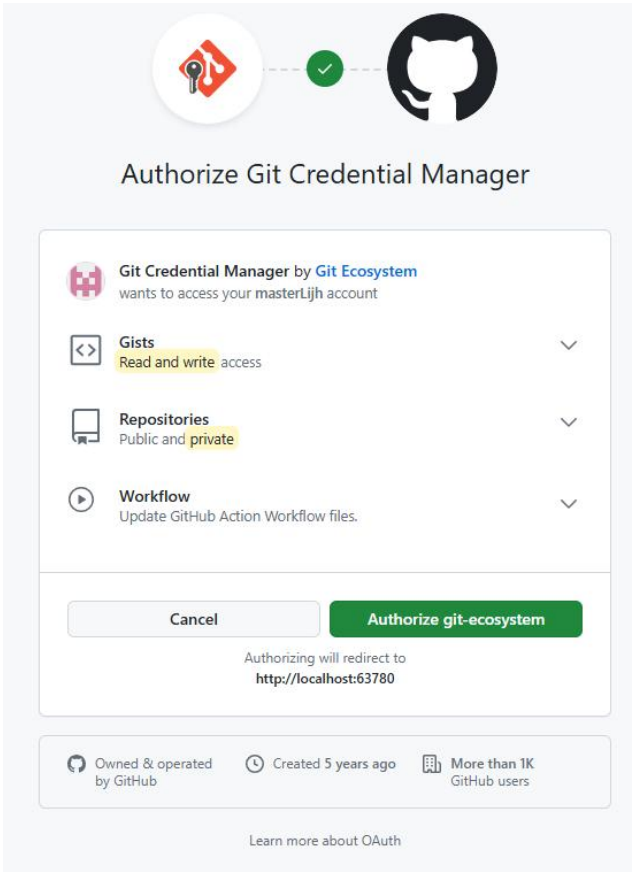
https://github.com/masterLijh/userName.github.io.git

$ git push -u origin main

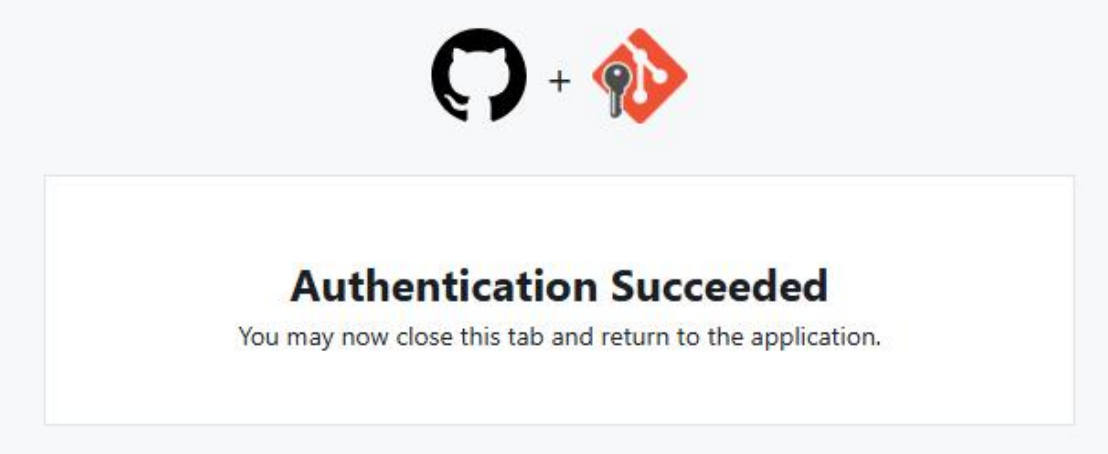
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



再次确认授权 gitBash 拥有访问改动远程代码的权限，如下图所示：

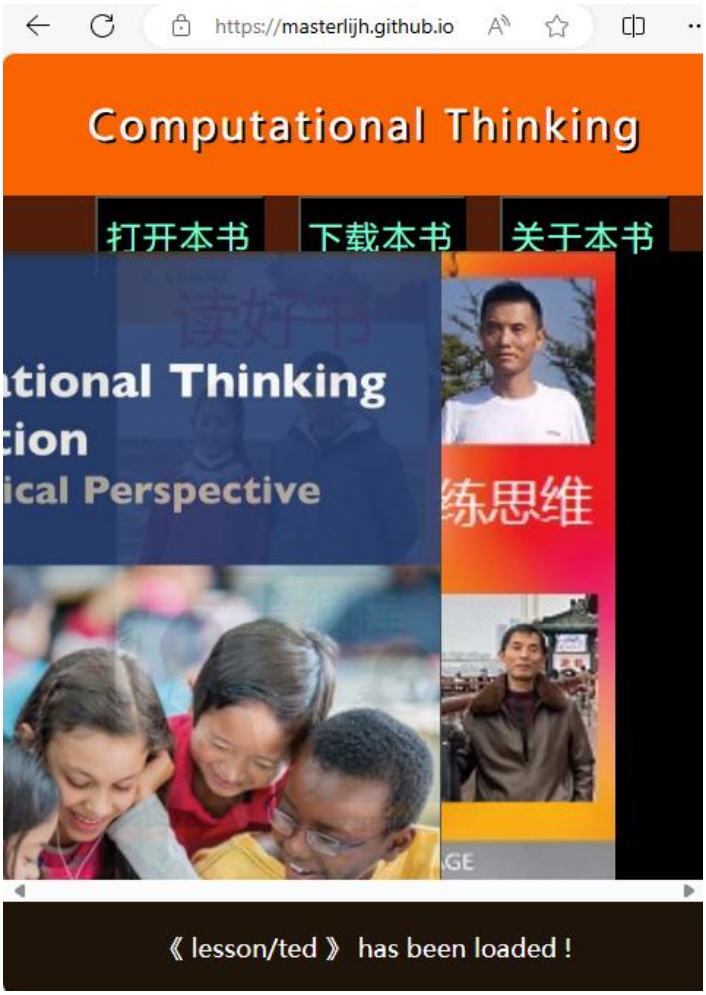


最后，GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。



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

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



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