成 绩：

江 西 科 技 师 范 大 学

毕业设计（论文）

题目（中文）：基于 Web 客户端技术的个性化 UI 设计和实现

（外文）：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：

学　　号：

指导教师：

年　月　日

【摘要】
通读附件的 2 个摘要案例，按自己项目的需要再总结，精炼地写 300 至 400 字。

第 1 章 前言

1.1 项目概述

本项目基于 Web 客户端技术开发，以适应移动互联网时代 Web application 的前端需求。本文通过广泛查阅技术资料和相关文献，特别是 mozilla 组织的 MDN 社区的技术实践文章，通过了 HTML 内容建模、CSS 样式设计和 JavaScript 功能编程的基本技术和技巧。结合本学科的核心课程知识，实现了一个个性化的基于 web 客户端技术的响应式用户界面(UI)，该 UI 能够较好的适配各种屏幕。在功能上，通过 DOM 技术和事件监听实现了对鼠标、触屏和键盘的底层事件的支持并实现了流畅地响应。基于面向对象的思想，为鼠标和触屏设计了对象模型，并通过代码实现了对这类指向性设备的模拟。为了能更好地应用工程思想、设计和开发管理项目，本项目采用了软件工程的增量式开发模式。共进行了 6 次项目迭代开发，每次迭代都经历了 ADIT(A：Analyse 分析 D：Design 设计 Implement：实现 Test：测试)四个经典开发阶段。通过逐步求精的方式编写了 UI 应用程序。最后，为了分享和共享代码，与其他开发者合作，使用了 git 工具进行代码和开发过程日志记录。总共进行了 7 次代码提交操作，详细记录和展现了开发思路和代码优化的过程，然后通过 gitbash 将项目上传到 GitHub 并建立了自己的代码仓库，将该代码仓库设置成为了 HTTP 服务器，实现了全球便捷访问。

1.2 研学计划

表 1 研学计划表

时间	操作
第一周	用三段论的方式完成了页面的设计与搭建
第二周	给页面添加了图片和导航按钮，完善页面
第三周	在页面的主区域中设计了鼠标移动事件，为 PC 端添加了键盘响应区，为后续的键盘事件做好铺垫。

第四周	继续探讨鼠标事件的应用，在页面的主区域添加了鼠标拖动事件。
第五周	在页面主区域为移动端添加了触屏拖动事件，为移动端用户提供方便。并在键盘响应区添加了键盘的监听事件，监听键盘的按下和抬起
第六周	优化了键盘的监听事件，阻止用户按下键盘的功能键，优化了显示效果。

1.3 研究方法

- 为了确保论文的严谨性和科学性，本论文采用了 4 种研究方法。首先采用了文献综述法，通过查询 MDN、课外电子书籍等相关资料文献，了解 HTML、CSS 和 JavaScript 最新标准、最佳实践以及它们在现代 Web 开发中的应用趋势。这一过程不仅为我们提供了坚实的理论基础，还帮助我们明确了研究方向和技术创新点。其次，采用增量开发策略将整体开发工作细分为多个小周期或迭代。每个迭代中，我们专注于设计、实现，这种方法允许我们快速验证想法，及时调整方向，并确保每个功能的稳定性和可靠性，从而有效提升了开发效率和项目管理的灵活性。之后采用 Git 作为版本控制系统，我们对每一次代码提交都进行了详细的记录和管理。这不仅便于团队成员间的协作，还使得代码的历史变更可追溯，错误回滚迅速，保证了代码库的整洁和项目的可持续发展。通过 GitHub 平台，我们实现了代码的版本控制、分支管理及合并请求，促进了代码审查和知识共享。最后采用严格的代码审查机制，每次提交代码之前，都让同学审查这一过程不仅检查代码的语法正确性、逻辑清晰度，还关注代码风格的一致性、性能优化以及潜在的 bug。代码审查促进了团队成员之间的技术交流，提高了代码质量，减少了后期维护成本，确保了软件项目的健壮性和可维护性。

第 2 章 Web 平台和客户端技术概述

Web 平台和客户端技术是构成现代 Web 应用的基础，包含了用户设备上的软件到 Web 内容呈现的整个过程。Web 浏览器不仅是访问网页的工具，它们还是一个复杂的生态系统，支持多种 Web 标准和 API。HTML 用于构建页面的基本结构，通过标签描述内容和意义；CSS 则赋予这些内容视觉样式，控制布局和颜色等外观属性。JavaScript，一种强大的脚本语言，赋予页面动态功能，如用户交互、数据处理和网络通信。

1989 年，蒂姆·伯纳斯-李爵士发明了万维网(见最初的提案)。他创造了“万维网”这个词，并在 1990 年 10 月编写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)“WorldWideWeb”。他的这一系列创举，不仅标志着 Web 时代的到来，更为后续的技术革命埋下了伏笔。

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。在此之后，Web 平台经历了爆炸性的增长，从简单的文档展示进化到支持复杂应用与服务的平台。CSS 的引入，为网页设计带来了革命性的变化，使得开发者能够独立于内容控制页面的外观和布局，创造出丰富多彩的视觉体验。而 JavaScript 的加入，则彻底改变了 Web 的交互模式，使得网页能够响应用户的操作，执行复杂的逻辑，甚至与服务器进行异步通信，为实时应用和动态内容的展现提供了可能。

如今，Web 平台和客户端技术已发展成为一个高度复杂且不断进化的生态系统，包括但不限于先进的 Web API、高性能的前端框架、移动优先的响应式设计、渐进式 Web 应用、以及对安全性和性能优化的持续追求。这些技术的融合，不仅推动了 Web 应用的边界，也重新定义了用户与互联网互动的方式，为数字时代的信息交流与服务提供了无限可能。

第 3 章 项目的增量式迭代开发模式

瀑布模型，作为软件工程领域的一种经典开发方法论，自 20 世纪 70 年代起就被广泛应用，它体现了软件开发过程的线性和顺序性。该模型将软件开发活动划分为一系列连续的阶段，如同瀑布流水般从上游流向下游，每个阶段的输出作为下一阶段的输入，各个阶段依次进行，不可逆向返回或跳过。但是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。

增量式迭代开发，是将庞大的项目一步步分解为一系列较小的、可管理的模块，每个增量模块都代表着产品的一个功能的子集，逐一实现这些增量，确保在每个迭代周期结束时，都能够产出可部署的、带有新功能的软件版本。而这种开发模式更加适合小型开发者，因此

本项目采用该设计模式来完成本项目。

任何的开发模式都需要遵循一下 4 个阶段，分别为分析（Analysis）、设计（Design）、实施（Implementation）、测试（test）。如图 1 所示。本项目中我一共做了六次项目的开发迭代

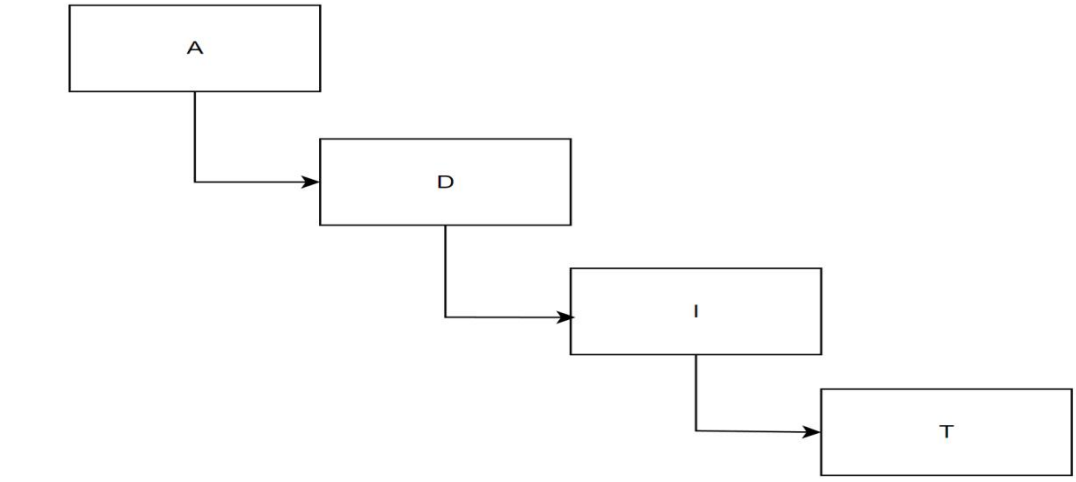


图 1 ADIT 过程

第 4 章 内容设计概要

4.1 分析和设计

这一步是项目的初次开发，在着手设计个性化 UI 之前，构建一个清晰的页面布局是至关重要的基础工作。这一步是设计一个精美的 Html 页面的关键，其中<body>标签内嵌入了三个核心部分：<header>、<main>和<footer>，其中<header>标签内容作为网站的头部，承载着呈现标题的作用，<main>标签内容是作为网站的主题内容区域，在此区域展示后续的鼠标和触屏交互功能。<footer>标签是页面底部，通常是包含版权信息声明。

而为了使这三大块能够在视觉更加清晰，在设计 CSS 样式时，边框不仅起到视觉分隔的作用，还能帮助用户感知页面的流动性和连续性。如**错误！未定义书签。**所示，通过为<header>、<main>和<footer>分别赋予 2 像素宽的蓝色边框，不仅清晰界定了各自的范围，也为页面增添了一抹秩序与和谐。这样的设计选择不仅强化了布局的逻辑性，还使得各部分内容的区分变得直观，即便是初次访问的用户也能迅速把握页面结构。在 PC 端，边框的存在使得页面结构更加清晰，

而在移动端，通过媒体查询和响应式设计，边框可以适配不同的屏幕尺寸，确保内容在任何设备上都能保持一致的布局。

图 4-4 侧重于 PC 端的展示，清晰地展现了边框界定的每个区域，以及如何在保持美观的同时，确保信息的有效组织。而图 4-5，则聚焦于移动端的响应式呈现，展示在屏幕尺寸变化时，边框如何灵活调整，维持页面结构的完整性，确保不同设备上的用户体验连贯一致。

综上所述，通过这样合理的 Html 页面概要设计和 CSS 样式设计，为个性化 UI 设计打下了坚实的基础，能够帮助设计师构建清晰、连贯的网页结构。无论用户使用何种设备，都能享受到一致且优质的在线体验。这不仅提升了页面的可读性和用户友好度，更为后续的创意设计和交互优化铺平了道路，确保网站在视觉表现和功能实用性上都能达到预期目标。首页的 URL 地址如图 4-6。
页面展示如图 4-1 用例图所示：

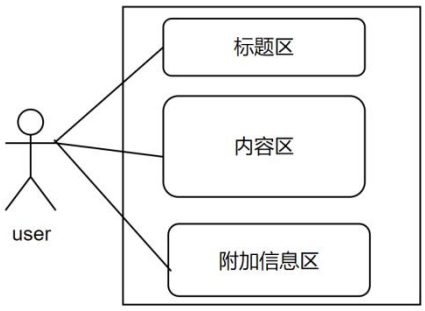


图 4-1 用例图

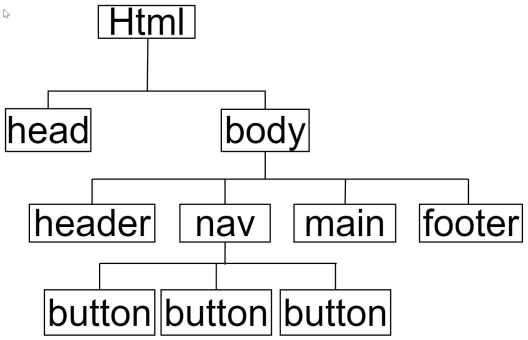


图 4-2 Dom 树

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

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

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

图 4-3 三大块的 CSS 样式

4.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
   现代电影赏析
</header>
<nav>
  <button>导航一</button>
  <button>导航二</button>
  <button>导航三</button>
</nav>
 <main>
  内容区域
 </main>

<footer>
CopyRight from 杨益 江西科技师范大学 2022--2025
</footer>
```

二、CSS 代码编写如下：

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

  header {
    height: 80px;
    border: 2px solid blue;
  }
  main {
    height: 300px;
    border: 2px solid blue;
  }
  nav {
    border: 2px solid blue;
    height: 50px;
  }

  footer {
    height: 50px;
    border: 2px solid blue;
  }
```

4.3 项目的运行和测试

项目的运行和测试至少要通过两类终端，本文此处给出 PC 端和移动端用 Chrome 浏览器打开项目的结果，如下图 4-4 和图 4-5 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 4-6 的二维码，运行测试本项目的第一次开发的阶段性效果。

图 4-4 PC 端内容概要设计

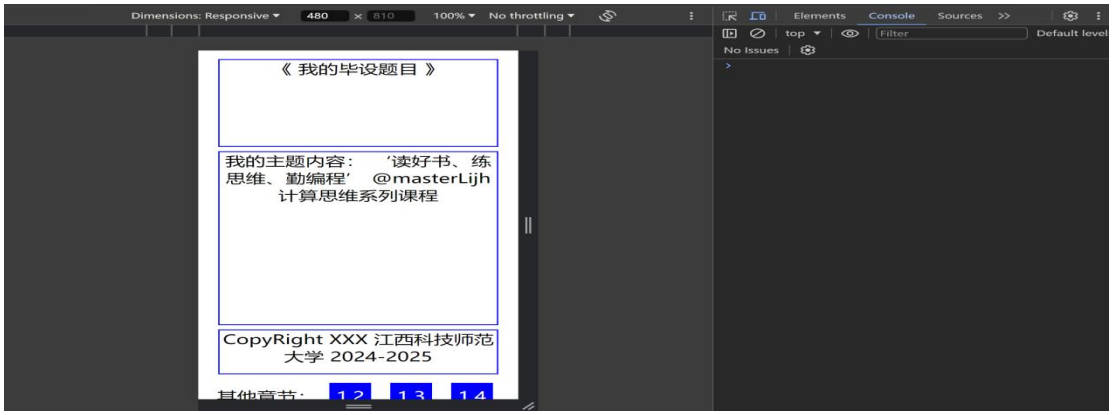


图 4-5 移动端内容概要设计

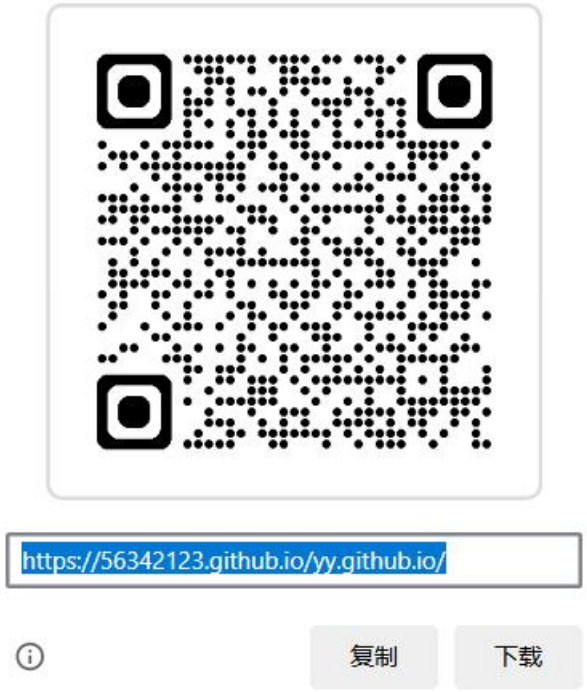


图 4-6 首页页面 URL

4.4 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。

```
77367@LAPTOP-3SL33RJ4 MINGW64 /d/code/homework/homework (master)
$ git config user.name yangyi

77367@LAPTOP-3SL33RJ4 MINGW64 /d/code/homework/homework (master)
$ git config user.eamil yangyi

77367@LAPTOP-3SL33RJ4 MINGW64 /d/code/homework/homework (master)
$ git commit -m '第一次提交主页'
Author identity unknown
```

