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：朱红燕

学　　号：20213648

指导教师：李健宏

2024 年 6 月 10 日

# 目录


1. 前言 ..... 2
1.1. 毕设任务分析 ..... 2
1.1.1. 毕业设计的目的和意义 ..... 2
1.1.2. 毕业设计/论文规划路线 ..... 2
1.2. 研学计划 ..... 2
1.3. 研究方法 ..... 3
1.3.1. 毕业设计开发 ..... 3
1.3.2. 毕业论文撰写 ..... 4
2. 技术总结和文献综述 ..... 4
2.1. Web 平台和客户端技术概述 ..... 4
2.2. 软件开发的过程管理 ..... 5
2.2.1. 瀑布模型 ..... 5
2.2.2. 增量模型 ..... 6
3. 内容设计概要 ..... 7
3.1. 项目分析与设计 ..... 7
3.2. 项目编程与实现 ..... 7
3.3. 项目测试与运行 ..... 10
3.4. 项目代码提交与版本管理 ..... 11
4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计 ..... 12
4.1. 分析与设计 ..... 12
4.2. 编程与实现 ..... 13
4.3. 测试与运行 ..... 15
5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI ..... 16
5.1. 分析与设计 ..... 16
5.2. 编程与实现 ..... 17
5.3. 测试与运行 ..... 20
6. 个性化 UI 设计中对鼠标交互的设计开发 ..... 22
6.1. 分析与设计 ..... 22

6.2. 编程与实现 ..... 23
6.3. 测试与运行 ..... 24
7. 对触屏和鼠标的通用交互操作的设计开发 ..... 25
7.1. 分析与设计 ..... 25
7.2. 编程与实现 ..... 26
7.3. 测试与运行 ..... 29
8. UI 的个性化键盘交互控制的设计开发 ..... 30
8.1. 分析与设计 ..... 30
8.2. 编程与实现 ..... 30
8.3. 测