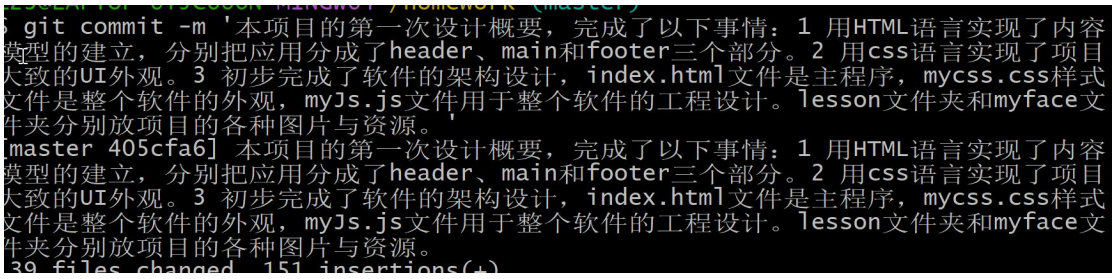

```
77367@LAPTOP-3SL33RJ4 MINGW64 /d/code/homework/homework (master)
$ git config user.name yangyi

77367@LAPTOP-3SL33RJ4 MINGW64 /d/code/homework/homework (master)
$ git config user.eamil yangyi

77367@LAPTOP-3SL33RJ4 MINGW64 /d/code/homework/homework (master)
$ git commit -m '第一次提交主页'
Author identity unknown
```


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
$ git commit -m '第一次提交主页'
[master (root-commit) 6a374c5] 第一次提交主页
 1 file changed, 61 insertions(+)
 create mode 100644 index.html
```

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，分别把应用分成了 header、main 和 footer 三个部分。2 用 css 语言实现了项目大致的 UI 外观。3 初步完成了软件的架构设计，index.html 文件是主程序，mycss.css 样式文件是整个软件的外观，myJs.js 文件用于整个软件的工程设计。lesson 文件夹和 myface 文件夹分别放项目的各种图片与资源。#添加提交的信息

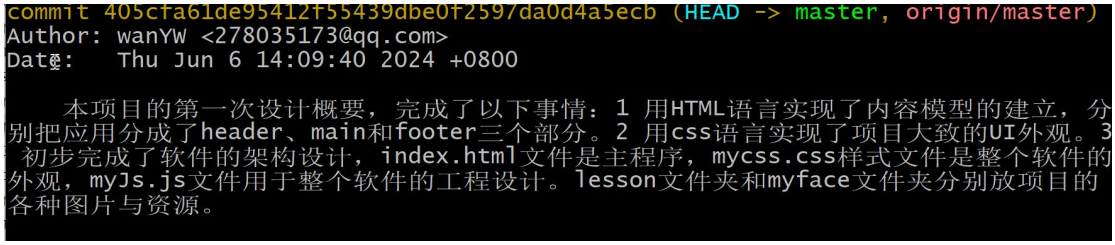
成功提交代码后，gitbash 的反馈如下所示：



项目代码仓库自此也开启了严肃的历史记录，我们可以输入日志命令查看，

```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：



第 5 章 移动互联时代的响应式设计和窄屏代码实现

5.1 分析和设计

2010 年，Ethan Marcotte 利用了三种工具：流动布局(fluid grids)，媒体查询(media queries)和弹性图片(scalable images)，创造了一个在不同分辨率设备下都能友好的自行调整显示规则的网站[8]。发表在文章“Responsive Web Design”中，译为“响应式网页设计”，这是一篇开创性的文章，响应式布局的概念就此诞生[9]。

在这移动互联网时代，响应式设计是网页设计的关键所在，尤其是针对于不同设备之间屏幕大小不同，响应式设计可以确保内容在不同的设备之间都能够呈现出良好的视觉效果与交互体验。尤其是在窄屏设备如智能手机和平板电脑上，屏幕尺寸的变化给网页设计带来了新的挑战。响应式设计通过动态调整布局、图像大小和交互元素，实现了在不同屏幕尺寸下内容的完美呈现。在构建响应式布局时，<header>、<main>和<footer>等元素的布局调整至关重要。以百分比设置这些元素的高度，是实现这种适应性的有效方法。例如，我们可以为<header>、<main>和<footer>设置 CSS 样式，比如说如图 5-3 所示，为<header>、<main>、<footer>都添加了 CSS 样式 height 高度，值都是相对于父元素 body 的高度的百分比。这样的设计保证了即使屏幕的高度不断变化，用户依然可以清晰的看到页面头部、主题区域以及底部。同时，百分比高度的设定也允许内容自适应填充，避免了在不同屏幕尺寸下出现空白或溢出的问题。字体大小的响应式调整同样不可或缺。在移动设备上，阅读体验的优化直接关系到信息的有效传达。通过 JavaScript 或 CSS 变量，依据视口宽度动态设定字体大小，确保文本既不会因字号过小而难以阅读，也不会过大到破坏布局的美感。故采用如图 5-4 所示的方式，先创建一个 UI 的全局变量，为其属性 appWidth 和 appHeight 设置值为视图的宽度和高度，然后将基础字体设置为宽度的二十分之一，确保一行最多只能有 20 个字，可以确保文本行长度适宜，提升阅读的舒适度。　　交互设计在窄屏环境下的响应式策略同样关键。增加如<nav>元素，并在其中精心设计功能性按钮，如导航菜单，不仅是为了美观，更是为了提升操作的便捷性。故在<header>和<main>标签之间增加了<nav>标签在里面存放一些功能性按钮，如图 5-5 所示，添加了导航一、导航二和导航三按钮，使得有限的屏幕空间得以高效利用，同时保持界面的简洁与直观。

响应式设计不仅仅关乎视觉效果，更关乎如何在有限的屏幕空间内提供高效、直观的交互。它要求设计者从用户的角度出发，充分考虑不同场景下的使用需求，通过灵活的布局、智能的内容调整和人性化的交互设计，构建出适应性强、体验优良的网页应用，通过细致的布局调整、内容适应和交互设计，我们可以确保网页在移动互联网时代的各种设备上都能呈现出优秀的用户体验。页面 URL 如图 5-7。

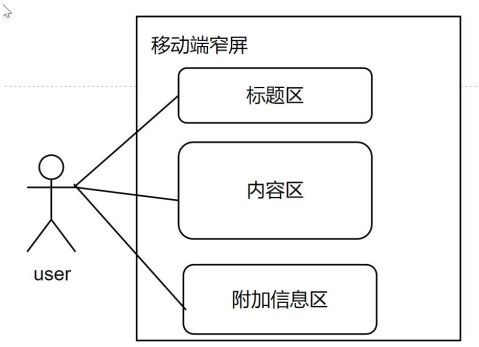


图 5-1 用例图

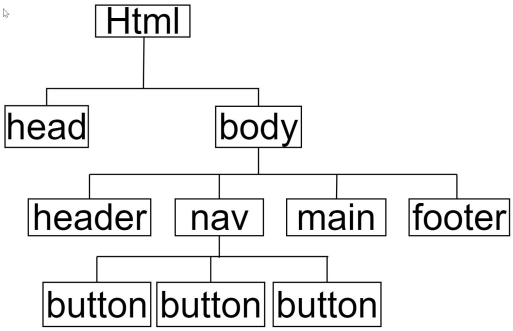


图 5-2 Dom 树

```css
header {
    height: 15%;
    border: 2px solid blue;
    font-size: 1.6em;
}

main {
    height: 70%;
    border: 2px solid blue;
    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 {
    min-height: 5%;
    border: 2px solid blue;
}
```

图 5-3 响应式设计 CSS 样式

```javascript
var UI = {};
// 设置宽度
UI.appWidth = window.innerWidth;
// 设置高度
UI.appHeight = window.innerHeight;
// 基础字体的大小
let baseFont = parseInt(UI.appWidth / 20) + 'px';
// 通过修改body对象的字体大小，可以遗传给子代，从而实现自己的响应式设计
document.body.style.fontSize = baseFont;
// 把body的高度设置为屏幕的高度，实现了纵向全屏
// 通过CSS对body子对象的高度百分比进行分配，从而达到响应式设计的目标。
document.body.style.height = UI.appHeight - 70 + 'px';
```

图 5-4 字体的响应式设计

5.2 项目的实现和编程

一、HTML 代码编写如下：

```html
<header>
  现代电影赏析
</header>
<nav>
  <button>导航一</button>
```

```
  <button>导航二</button>
  <button>导航三</button>
</nav>
<main id="main">
  <div id="bookface">
    书的封面
  </div>
</main>
<footer>
  杨益 江西科技师范大学 2022--2025
</footer>
<div id="aid">
  用户键盘响应区
</div>
```

二、CSS 代码编写如下：

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

  body {
    position: relative;
  }
  header {
    height: 15%;
    border: 2px solid blue;
    font-size: 1.6em;
  }
  main {
    height: 70%;
    border: 2px solid blue;
    font-size: 1.2em;
    /* background-image: url('../lesson/CS.jpg'); */
    background-size: cover;
  }
  #bookface {
    width: 80%;
    height: 80%;
    border: 1px solid red;
    background-color: blanchedalmond;
    margin: auto;
  }
```

```
    nav {
      border: 2px solid blue;
      height: 10%;
      font-size: 1.1em;
    }
    footer {
      min-height: 5%;
      border: 2px solid blue;
    }
    #aid {
      position: absolute;
      left: 600px;
      top: 0;
      border: 3px solid blue;
    }
```

三、JS 代码编写如下：

```
 var UI = {};
    // 设置宽度  如果宽度大于 600，就只显示 600 宽度
    UI.appWidth = window.innerWidth >= 600 ? 600 :
window.innerWidth;
    // 设置高度
    UI.appHeight = window.innerHeight;
    // 基础字体的大小
    let baseFont = parseInt(UI.appWidth / 20);

    // 设置宽高
    setBodyStyle(baseFont, UI.appHeight, UI.appWidth);
    setAidStyle(UI.appWidth, UI.appHeight);
    // 尝试对鼠标设计 UI 控制
    var Mouse = {
      isDown: false,
      x: 0,
      deltaX: 0
    }
    $('bookface').addEventListener('mousedown', function (ev) {
      Mouse.isDown = true;
      let x = ev.pageX;
      let y = ev.pageY;
      console.log('鼠标按下了，坐标：(' + x + ',' + y + ')')
      $('bookface').textContent = '鼠标按下了，坐标：(' + x + ',' +
y + ')'
    })
    $('bookface').addEventListener('mousemove', function (ev) {
```

```
      let x = ev.pageX;
      let y = ev.pageY;
      console.log('鼠标正在移动')
      $('bookface').textContent = '鼠标正在移动，坐标：(' + x + ','
+ y + ')'
    })
    $('bookface').addEventListener('mouseup', function (ev) {
      let x = ev.pageX;
      let y = ev.pageY;
      console.log('鼠标抬起了，坐标：(' + x + ',' + y + ')')
      $('bookface').textContent = '鼠标抬起了，坐标：(' + x + ',' +
y + ')'
    })
    // 封装一个获取元素的方法
    function $(element) {
      if (typeof element !== 'string'
        || element === ''
        || element === null
        || element === undefined) {
        throw new Error('参数错误');
      }
      let dom = document.getElementById(element);
      if (!dom) {
        // 如果不存在
        dom = document.querySelector(element);
      }
      if (dom) {
        return dom;
      }
      // 如果不存在，就抛异常
      throw ('id 或选择器不存在');
    }
    /**
     * 给 body 设置样式
     * @param { number } baseFont 基础字体
     * @param { number } appHeight 设备高度
     * @param { number } appWidth 设备宽度
     */
    function setBodyStyle(baseFont, appHeight, appWidth) {
      // 通过修改 body 对象的字体大小，可以遗传给子代，从而实现自己的
响应式设计
      document.body.style.fontSize = baseFont + 'px';
      // 把 body 的高度设置为屏幕的高度，实现了纵向全屏
```

```
    // 通过 CSS 对 body 子对象的高度百分比进行分配，从而达到响应式设计的目标。
    document.body.style.height = appHeight - 70 + 'px';
    document.body.style.width = appWidth + 'px';
  }
  /**
   * 给 aid 设置样式
   * @param { number } appHeight 设备高度
   * @param { number } appWidth 设备宽度
   */
  function setAidStyle(appWidth, appHeight) {
    let aid = document.getElementById('aid');
    if (window.innerWidth <= 1000) {
      aid.style.display = 'none';
    }
    aid.style.width = window.innerWidth - appWidth - 30 + 'px';
    aid.style.height = appHeight - 62 + 'px';
  }
```

5.3 项目的运行和测试

项目的运行和测试至少要通过两类终端，本文此处给出移动端和 PC 端用 Chrome 浏览器打开项目的结果，如下图 5-5 和图 5-6 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 5-7 的二维码，运行测试本项目的第二次开发的阶段性效果。

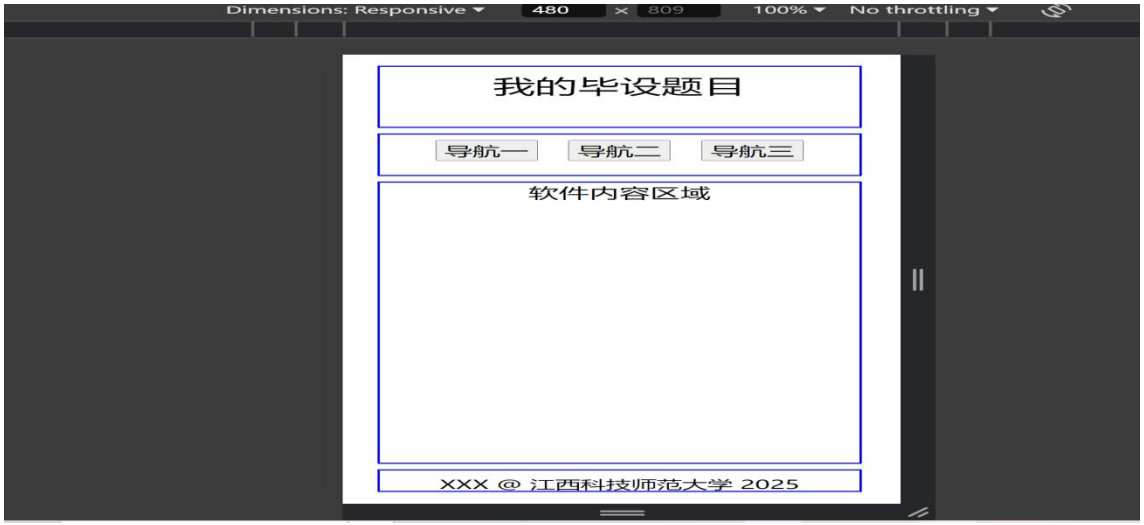


图 5-5 移动端响应式设计(窄屏)

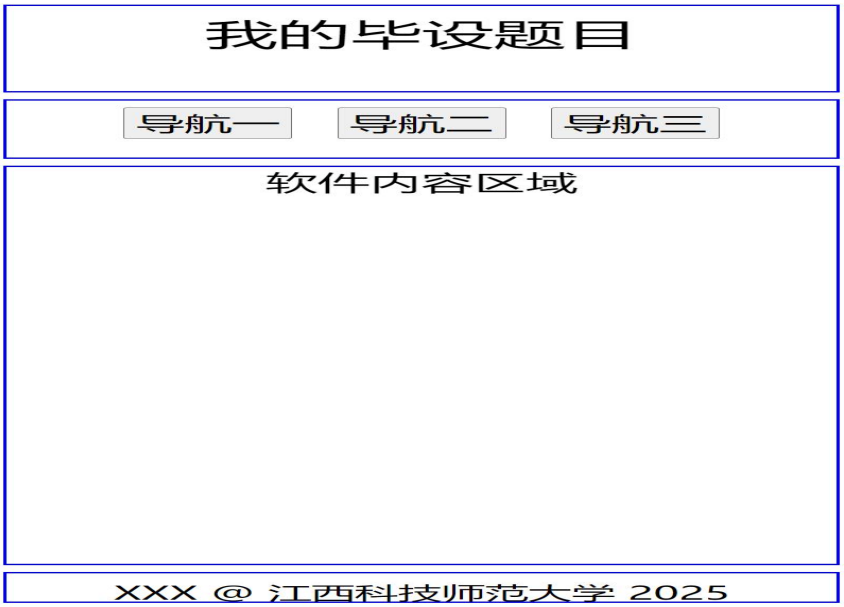


图 5-6 PC 端响应式设计

图 5-7 页面 URL

5.4 项目的代码提交和版本管理

本次提交作为第二次正式提交，在代码提交和版本管理环节，在 commit 提交的过程中，要做好文件代码的描述，方便后续代码查看。

编写好 1.2.html 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add . #添加所有修改过的文件
$ git commit -m '本次代码提交初步完成了响应式设计。代码实施分以下几步：1.为软件 4 个区域分配了高度的比率，通过把设备的高度全部分配给了 body 对象，结合前面的高度分配，实现各区域的响应式设计。2.为应用计算了基础字体的大小，并利用 body 的遗传
```

机制，结合 CSS 的字体的相对控制，实现了字体的响应式设计。'

成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m '本次代码提交初步完成了响应式设计。代码实施分以下几步：1.为软件4
个区域分配了高度的比率，通过把设备的高度全部分配给了body对象，结合前面的高度分配
，实现各区域的响应式设计。2.为应用计算了基础字体的大小，并利用body的遗传机制，结
合CSS的字体的相对控制，实现了字体的响应式设计。'
[master 3036d33] 本次代码提交初步完成了响应式设计。代码实施分以下几步：1.为软件4
个区域分配了高度的比率，通过把设备的高度全部分配给了body对象，结合前面的高度分配
，实现各区域的响应式设计。2.为应用计算了基础字体的大小，并利用body的遗传机制，结
合CSS的字体的相对控制，实现了字体的响应式设计。
 2 files changed, 146 insertions(+)
 create mode 100644 1.1.html
 create mode 100644 1.2.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：

```
commit 3036d33173707509163791aabe3d9b256eacac37c
Author: wanYW <278035173@qq.com>
Date:   Thu Jun 6 14:44:27 2024 +0800

    本次代码提交初步完成了响应式设计。代码实施分以下几步：1.为软件4个区域分配了
高度的比率，通过把设备的高度全部分配给了body对象，结合前面的高度分配，实现各区域
的响应式设计。2.为应用计算了基础字体的大小，并利用body的遗传机制，结合CSS的字体
的相对控制，实现了字体的响应式设计。
```

第 6 章 适用移动互联时代的响应式设计

6.1 分析和设计

在 UI 设计中，对鼠标模型的深入理解和有效控制是构建用户友好、互动性强的界面的关键要素。上述代码通过 JavaScript 实现了一个实例，让用户能够通过鼠标拖动页面上的“书的封面”元素，从而展示了如何巧妙地利用鼠标事件来控制页面元素的行为，进一步提升了用户体验。

如图 6-3 所示，代码首先创建了一个名为 Mouse 的 JavaScript 对象，用于存储鼠标操作的相关状态。isDown 属性记录鼠标是否被按下，x 存储鼠标点击时的初始 x 坐标，deltaX 则追踪鼠标在水平方向上的移动距离。这些属性构成了实现鼠标拖动的基础，它们实时更新以反映用户与页面的交互，为后续的事件处理提供了必要数据。

接下来，通过 addEventListener 方法，代码监听了四个关键的鼠标事件：mousedown、mousemove、mouseup 和 mouseout。当用户按下鼠标时，mousedown

事件被触发，此时记录下鼠标点击的位置，并更新 Mouse 对象的状态。在 mousemove 事件中，如果鼠标在元素内并且已被按下，元素的位置会跟随鼠标移动，实现拖拽效果。mouseup 事件处理鼠标释放，检查拖动距离是否超过了预设阈值 VALID_DISTANCE，以决定是否执行特定操作，这样可以避免因误触导致的不必要的响应。最后，mouseout 事件用于在鼠标离开元素时，恢复元素到其初始位置，确保用户在结束交互后界面能及时恢复原状。

此设计充分考虑了鼠标模型的各种基本操作和边界情况，不仅包括点击、移动和释放，还关注了交互的精确性和反馈。通过设置 VALID_DISTANCE，代码确保了只有当用户有意进行拖动操作时，才会产生相应的响应，增强了交互的灵敏度和可靠性。

此外，代码还利用 CSS 来调整元素的布局，以适应不同的设备尺寸。#bookface 元素的位置通过 JavaScript 动态计算，确保用户在不同设备上都能以相同的方式与其互动，增强了跨平台的一致性。同时，#aid 元素在屏幕宽度小于或等于 1000 像素的设备上被隐藏，优化了窄屏设备的界面布局，体现了设计者对不同设备特性的敏锐洞察和灵活应对。

综上所述，这段代码充分展示了如何在 UI 设计中对鼠标模型进行分析和控制，以创造丰富且自然的用户交互。通过精确监听和响应鼠标事件，结合 CSS 实现动态布局，设计师能够创建出符合用户习惯、具有深度参与感的界面。这种设计策略不仅适用于传统的桌面环境，同样适用于支持鼠标模拟操作的触屏设备，从而确保了设计的广泛适用性和兼容性。因此，掌握并灵活运用鼠标模型的分析和控制技巧，是构建高质量、跨平台 UI 设计不可或缺的能力。页面 URL 如图 6-6。

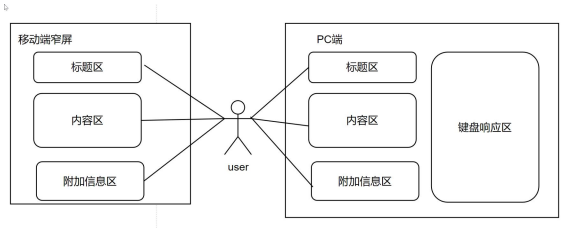


图 6-1 用例图

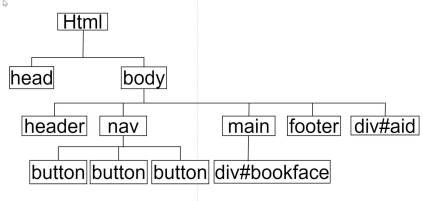


图 6-2 Dom 树

```
// 尝试对鼠标设计 UI 控制
   var Mouse = {
     isDown: false,
     x: 0,
     deltaX: 0
```

```
    }
    const VALID_DISTANCE = 50;
    const ORIGINAL_LEFT = 8 + '%';
    $('bookface').addEventListener('mousedown', function (ev) {
      Mouse.isDown = true;
      let x = ev.pageX;
      let y = ev.pageY;
      // 将鼠标点击的起始位置记录下来
      Mouse.x = x;
      console.log('鼠标按下了，坐标：(' + x + ',' + y + ')')
      $('bookface').textContent = '鼠标按下了，坐标：(' + x + ',' +
y + ')'

    })
    $('bookface').addEventListener('mousemove', function (ev) {
      ev.preventDefault()
      if (Mouse.isDown &&
        Math.abs($('bookface').offsetLeft) < UI.appWidth / 5) {
        let x = ev.pageX;
        let y = ev.pageY;
        Mouse.deltaX = x - Mouse.x;
        $('bookface').style.left = $('bookface').offsetLeft +
Mouse.deltaX + 'px'
      }
    })
    $('bookface').addEventListener('mouseup', function (ev) {
      Mouse.isDown = false
      let x = ev.pageX;
      let y = ev.pageY;
      // 设置有效的拖动距离
      $('bookface').textContent = '鼠标松开，坐标：(' + x + ',' +
y + ')'
      // 如果拖动距离大于有效拖动距离
      if (Math.abs(Mouse.deltaX) > VALID_DISTANCE) {
        $('bookface').textContent = '鼠标松开，这是有效拖动'
      }
      // 复原
      Mouse.x = 0;
      Mouse.deltaX = 0;
      $('bookface').style.left = ORIGINAL_LEFT;
    })
    // 当鼠标移开 bookface 的时候
    $('bookface').addEventListener('mouseout', function (ev) {
```

```
    // 复原
      Mouse.x = 0;
      Mouse.deltaX = 0;
      $('bookface').style.left = 8 + '%';
  })
```

图 6-3 鼠标事件分析

6.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
    现代电影赏析
  </header>
  <nav>
    <button>导航一</button>
    <button>导航二</button>
    <button>导航三</button>
  </nav>
  <main id="main">
    <div id="bookface">
      书的封面
    </div>
  </main>

  <footer>
    万一苇 江西科技师范大学 2022--2025
  </footer>
  <div id="aid">
    用户键盘响应区
  </div>
```

二、CSS 代码编写如下：

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

    body {
      position: relative;
    }
    header {
      height: 15%;
```

```
    border: 2px solid blue;
    font-size: 1.6em;
  }
  main {
    height: 70%;
    border: 2px solid blue;
    font-size: 1.2em;
    /* background-image: url('../lesson/CS.jpg'); */
    background-size: cover;
    position: relative;
  }
  #bookface {
    width: 80%;
    height: 80%;
    border: 1px solid red;
    background-color: blanchedalmond;
    position: absolute;
    left: 8%;
    top: 8%;
  }
  nav {
    border: 2px solid blue;
    height: 10%;
    font-size: 1.1em;
  }
  footer {
    min-height: 5%;
    border: 2px solid blue;
  }
  #aid {
    position: absolute;
    left: 600px;
    top: 0;
    border: 3px solid blue;
  }
```

三、JS 代码编写如下：

```
var UI = {};
  // 设置宽度  如果宽度大于 600，就只显示 600 宽度
  UI.appWidth = window.innerWidth >= 600 ? 600 :
window.innerWidth;
  // 设置高度
  UI.appHeight = window.innerHeight;
```

```
    // 基础字体的大小
    let baseFont = parseInt(UI.appWidth / 20);

    // 设置宽高
    setBodyStyle(baseFont, UI.appHeight, UI.appWidth);
    setAidStyle(UI.appWidth, UI.appHeight);
    // 尝试对鼠标设计 UI 控制
    var Mouse = {
      isDown: false,
      x: 0,
      deltaX: 0
    }
    const VALID_DISTANCE = 50;
    const ORIGINAL_LEFT = 8 + '%';
    $('bookface').addEventListener('mousedown', function (ev) {
      Mouse.isDown = true;
      let x = ev.pageX;
      let y = ev.pageY;
      // 将鼠标点击的起始位置记录下来
      Mouse.x = x;
      console.log('鼠标按下了，坐标：(' + x + ',' + y + ')')
      $('bookface').textContent = '鼠标按下了，坐标：(' + x + ',' +
y + ')'
    })
    $('bookface').addEventListener('mousemove', function (ev) {
      ev.preventDefault()
      if (Mouse.isDown &&
        Math.abs($('bookface').offsetLeft) < UI.appWidth / 5) {
        let x = ev.pageX;
        let y = ev.pageY;
        Mouse.deltaX = x - Mouse.x;
        $('bookface').style.left = $('bookface').offsetLeft +
Mouse.deltaX + 'px'
      }
    })
    $('bookface').addEventListener('mouseup', function (ev) {
      Mouse.isDown = false
      let x = ev.pageX;
      let y = ev.pageY;
      // 设置有效的拖动距离
      $('bookface').textContent = '鼠标松开，坐标：(' + x + ',' +
y + ')'
      // 如果拖动距离大于有效拖动距离
```

```
    if (Math.abs(Mouse.deltaX) > VALID_DISTANCE) {
      $('bookface').textContent = '鼠标松开，这是有效拖动'
    }
    // 复原
    Mouse.x = 0;
    Mouse.deltaX = 0;
    $('bookface').style.left = ORIGINAL_LEFT;
  })
  // 当鼠标移开 bookface 的时候
  $('bookface').addEventListener('mouseout', function (ev) {
    // 复原
      Mouse.x = 0;
      Mouse.deltaX = 0;
      $('bookface').style.left = 8 + '%';
  })
  // 封装一个获取元素的方法
  function $(element) {
    if (typeof element !== 'string'
      || element === ''
      || element === null
      || element === undefined) {
      throw new Error('参数错误');
    }
    let dom = document.getElementById(element);
    if (!dom) {
      // 如果不存在
      dom = document.querySelector(element);
    }
    if (dom) {
      return dom;
    }
    // 如果不存在，就抛异常
    throw ('id 或选择器不存在');
  }
  /**
   * 给 body 设置样式
   * @param { number } baseFont 基础字体
   * @param { number } appHeight 设备高度
   * @param { number } appWidth 设备宽度
   */
  function setBodyStyle(baseFont, appHeight, appWidth) {
    // 通过修改 body 对象的字体大小，可以遗传给子代，从而实现自己的响应式设计
```

```
    document.body.style.fontSize = baseFont + 'px';
    // 把 body 的高度设置为屏幕的高度，实现了纵向全屏
    // 通过 CSS 对 body 子对象的高度百分比进行分配，从而达到响应式设计的目标。
    document.body.style.height = appHeight - 70 + 'px';
    document.body.style.width = appWidth + 'px';
  }
  /**
   * 给 aid 设置样式
   * @param { number } appHeight 设备高度
   * @param { number } appWidth 设备宽度
   */
  function setAidStyle(appWidth, appHeight) {
    let aid = $('aid');
    if (window.innerWidth <= 1000) {
      aid.style.display = 'none';
    }
    aid.style.width = window.innerWidth - appWidth - 30 + 'px';
    aid.style.height = appHeight - 62 + 'px';
  }
```

6.3 项目的运行和测试

项目的运行和测试至少要通过两类终端，本文此处给出移动端和 PC 端用 Chrome 浏览器打开项目的结果，如下图 6-4 和图 6-5 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 6-6 的二维码，运行测试本项目的第三次开发的阶段性效果。

现代电影赏析

导航一 导航二 导航三

鼠标松开! 本次算无效拖动!

杨益 江西科技师范大学 2022--2025

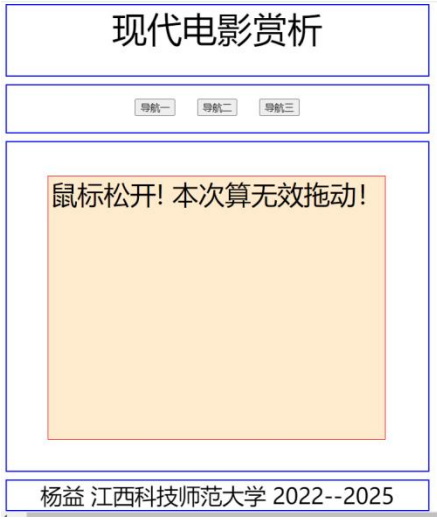


图 6-4 移动端

图 6-5 PC 端

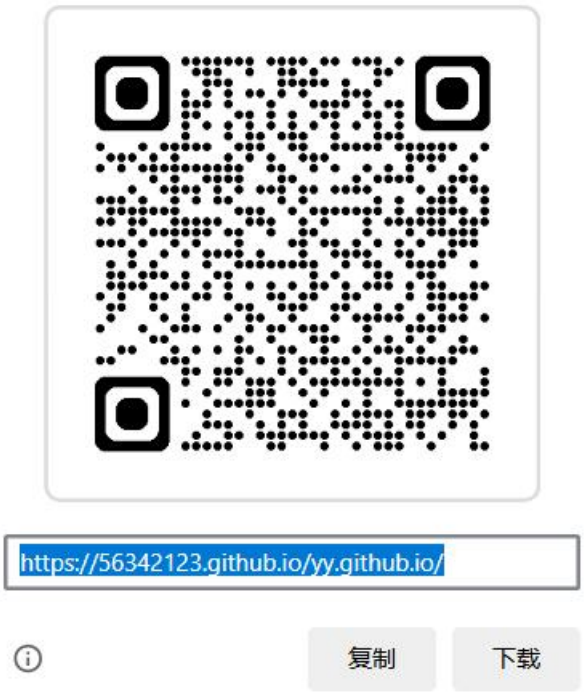


图 6-6 页面 URL

6.4 项目的代码提交和版本管理

相较于上一次提交，本次的代码提交添加了 Mouse 鼠标模型，为后续的鼠标事件和触屏事件增加了做好铺垫。

编写好 1.3.html 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add . #添加所有修改过的文件
$ git commit -m '对屏幕超过 1000 像素的用户屏幕增加了一个用于键盘反馈的 UI 界面，还对主区域书的封面元素设置了鼠标按下、鼠标移动和鼠标抬起做了事件响应。'
```

成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m '对屏幕超过1000像素的用户屏幕增加了一个用于键盘反馈的UI界面，还对主区域书的封面元素设置了鼠标按下、鼠标移动和鼠标抬起做了事件响应。'
[master 8e9f325] 对屏幕超过1000像素的用户屏幕增加了一个用于键盘反馈的UI界面，还对主区域书的封面元素设置了鼠标按下、鼠标移动和鼠标抬起做了事件响应。
 1 file changed, 183 insertions(+)
 create mode 100644 1.3.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：

```
commit 8e9f325d0ab089fa94006eb2ff86b8fed94e64f7 (HEAD -> master, origin/master)
Author: wanYW <278035173@qq.com>
Date:   Thu Jun 6 15:31:33 2024 +0800

    对屏幕超过1000像素的用户屏幕增加了一个用于键盘反馈的UI界面，还对主区域书的封面元素设置了鼠标按下、鼠标移动和鼠标抬起做了事件响应。
```

第 7 章 个性化 UI 设计中鼠标模型

7.1 分析和设计

在 UI 设计中，对鼠标模型的深入理解和有效控制是构建用户友好、互动性强的界面的关键要素。上述代码通过 JavaScript 实现了一个实例，让用户能够通过鼠标拖动页面上的“书的封面”元素，从而展示了如何巧妙地利用鼠标事件来控制页面元素的行为，进一步提升了用户体验。

如图 7-3 所示，代码首先创建了一个名为 Mouse 的 JavaScript 对象，用于存储鼠标操作的相关状态。isDown 属性记录鼠标是否被按下，x 存储鼠标点击时的初始 x 坐标，deltaX 则追踪鼠标在水平方向上的移动距离。这些属性构成了实现鼠标拖动的基础，它们实时更新以反映用户与页面的交互，为后续的事件处理提供了必要数据。

接下来，通过 addEventListener 方法，代码监听了四个关键的鼠标事件：mousedown、mousemove、mouseup 和 mouseout。当用户按下鼠标时，mousedown 事件被触发，此时记录下鼠标点击的位置，并更新 Mouse 对象的状态。在 mousemove 事件中，如果鼠标在元素内并且已被按下，元素的位置会跟随鼠标移动，实现拖拽效果。mouseup 事件处理鼠标释放，检查拖动距离是否超过了预设阈值 VALID_DISTANCE，以决定是否执行特定操作，这样可以避免因误触导致的不必要的响应。最后，mouseout 事件用于在鼠标离开元素时，恢复元素到其初始位置，确保用户在结束交互后界面能及时恢复原状。

此设计充分考虑了鼠标模型的各种基本操作和边界情况，不仅包括点击、移动和释放，还关注了交互的精确性和反馈。通过设置 VALID_DISTANCE，代码确保了只有当用户有意进行拖动操作时，才会产生相应的响应，增强了交互的灵敏度和可靠性。

此外，代码还利用 CSS 来调整元素的布局，以适应不同的设备尺寸。#bookface 元素的位置通过 JavaScript 动态计算，确保用户在不同设备上都能以相同的方式与其互动，增强了跨平台的一致性。同时，#aid 元素在屏幕宽度小于或等于 1000 像素的设备上被隐藏，优化了窄屏设备的界面布局，体现了设计者对不同设备特性的敏锐洞察和灵活应对。

综上所述，这段代码充分展示了如何在 UI 设计中对鼠标模型进行分析和控制，以创造丰富且自然的用户交互。通过精确监听和响应鼠标事件，结合 CSS 实现动态布局，设计师能够创建出符合用户习惯、具有深度参与感的界面。这种设计策略不仅适用于传统的桌面环境，同样适用于支持鼠标模拟操作的触屏设备，从而确保了设计的广泛适用性和兼容性。因此，掌握并灵活运用鼠标模型的分析和控制技巧，是构建高质量、跨平台 UI 设计不可或缺的能力。页面 URL 如**错误！未定义书签。**

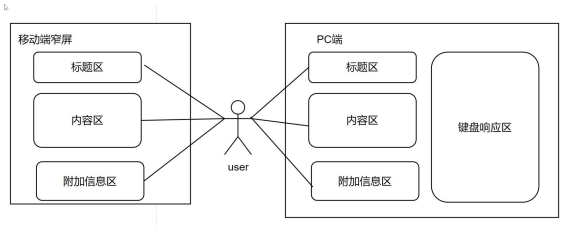


图 7-1 用例图

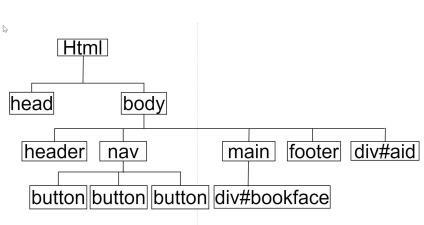


图 7-2 Dom 树

```
// 尝试对鼠标设计 UI 控制
  var Mouse = {
    isDown: false,
    x: 0,
    deltaX: 0
  }
  const VALID_DISTANCE = 50;
  const ORIGINAL_LEFT = 8 + '%';
  $('bookface').addEventListener('mousedown', function (ev) {
    Mouse.isDown = true;
    let x = ev.pageX;
```

```
      let y = ev.pageY;
      // 将鼠标点击的起始位置记录下来
      Mouse.x = x;
      console.log('鼠标按下了，坐标：(' + x + ',' + y + ')')
      $('bookface').textContent = '鼠标按下了，坐标：(' + x + ',' +
y + ')'

    })
    $('bookface').addEventListener('mousemove', function (ev) {
      ev.preventDefault()
      if (Mouse.isDown &&
        Math.abs($('bookface').offsetLeft) < UI.appWidth / 5) {
        let x = ev.pageX;
        let y = ev.pageY;
        Mouse.deltaX = x - Mouse.x;
```

```
        $('bookface').style.left = $('bookface').offsetLeft +
Mouse.deltaX + 'px'
      }
    })
    $('bookface').addEventListener('mouseup', function (ev) {
      Mouse.isDown = false
      let x = ev.pageX;
      let y = ev.pageY;
      // 设置有效的拖动距离
      $('bookface').textContent = '鼠标松开，坐标：(' + x + ',' +
y + ')'
      // 如果拖动距离大于有效拖动距离
      if (Math.abs(Mouse.deltaX) > VALID_DISTANCE) {
        $('bookface').textContent = '鼠标松开，这是有效拖动'
      }
      // 复原
      Mouse.x = 0;
      Mouse.deltaX = 0;
      $('bookface').style.left = ORIGINAL_LEFT;
    })
    // 当鼠标移开 bookface 的时候
    $('bookface').addEventListener('mouseout', function (ev) {
      // 复原
        Mouse.x = 0;
        Mouse.deltaX = 0;
        $('bookface').style.left = 8 + '%';
    })
```

图 7-3 鼠标事件分析

7.2 项目的实现和编程

一、HTML 代码编写如下：

```
 <header>
  现代电影赏析
</header>
<nav>
  <button>导航一</button>
  <button>导航二</button>
  <button>导航三</button>
</nav>
<main id="main">
  <div id="bookface">
    书的封面
  </div>
</main>

<footer>
  万一苇 江西科技师范大学 2022--2025
</footer>
<div id="aid">
  用户键盘响应区
</div>
```

二、CSS 代码编写如下：

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

  body {
    position: relative;
  }
  header {
    height: 15%;
    border: 2px solid blue;
    font-size: 1.6em;
  }
  main {
```

```
    height: 70%;
    border: 2px solid blue;
    font-size: 1.2em;
    /* background-image: url('../lesson/CS.jpg'); */
    background-size: cover;
    position: relative;
  }
  #bookface {
    width: 80%;
    height: 80%;
    border: 1px solid red;
    background-color: blanchedalmond;
    position: absolute;
    left: 8%;
    top: 8%;
  }
  nav {
    border: 2px solid blue;
    height: 10%;
    font-size: 1.1em;
  }
  footer {
    min-height: 5%;
    border: 2px solid blue;
  }
  #aid {
    position: absolute;
    left: 600px;
    top: 0;
    border: 3px solid blue;
  }
```

三、JS 代码编写如下：

```
var UI = {};
    // 设置宽度  如果宽度大于 600，就只显示 600 宽度
    UI.appWidth = window.innerWidth >= 600 ? 600 :
window.innerWidth;
    // 设置高度
    UI.appHeight = window.innerHeight;
    // 基础字体的大小
    let baseFont = parseInt(UI.appWidth / 20);
```

```
    // 设置宽高
    setBodyStyle(baseFont, UI.appHeight, UI.appWidth);
    setAidStyle(UI.appWidth, UI.appHeight);
    // 尝试对鼠标设计 UI 控制
    var Mouse = {
      isDown: false,
      x: 0,
      deltaX: 0
    }
    const VALID_DISTANCE = 50;
    const ORIGINAL_LEFT = 8 + '%';
    $('bookface').addEventListener('mousedown', function (ev) {
      Mouse.isDown = true;
      let x = ev.pageX;
      let y = ev.pageY;
      // 将鼠标点击的起始位置记录下来
      Mouse.x = x;
      console.log('鼠标按下了，坐标：(' + x + ',' + y + ')')
      $('bookface').textContent = '鼠标按下了，坐标：(' + x + ',' +
y + ')'
})
    $('bookface').addEventListener('mousemove', function (ev) {
      ev.preventDefault()
      if (Mouse.isDown &&
        Math.abs($('bookface').offsetLeft) < UI.appWidth / 5) {
        let x = ev.pageX;
        let y = ev.pageY;
        Mouse.deltaX = x - Mouse.x;
        $('bookface').style.left = $('bookface').offsetLeft +
Mouse.deltaX + 'px'
      }
    })
    $('bookface').addEventListener('mouseup', function (ev) {
      Mouse.isDown = false
      let x = ev.pageX;
      let y = ev.pageY;
      // 设置有效的拖动距离
      $('bookface').textContent = '鼠标松开，坐标：(' + x + ',' +
y + ')'
      // 如果拖动距离大于有效拖动距离
      if (Math.abs(Mouse.deltaX) > VALID_DISTANCE) {
        $('bookface').textContent = '鼠标松开，这是有效拖动'
      }
```

```
    // 复原
    Mouse.x = 0;
    Mouse.deltaX = 0;
    $('bookface').style.left = ORIGINAL_LEFT;
  })
  // 当鼠标移开 bookface 的时候
  $('bookface').addEventListener('mouseout', function (ev) {
    // 复原
      Mouse.x = 0;
      Mouse.deltaX = 0;
      $('bookface').style.left = 8 + '%';
  })
  // 封装一个获取元素的方法
  function $(element) {
    if (typeof element !== 'string'
      || element === ''
      || element === null
      || element === undefined) {
      throw new Error('参数错误');
    }
    let dom = document.getElementById(element);
    if (!dom) {
      // 如果不存在
      dom = document.querySelector(element);
    }
    if (dom) {
      return dom;
    }
    // 如果不存在，就抛异常
    throw ('id 或选择器不存在');
  }
  /**
   * 给 body 设置样式
   * @param { number } baseFont 基础字体
   * @param { number } appHeight 设备高度
   * @param { number } appWidth 设备宽度
   */
  function setBodyStyle(baseFont, appHeight, appWidth) {
    // 通过修改 body 对象的字体大小，可以遗传给子代，从而实现自己的响应式设计
    document.body.style.fontSize = baseFont + 'px';
    // 把 body 的高度设置为屏幕的高度，实现了纵向全屏
```

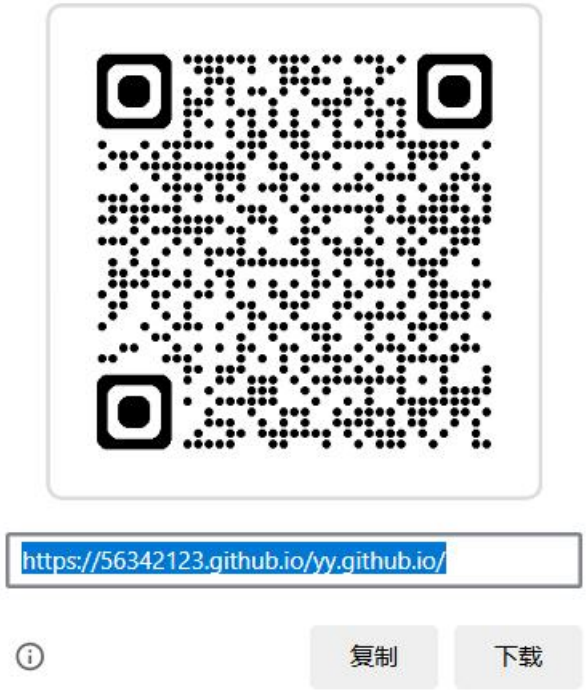
```
    // 通过 CSS 对 body 子对象的高度百分比进行分配，从而达到响应式设计的目标。
    document.body.style.height = appHeight - 70 + 'px';
    document.body.style.width = appWidth + 'px';
  }
  /**
   * 给 aid 设置样式
   * @param { number } appHeight 设备高度
   * @param { number } appWidth 设备宽度
   */
  function setAidStyle(appWidth, appHeight) {
    let aid = $('aid');
    if (window.innerWidth <= 1000) {
      aid.style.display = 'none';
    }
    aid.style.width = window.innerWidth - appWidth - 30 + 'px';
    aid.style.height = appHeight - 62 + 'px';
  }
```

7.3 项目的运行和测试

项目的运行和测试只需要通过一类终端，本文此处给出 PC 端用 Chrome 浏览器打开项目的结果，如下图 6-4 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 7-5 的二维码，运行测试本项目的第三次开发的阶段性效果。

图 7-4 PC 端

图 7-5 页面 URL



7.4 项目的代码提交和版本管理

编写好 1.3.html 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add . #添加所有修改过的文件
$ git commit -m '在 1.4.html 内增加了一个鼠标模型 Mouse 用来模拟鼠标的操作数据，并根据鼠标模型和用户操作设计了 UI 反馈。另外增加了底层函数$用于快速抓取页面元素对象， 在代码中已经反复使用$函数。'
```

成功提交代码后，gitbash 的反馈如下所示：

```
123@LAPTOP-UTJC666N MINGW64 /homework (master)
$ git commit -m '在1.4.html内增加了一个鼠标模型Mouse用来模拟鼠标的操作数据，并根
据鼠标模型和用户操作设计了UI反馈。另外增加了底层函数$用于快速抓取页面元素对象，
在代码中已经反复使用$函数。'
[master fbb2997] 在1.4.html内增加了一个鼠标模型Mouse用来模拟鼠标的操作数据，并根
据鼠标模型和用户操作设计了UI反馈。另外增加了底层函数$用于快速抓取页面元素对象，
在代码中已经反复使用$函数。
 1 file changed, 212 insertions(+)
 create mode 100644 1.4.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：

```
commit fbb299751f3e6157b6a4b2f357c1d8165a30edb7 (HEAD -> master, origin/master)
Author: wanYW <278035173@qq.com>
Date:   Thu Jun 6 15:47:02 2024 +0800

    在1.4.html内增加了一个鼠标模型Mouse用来模拟鼠标的操作数据，并根据鼠标模型和
用户操作设计了UI反馈。另外增加了底层函数$用于快速抓取页面元素对象，在代码中已经
反复使用$函数。

commit 8e9f325d0ab089fa94006eb2ff86b8fed94e64f7
```

第 8 章 通用的 UI 设计

8.1 分析和设计

在 Web 开发中，创建一套能够同时适应触屏和鼠标操作的通用用户界面（UI）至关重要。这样的设计不仅优化了用户体验，减少了开发者在不同平台上的重复工作，还提升了整体应用的性能和维护性。通过精心设计和编程，我们可以确保无论是 PC 端的鼠标用户还是移动端的触屏用户，都能流畅地与网站或应用进行交互。响应式设计是实现跨平台兼容性的基础。我们需要通过 CSS 样式和使用 JavaScript 来根据设备动态的对 Dom 元素设置高度、宽度以及位置，以适应各种屏幕尺寸和输入方式。

代码如图 8-3 所示，首先先创建一个 Pointer 对象用来表示鼠标或触屏的指针对象来抽象鼠标和触屏的交互，并为该对象添加 3 个属性分别为 isDown 代表着鼠标或触屏是否按下，x 代表着鼠标或触屏按下的横坐标位置，deltaX 代表着鼠标或触屏最后移动的位置与 x 之间的差值，这个 Pointer 对象可以作为一个全局变量，以便在任何地方访问和更新。然后编写 3 个方法分别为 handleBegin、handleMoving 以及 handleEnd，这三个方法分别代表着处理开始交互的事件，如 mousedown 或 touchstart。它将 isDown 设置为 true，并记录下开始交互时的横坐标位置。处理交互过程中的移动事件，如 mousemove 或 touchmove。如果 isDown 为 true，则更新 deltaX 以反映指针或触摸点的移动。处理结束交互的事件，如 mouseup 或 touchend。它将 isDown 重置为 false。之后为<body>元素添加 6 个监听事件分别监听鼠标的开始、移动和结束以及触屏的开始、移动和结束。

这样，我们就可以根据 Pointer 对象的状态来处理用户交互，无论他们使用的是鼠标还是触屏。例如，我们可以使用 deltaX 来判断用户是向左还是向右滑动，从而触发相应的功能，如切换页面等。然后为了减少全局变量的影响，我们可以将这些代码封装在函数或者模块中，保证代码的整洁和可维护性。通过这种方式，我们确保了在 PC 和移动端的交互一致性，提高了用户在不同设备上的体验。同时，由于 Pointer 对象作为全局变量，可以在整个应用中共享和更新，简化了代码的维护和扩展。

通过以上策略，我们能够创建一个既美观又易用的 Web 应用，无论用户使用鼠标还是触屏，都能享受到一致且流畅的交互体验。这样的通用 UI 设计不仅提升了用户体验，也简化了开发流程，使得维护和扩展变得更加容易。在现代

Web 开发中，这种跨设备、跨平台的设计方法已经成为必不可少的一部分，它体现了对用户需求的深刻理解和对技术的灵活运用。页面 URL 如**错误！未定义书签。**所示。

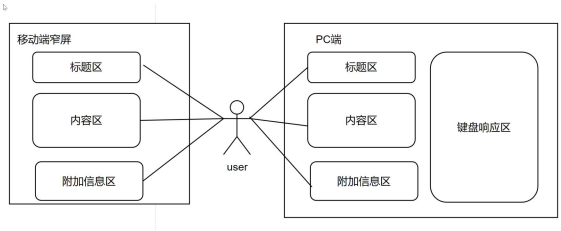


图 8-1 用例图

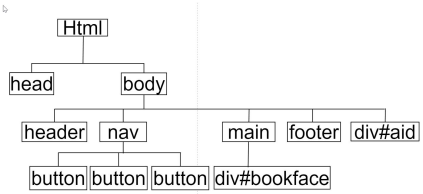


图 8-2 Dom 树

```
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
```

图 8-3 鼠标和触屏事件

8.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
 <p id="book">
  《我的毕设题目》
 </p>
</header>
<nav>
 <button>向前</button>
 <button>向后</button>
 <button>其他</button>
</nav>


<main id="main">
 <div id="bookface">
  这是书的封面图<br>
  在此对象范围拖动鼠标/滑动触屏<br>
  拖动/滑动超过 100 像素，视为有效 UI 互动!
 </div>
</main>
<footer>
 CopyRight XXX 江西科技师范大学 2024--2025
</footer>
<div id="aid">
 <p>用户键盘响应区</p>
</div>
```

四、CSS 代码编写如下：

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

 header {
  border: 3px solid green;
  height: 10%;
```

```
 font-size: 1em;
}
nav {
 border: 3px solid green;
 height: 10%;
}
main {
 border: 3px solid green;
 height: 70%;
 font-size: 0.8em;
 position: relative;
}
#box {
 position: absolute;
 right: 0;
 width: 100px;
}
footer {
 border: 3px solid green;
 height: 10%;
 font-size: 0.7em;
}
body {
 position: relative;
}
button {
 font-size: 1em;
}
#aid {
 position: absolute;
 border: 3px solid blue;
 top: 0px;
 left: 600px;
}
```

```
#bookface {
 position: absolute;
 width: 80%;
 height: 80%;
 border: 1px solid red;
 background-color: blanchedalmond;
 left: 7%;
 top: 7%;
```

```
}
```

8.3 项目的运行和测试

项目的运行和测试至少要通过两类终端，本文此处给出移动端和 PC 端用 Chrome 浏览器打开项目的结果，如下图 8-4 和图 8-5 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 8-6 的二维码，运行测试本项目的第四次开发的阶段性效果。

图 8-4 移动端页面展示

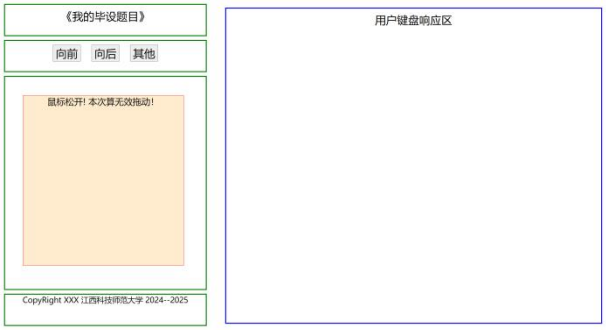


图 8-5 PC 端页面展示

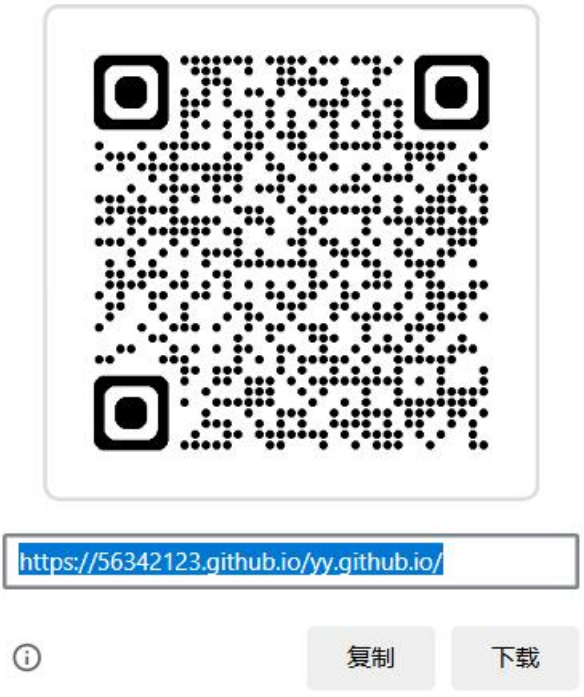


图 8-6 页面 URL

8.4 项目的代码提交和版本管理

编写好 1.5.html 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add . #添加所有修改过的文件
$ git commit -m ' 为鼠标和触屏设计一套代码实现 UI 控制，创建了一个 Pointer 对象，分别存放开始的位置、最后的位置以及是否按下的属性，添加 movedown touchstart mouseup touchend mouseout mousemove touchmove 的监听事件。并为键盘响应区监听了 keyPress 事件，为后续的键盘事件做好铺垫。'
```

成功提交代码后，gitbash 的反馈如下所示：

```
123@LAPTOP-UTJC666N MINGW64 /homework (master)
$ git commit -m '为鼠标和触屏设计一套代码实现UI控制，创建了一个Pointer对象，分别存放开始的位置、最后的位置以及是否按下的属性，添加movedown touchstart mouseup touchend mouseout mousemove touchmove的监听事件。并为键盘响应区监听了keyPress事件，为后续的键盘事件做好铺垫。'
[master c8b5bc3] 为鼠标和触屏设计一套代码实现UI控制，创建了一个Pointer对象，分别存放开始的位置、最后的位置以及是否按下的属性，添加movedown touchstart mouseup touchend mouseout mousemove touchmove的监听事件。并为键盘响应区监听了keyPress事件，为后续的键盘事件做好铺垫。
 1 file changed, 226 insertions(+)
 create mode 100644 1.5.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：

```
$ git log
commit c8b5bc31323f8b5926ce9de8f7c4316a615e7ec9 (HEAD -> master, origin/master)
Author: wanYW <278035173@qq.com>
Date:   Thu Jun 6 16:54:57 2024 +0800

    为鼠标和触屏设计一套代码实现UI控制，创建了一个Pointer对象，分别存放开始的位置、最后的位置以及是否按下的属性，添加movedown touchstart mouseup touchend mouseout mousemove touchmove的监听事件。并为键盘响应区监听了keyPress事件，为后续的键盘事件做好铺垫。

commit fbb299751f3e6157b6a4b2f357c1d8165a30edb7
```

第 9 章 UI 的个性化键盘控制

9.1 分析和设计

在 Web 开发中，UI 的个性化键盘控制是提升用户体验的核心策略之一。通过监听 keydown 和 keyup 事件，开发者可以实时捕捉到用户的键盘输入，进而设计出更为自然、顺畅的交互流程。event. preventDefault () 方法在此过程中扮演着至关重要的角色，它允许我们阻止浏览器对特定键盘事件的默认处理，从而实现自定义的交互逻辑。

这两个底层键盘事件，一个在按键按下时触发，另一个在按键释放时触发，

为开发者提供了洞察用户输入行为的窗口。在图 9-3 所示的示例中，我们通过在<body>元素上添加事件监听器，确保无论用户焦点在何处，都能捕捉到键盘输入。event. preventDefault()的使用，防止了浏览器对特定键盘事件的默认响应，如页面滚动或表单的自动提交，使得我们可以根据需求定制交互体验。

如图 9-3。展示了用户输入的按键实时显示在“键盘响应区”，keydown 事件记录并输出按下键的值和键码，而 keyup 事件则在图 9-4 的 keyStatus 区域显示键已弹起的信息。此外，我们还会判断所抬起的键是否是字母或特殊符号，如果是，则在 typeText 区域进行显示，如果不是，表示是功能键，我们选择不进行打印。

对于移动端设备，我们需要考虑到屏幕尺寸的影响。在窄屏模式下（如图 9-6 所示），屏幕宽度小于 1000 像素时，由于空间限制，键盘响应区可能不会显示，此时我们需要考虑其他交互方式，如触摸事件。而在宽屏设备上（如图 9-7 所示），由于有足够的空间，键盘响应区会正常显示，用户可以享受到与桌面端类似的键盘交互体验。

然而，优秀的设计不仅在于技术的应用，更在于对用户需求和习惯的理解。在实现自定义键盘控制时，我们必须确保其既高效又直观，避免给用户带来困扰。例如，虽然可以阻止 Enter 键的默认提交功能，但大多数情况下，用户期待 Enter 键在表单中起到提交的作用，因此我们需要在适当的情况下谨慎使用 event. preventDefault()。通过不断实践和优化，我们可以根据不同的用户群体和场景调整键盘控制的实现。无论是桌面端的复杂交互，如游戏控制、表单验证，还是移动端的简洁操作，keydown 和 keyup 事件结合 event. preventDefault()都能提供强大的支持。这些底层事件为我们打开了新的设计思路，让我们能够创造出前所未有的用户体验。

总之，keydown 和 keyup 事件结合 event. preventDefault()为 Web 开发者提供了丰富的工具，让我们能够在交互设计中实现更多可能性。通过深思熟虑的设计和不断迭代，我们可以不断拓展 Web 应用的交互边界，为用户提供更加自然、流畅的使用体验，从而推动 Web 开发领域的创新和发展。页面 URL 如图 9-8 所示。

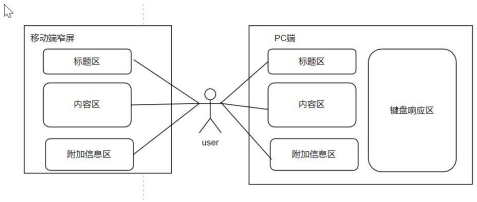


图 9-1 用例图

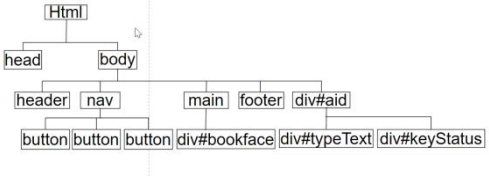


图 9-2 Dom 树

```
$("body").addEventListener("keydown", function(ev) {
    ev.preventDefault();
    let key = ev.key;
    $("keyStatus").textContent = '按下键: ' + key;
});
$("body").addEventListener("keyup", function(ev) {
    ev.preventDefault();
    let key = ev.key;
    $("keyStatus").textContent = key +  "弹起";
    $("typeText").textContent += key;
});
```

图 9-3 keydown 和 keyup 事件

图 9-4 ev.preventDefault()方法结果

```
<div id="aid">
    用户键盘响应区
    <div id="typeText"></div>
    <div id="keyStatus"></div>
</div>
```

图 9-5 键盘响应区

9.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
        现代电影赏析
    </header>
    <nav>
        <button>导航一</button>
        <button>导航二</button>
        <button>导航三</button>
    </nav>
    <main id="main">
        <div id="bookface">
            书的封面
        </div>
    </main>

    <footer>
        万一苇 江西科技师范大学 2022--2025
    </footer>
    <div id="aid">
        用户键盘响应区
        <div id="typeText"></div>
        <div id="keyStatus"></div>
    </div>
```

二、CSS 代码编写如下：

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

        body {
            position: relative;
        }
        header {
            height: 15%;
            border: 2px solid blue;
            font-size: 1.6em;
        }
        main {
            height: 70%;
            border: 2px solid blue;
```

```
        font-size: 1.2em;
        /* background-image: url('../lesson/CS.jpg'); */
        background-size: cover;
        position: relative;
    }
    #bookface {
        width: 80%;
        height: 80%;
        border: 1px solid red;
        background-color: blanchedalmond;
        position: absolute;
        left: 8%;
        top: 8%;
    }
    nav {
        border: 2px solid blue;
        height: 10%;
        font-size: 1.1em;
    }
    footer {
        min-height: 5%;
        border: 2px solid blue;
    }
    #aid {
        position: absolute;
        left: 600px;
        top: 0;
        border: 3px solid blue;
    }
    #typeText {
        color: blue;
        word-break: break-all;
        border: 1px solid blue;
        height: 10%;
        width: 95%;
    }
    #keyStatus {
        position: absolute;
        bottom: 0;
        border: 1px solid blue;
        width: 90%;
        height: 10%;
    }
```

三、JS 代码编写如下：

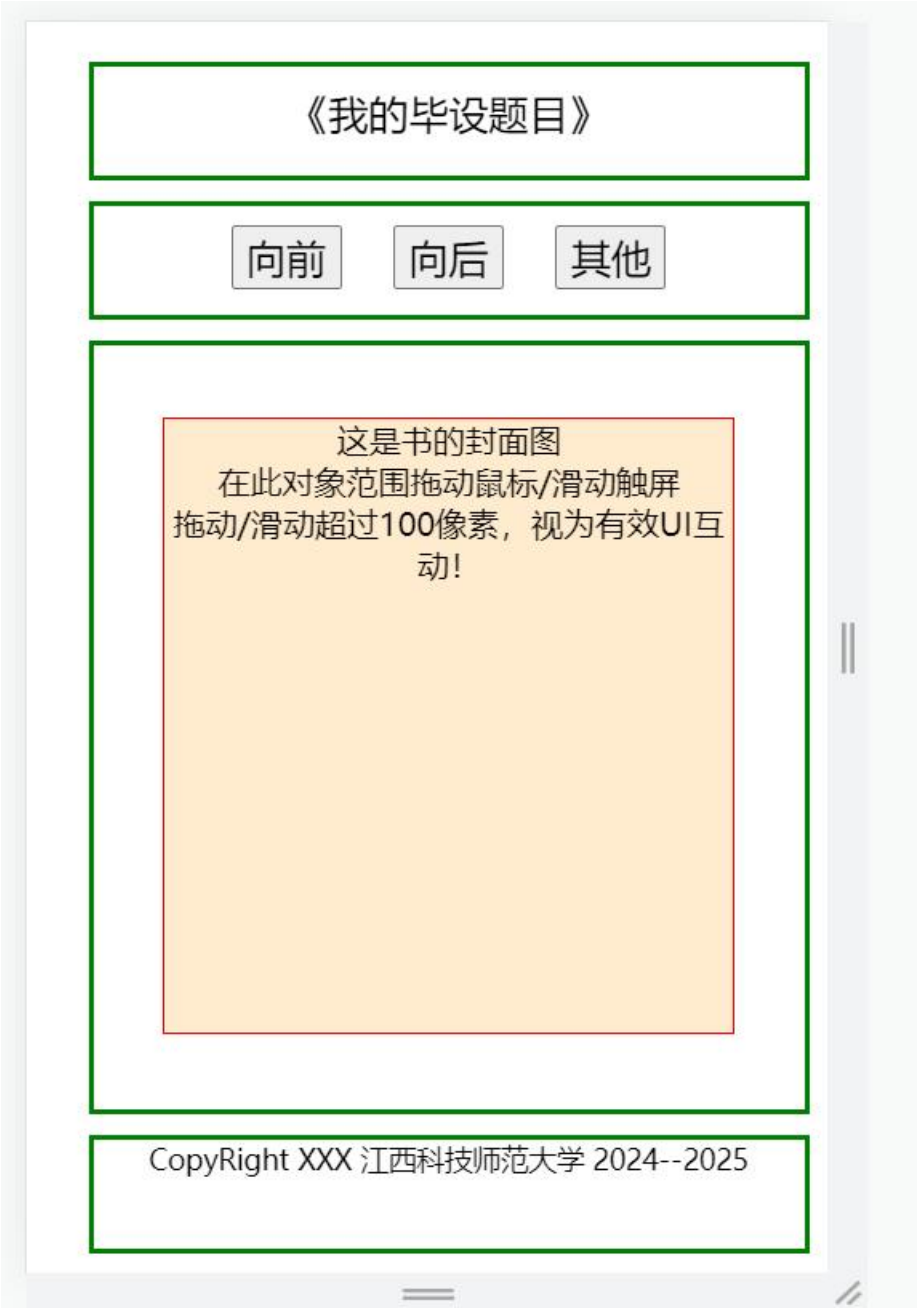
```
var UI = {};
        // 设置宽度  如果宽度大于 600，就只显示 600 宽度
        UI.appWidth = window.innerWidth >= 600 ? 600 :
window.innerWidth;
        // 设置高度
        UI.appHeight = window.innerHeight;
        // 基础字体的大小
        let baseFont = parseInt(UI.appWidth / 20);

        // 设置宽高
        setBodyStyle(baseFont, UI.appHeight, UI.appWidth);
        setAidStyle(UI.appWidth, UI.appHeight);
        //尝试对鼠标和触屏同时进行设计 UI 控制
        var Pointer = {};
        Pointer.isDown = false;
        Pointer.x = 0;
        Pointer.deltaX = 0;
        { //Code Block begin
            const ORIGINAL_LEFT = 8 + '%';
            const VALID_DRAG_DISTANCE = 100;
            let handleBegin = function (ev) {
                Pointer.isDown = true;
                // 记录下起始位置
                Pointer.x = ev.pageX || ev.touches[0].pageX
                Pointer.y = ev.pageY || ev.touches[0].pageY
                if (ev.touches) {
                    console.log("Touch begin : " + "(" + Pointer.x
+ "," + Pointer.y + ")");
                    $("bookface").textContent = "触屏事件开始，坐标:
" + "(" + Pointer.x + "," + Pointer.y + ")";
                } else {
                    console.log("MouseDown 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) >
VALID_DRAG_DISTANCE) {
                    $("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 = ORIGINAL_LEFT;
                }
            } else {
                $("bookface").textContent = "鼠标松开!";
                if (Math.abs(Pointer.deltaX) >
VALID_DRAG_DISTANCE) {
                    $("bookface").textContent += ", 这是有效拖
动! ";
                } else {
                    $("bookface").textContent += " 本次算无效拖
动! ";
                    $("bookface").style.left = ORIGINAL_LEFT;
                }
            }
        };
        let handleMoving = function (ev) {
            ev.preventDefault();
            if (Pointer.isDown) {
                if (ev.touches) {
                    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 {
                    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('本次触发了点击事件：click。');
        });
       /*  $("body").addEventListener("keypress", function
(ev) {
            let key = ev.key;
            $("outputText").textContent += key;
        }); */
        // keydown 和 keyup 事件增加 ev.preventDefault()以后，键
盘 keypress 事件被阻止
        $("body").addEventListener("keydown", function(ev) {
            ev.preventDefault();
            let key = ev.key;
            $("keyStatus").textContent = '按下键：' + key;
        });
        $("body").addEventListener("keyup", function(ev) {
            ev.preventDefault();
            let key = ev.key;
            $("keyStatus").textContent = key +  "弹起";
            $("typeText").textContent += key;
        });
    } //Code Block  end
```

9.3 项目的运行和测试

项目的运行和测试至少要通过两类终端，本文此处给出移动端和 PC 端用 Chrome 浏览器打开项目的结果，如下图 9-6 和图 9-7 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 8-6 的二维码，运行测试本项目的第四次开发的阶段性效果。



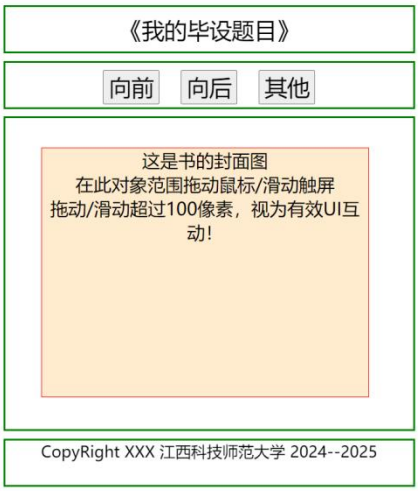


图 9-6 移动端窄屏情况

图 9-7 移动端宽屏情况

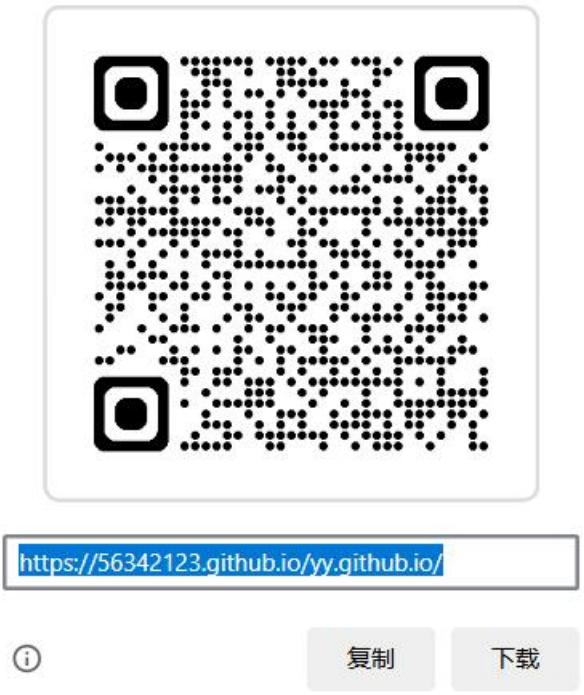


图 9-8 页面 URL

五、JS 代码编写如下：

```
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 $
```

9.4 项目的代码提交和版本管理

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){ //学生须研究这个逻辑的作用
        return false ;
    }
    let puncs =
['~','`','!','@','#','$','%','^','&','*','(',')','-','_','+','=',',','.
',';',';','<','>','?','/',' ','\'','\"'] ;
    if ( (k >= 'a' && k <= 'z') || (k >= 'A' && k <= 'Z')
|| (k >= '0' && k <= '9'))  {
        console.log("letters") ;
        return true ;
    }
    for (let p of puncs ){
     if (p === k) {
          console.log("puncs") ;
         return true ;
        }
    }
    return false ;
     //提出更高阶的问题，如何处理连续空格和制表键 tab?
   }   //function printLetter(k)
});
```

编写好 1.6.html 的代码，测试运行成功后，执行下面命令提交代码：

```
$ git add . #添加所有修改过的文件
$ git commit -m '为键盘响应区添加了 keyup 和 keydown 的监听事件，探讨了
preventDefault()方法的妙用。'
```

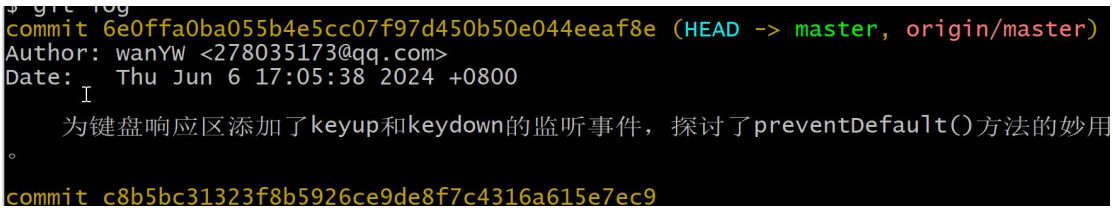
成功提交代码后，gitbash 的反馈如下所示：

```
123@LAPTOP-UTJC666N MINGW64 /homework (master)
$ git commit -m '为键盘响应区添加了keyup和keydown的监听事件，探讨了preventDefaul
t()方法的妙用。'
[master 6e0ffa0] 为键盘响应区添加了keyup和keydown的监听事件，探讨了preventDefaul
t()方法的妙用。
 1 file changed, 267 insertions(+)
 create mode 100644 1.6.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：



第 10 章 本项目中的高质量代码

10.1 分析和设计

编写高质量代码是一个项目的整体过程。这不仅仅是完成任务；它是一门艺术，也是一门科学。目标是实现清晰、高效、易于维护和可扩展性的项目。这个过程涉及很多方面，包括但不限于正确的函数设计、仔细的代码结构组织、合适的模型架构以及对全局变量的使用的严格控制。函数作为代码的基本单元，其设计的好坏直接关系到代码的可读性和可维护性。理想情况下，函数应该像一台复杂的机器，专注于一项任务，而不是冗余或过于复杂。在名称上，函数的名称应该直观地体现其功能，使人们一目了然。保持参数列表简短，以避免与太多输入混淆。返回值必须明确定义，以便调用者知道会发生什么。另外，通过合理的划分函数，可以有效降低代码耦合度，提高模块之间的独立性。代码块的组织也会影响代码的可读性。通过适当的缩进、空行和清晰的逻辑部分，可以使得代码将变得流畅且易于其他人阅读和理解。模块化设计是组织代码块的高级形式。通过将相关功能组合到独立的模块或.js 文件中，这可以大大提高代码结构的清晰度和可重用性。此外，在 JavaScript 开发中，良好的异常处理和带有注释的详细文档是确保代码质量的关键，异常处理通过使用 try...catch 结构来捕获可能的运行时错误，避免程序突然中断，保证了代码的稳定性。例如，在尝试访问可能不存在的对象属性或调用可能抛出错误的方法时，可以将这些操作放入 try 块中，然后在 catch 块中处理异常，提供合适的反馈或恢复机制。同时，详细的文档注释能为代码提供清晰的上下文。它们描述函数的用途、输入参数、返回值，甚至示例用法，使代码更加可靠和可维护，帮助其他开发者快速理解代码功能。模型设计，特别是在面向对象编程中，是软件架构的基石。可以通过遵循单一职责和开放/封闭等设计模式和原则来创建灵活且可扩展的系统。这意味着每个模块必须

有明确的职责，并且设计必须允许在不更改源代码的情况下进行扩展。使用全局变量时必须小心。全局变量由于其高可见性和潜在的副作用而常常成为代码维护的问题。在局部作用域中使用变量、通过函数参数传递数据或使用类封装来管理状态可以帮助减少全局依赖性，如图 10-1 所示，使用代码块的方式来降低全局变量的使用，并使代码更具可预测性和可测试性。这促进了模块化，使每个部分更加独立并减少了错误传播的可能性。综上所述，编写高质量代码是一项综合性工程，需要开发人员完善功能设计、代码组织、模型架构、变量管理等诸多参数。通过遵循上述原则，我们可以创建强大而优雅的代码，为我们软件项目的长期成功奠定坚实的基础。

```
{ //Code Block begin
      const ORIGINAL_LEFT = 8 + '%';
      const VALID_DRAG_DISTANCE = 100;


      let handleBegin = function (ev) {
        Pointer.isDown = true;
        // 记录下起始位置
        Pointer.x = ev.pageX || ev.touches[0].pageX
        Pointer.y = ev.pageY || ev.touches[0].pageY
        if (ev.touches) {
          console.log("Touch begin : " + "(" + Pointer.x + "," +
Pointer.y + ")");
          $("bookface").textContent = "触屏事件开始，坐标：" + "("
+ Pointer.x + "," + Pointer.y + ")";
        } else {
          console.log("MouseDown 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) > VALID_DRAG_DISTANCE) {
            $("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 = ORIGINAL_LEFT;
          }
        } else {
          $("bookface").textContent = "鼠标松开!";
          if (Math.abs(Pointer.deltaX) > VALID_DRAG_DISTANCE) {
            $("bookface").textContent += ", 这是有效拖动! ";
          } else {
            $("bookface").textContent += " 本次算无效拖动! ";
            $("bookface").style.left = ORIGINAL_LEFT;
          }
        }
      };
      let handleMoving = function (ev) {
        ev.preventDefault();
        if (Pointer.isDown) {
          if (ev.touches) {
            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 {
            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('本次触发了点击事件：click。');
      });
     /* $("body").addEventListener("keypress", function (ev) {
        let key = ev.key;
        $("outputText").textContent += key;
      }); */
      // keydown 和 keyup 事件增加 ev.preventDefault()以后，键盘
keypress 事件被阻止
      $("body").addEventListener("keydown", function(ev) {
        ev.preventDefault();
        let key = ev.key;
        $("keyStatus").textContent = '按下键：' + key;
      });
      $("body").addEventListener("keyup", function(ev) {
        ev.preventDefault();
        let key = ev.key;
        $("keyStatus").textContent = key + "弹起";
        $("typeText").textContent += key;
      });
    } //Code Block end
```

图 10-1 代码块

10.2 项目的实现和编程

一、JS 代码编写如下：

```
    /**
     * 给 body 设置样式
     * @param { number } baseFont 基础字体
     * @param { number } appHeight 设备高度
     * @param { number } appWidth 设备宽度
     */

function setBodyStyle(baseFont, appHeight, appWidth) {
          // 通过修改 body 对象的字体大小，可以遗传给子代，从而实现
自己的响应式设计
          document.body.style.fontSize = baseFont + 'px';
          // 把 body 的高度设置为屏幕的高度，实现了纵向全屏
          // 通过 CSS 对 body 子对象的高度百分比进行分配，从而达到响
应式设计的目标。
          document.body.style.height = appHeight - 70 + 'px';
          document.body.style.width = appWidth + 'px';
```

```
        }

        /**
         * 给 aid 设置样式
         * @param { number } appHeight 设备高度
         * @param { number } appWidth 设备宽度
         */
        function setAidStyle(appWidth, appHeight) {
            let aid = $('aid');
            if (window.innerWidth <= 1000) {
                aid.style.display = 'none';
            }
            aid.style.width = window.innerWidth - appWidth - 30 +
'px';
            aid.style.height = appHeight - 62 + 'px';
        }
```

10.3 项目的代码提交和版本管理

第 11 章 用 gitBash 工具管理本项目的 http 服务器

11.1 经典 Bash 工具介绍

在当今现代软件开发的时代，良好的代码版本管理是保持一个项目整洁和团队协作的关键。Git 是一个开源的分布式版本控制系统，是目前世界上最先进、最流行的版本控制系统，可以快速高效地处理从很小到非常大的项目版本管理。其特点为：项目越大越复杂，协同开发者越多，越能体现出 Git 的高性能和高可用性[10]。Git 的分布式的特点意味着每个开发者在本地机器上都可以拥有一个完整的项目副本，可以对这个副本进行独立的开发和测试，不需要依赖中央服务器。这样的方式允许开发者可以自由地创建分支，实验新的功能或修复 bug，然后通过合并请求将更改合并返回主分支。这种灵活性允许团队成员并行工作，提高了开发效率，减少了等待时间。

在项目的根目录中打开 git bash 通过执行 git init 命令创建 git 仓库，这个简单的过程就构建起了一个强大的代码版本管理控制体系，让项目中的每一个文件

创建、每一次文件改动，都会被 git 追踪和管理，随着代码的开发进行，html、css 和 js 文件都会被频繁修改，为了将这些变化都被 git 记录，使用 git add . 命令添加所有改动的文件到 git 暂存区，在暂存区中所有的改动都将等待被纳入到正式提交，实现对代码变更的精细控制。并且当添加到暂存区后，可以使用 git status 命令来查看当前 git 仓库的状态，Git 会告知当前所处的分支名称，这对于多分支开发环境非常重要，帮助确认自己的工作环境，Git 还会区分并报告哪些文件已被修改但尚未暂存，哪些文件已被暂存等待提交，以及哪些全新的文件还未被 Git 跟踪如图 11-1 所示。

提交是 Git 的核心操作，通过 git commit -m 命令，开发者可以将暂存区的改动永久保存，正式的将代码保存到本地仓库中，为之后提交的远程 GitHub 仓库做好铺垫。提交信息应当简洁明了，描述本次提交的主要改动。这些提交信息为日后的代码审查、故障排查提供了宝贵的信息。当开发者提交完代码后，可以使用 git log 命令来将查看详细的历史记录，这个命令会列出所有之前的提交，每个提交都有其唯一的哈希值，作者信息，提交时间，以及在提交时所写的描述信息。这些日志可以帮助你追踪代码的变化，理解项目的发展历程，以及回溯到特定的代码状态，搭配 git reset 命令可以帮助开发者快速的回到特点的代码状态。

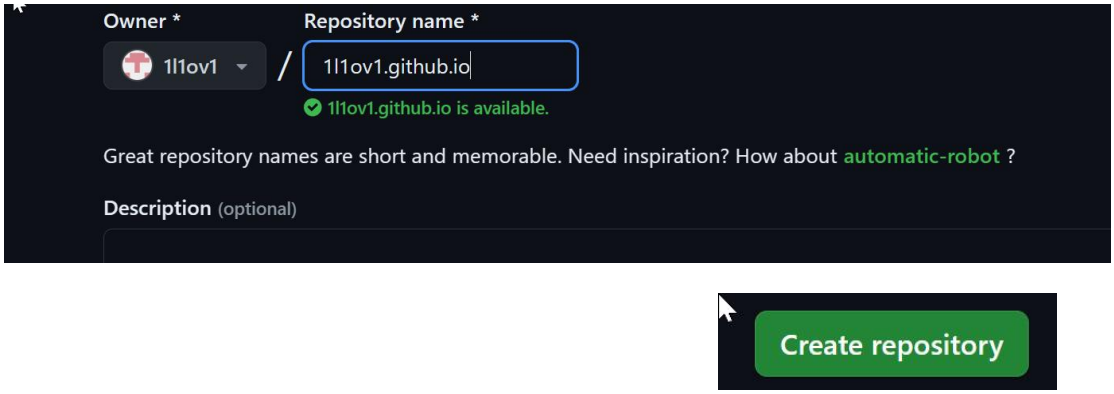
远程仓库是 Git 协作的另一个核心元素。通过 git remote add origin <repository_url>，开发者可以将本地仓库与 GitHub 远程仓库关联，使用 git push 和 git pull 进行代码的同步，实现团队间的共享和协作。总而言之，Git 不仅是一个版本控制系统，更是一个促进团队协作、提升开发效率的平台。通过 git add 和 git commit -m 等核心命令，以及其丰富的分支管理、远程协作和冲突解决功能，Git 在 HTML、CSS 和 JavaScript 等前端开发中发挥着无可替代的作用。开发者通过熟练掌握 Git，可以更好地驾驭代码的演化，确保项目的稳定性和团队的协同工作，进而创造出更高品质的软件产品。而将代码公开上传到 Git Hub 上这一行为是对开源精神的践行，它不仅仅是技术分享，更是一种促进全球协作与创新的文化体现。

```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory
        modified:   example/1.5.html
        modified:   example/index.html
        modified:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        example/1.6.html
        lesson/CSS.jpg
        lesson/CT.jpg
        lesson/GRE.jpg
        lesson/Git.jpg
        lesson/NinjaJS.jpg
```

图 11-1 git status 结果

11.2 通过 gitHub 平台实现本项目的全球域名

11.3 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。

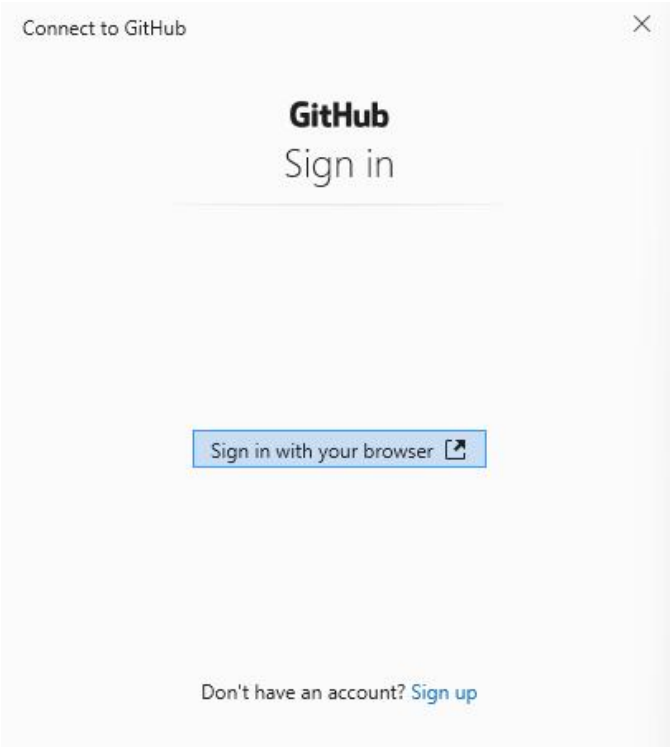
11.4 设置本地仓库和远程代码仓库的链接

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

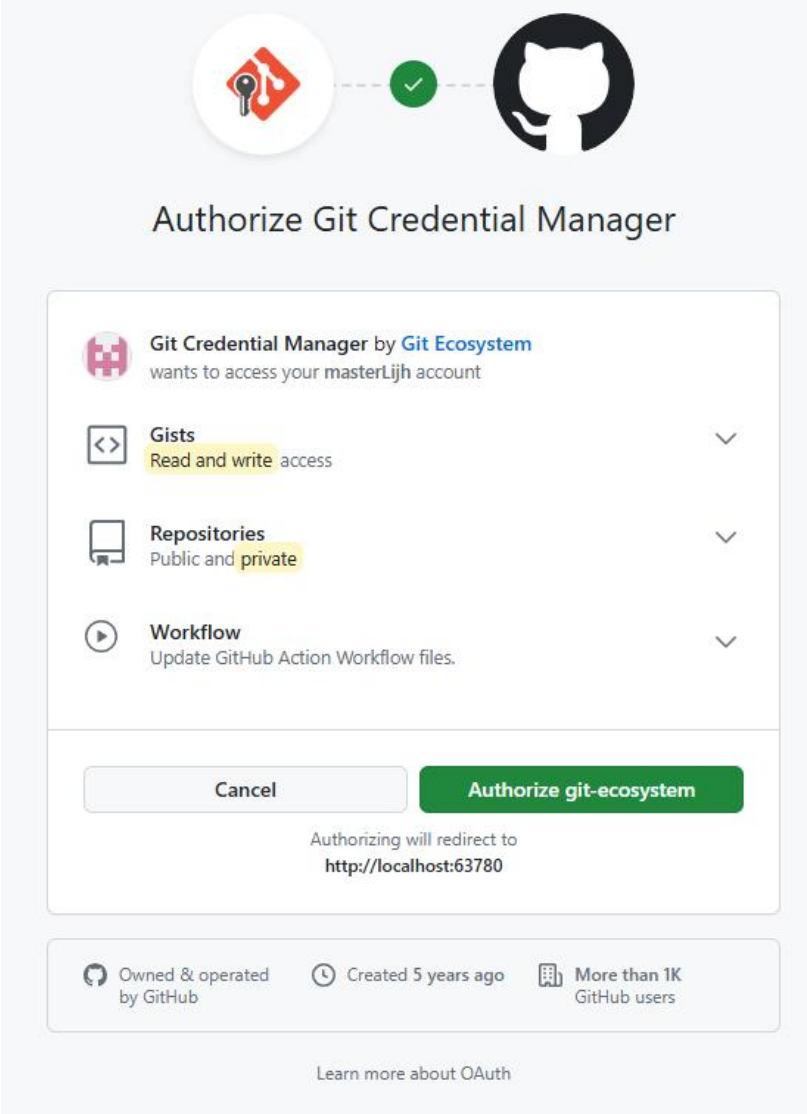
```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md  #添加了一个文件名叫做 README，内容为 WebUI 应用的远程 http 服务器设置的 markdown 文件
$ git init #初始化仓库
$ git add README.md # 将文件添加进仓库的暂存区中
```

```
$ git commit -m "这是我第一次把代码仓库上传至 gitHub 平台" #提交暂存区中的内容
$ git branch -M master# 创建分支 master
$ git remote add origin https://github.com/1l1ov1/1l1ov1.github.io.git #添加远程仓库
$ git push -u origin master#将本地仓库中的文件提交到远程仓库中
```

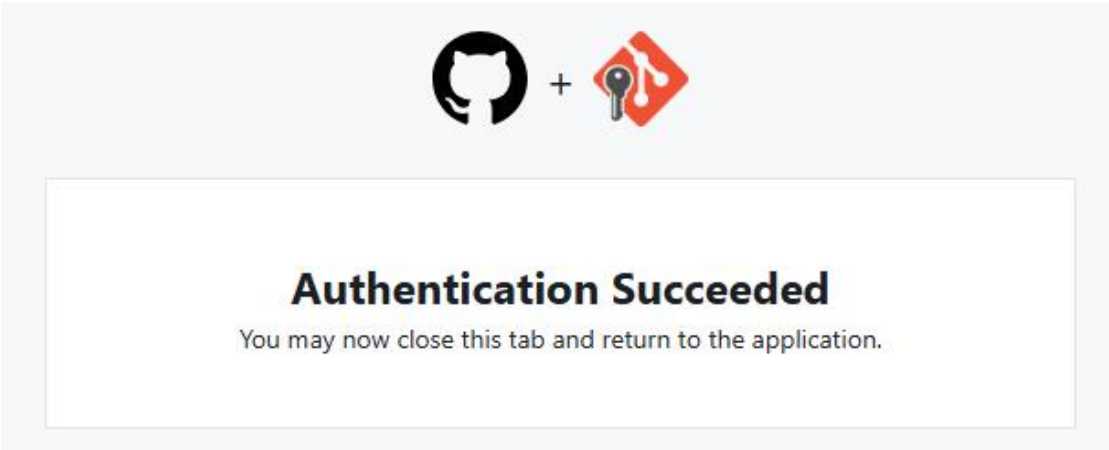
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



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：

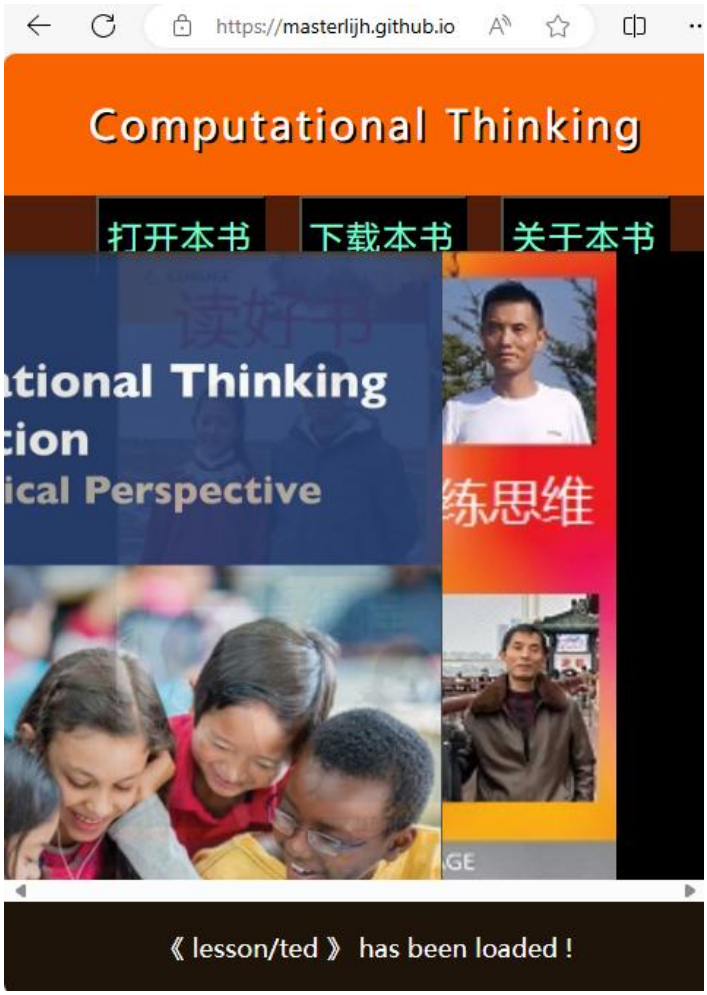


最后，GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。



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：



全文完成，谢谢！

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

[8]. 王政.Web 前端开发技术以及优化研究[J].电脑知识与技术,2013,9(22):5037-5038.
[9]. 彭显雯. 基于 MVVM 模式的响应式轻量级前端组件设计与实现[D].华中科技大学,2019.
[10]. 仇礼钦,王鑫,盛飞龙,等.基于 Git 的软件项目管理配置方法及应用实践[J].机电工程技术,2023,52(05):223-227.

附件：论文的三段论写作心法、摘要案例 1、摘要案例 2

写作心法：实质上任何章节都可以按三段论模式写，第一段是写研究的背景和目标，第二段是写你所开展的工作步骤和工作量，第三段是用了哪些方法和工作的结果或意义。

【摘要案例 1】近十年来，html5 为核心的 web 标准的软件开发技术以其跨平台、开源的优势广泛地运用在各个领域的应用软件开发中。通过分析本次毕设任务，本项目选择 html5 的 web 客户端技术为技术路线，展开对程序设计和软件开发的研究和实践。通过广泛查阅相关技术书籍、开发者论坛和文献，设计开发了一个个性化的用户界面（UI）的应用程序。在开发中综合应用了 html 语言进行内容建模、css 语言展开 UI 的外观设计、javascript 语言编程实现 UI 的交互功能，除直接使用了 web 客户端最底层的 API 外，本项目的每条代码都是手工逐条编写，没有导入他人的任何的代码（框架和库）。本项目也采用了响应式设计编程，可以智能地适应移动互联网时代用户屏幕多样化的需要；另外大量地运用了面向对象的程序设计思想，比如用代码构建了一个通用的 pointer 模型，该模型仅用一套代码就实现了对鼠标和触屏的控制，实现了高质量的代码，这也是本项目的亮点。从工程管理的角度看，本项目采用的增量式开发模式，以逐步求精的方式展开了六次代码的增量式重构（A:Analysis，D:Design，I: Implementation，T:Testing），比较愉快地实现项目的设计开发和测试。从代码的开源和分享的角度看，本项目采用了 git 工具进行版本管理，在漫长

的开发过程中重构代码六次并正式做了代码提交，另外在测试中修改提交了代码两次，最后利用gitbash工具 把本项目的代码仓库上传到著名的github上,再利用github提供的http服务器，本项目实现了UI应用在全球互联网的部署，我们可以通过地址和二维码便捷地跨平台高效访问这个程序。

【摘要案例2】： Web技术以其跨操作系统平台的优势成为了广泛流行的软件开发手段，为了适应移动互联网时代软件项目的前端需求，本项目以Web客户端技术为研究学习内容，广泛查阅了技术资料与相关文献，尤其是mozilla组织的MDN社区的技术实践文章，探索了HTML内容建模、CSS样式设计和JavaScript功能编程的基本技术和技巧。通过集成上述技术，再应用本科的相关课程的知识，实现了一个个性化的用户界面（UI: uer interface）的项目，该用户界面以响应式技术为支撑做到了最佳适配用户屏幕,程序可以动态适用于当前PC端和移动端设备；在功能上以DOM技术和事件驱动模式的程序为支撑实现了对鼠标、触屏、键盘的底层事件响应和流畅支持，为鼠标和触屏设计了一个对象模型，用代码实现了对这类指向性设备的模拟（这是本项目模型研究法的一次创新实践，也是本项目的亮点。）。为了处理好设计和开发的关系,项目用了工程思想管理,使用了软件工程的增量式开发模式,共做了6次项目迭代开发,每次迭代都经历了开发4个经典开发阶段(A:Analysis,D:Design,I: Implementation, T:Testing），以逐步求精的方式编写了本UI的应用程序。为了分享和共享本代码，与网上的开发者共同合作，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12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过程，最后通过gitbash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。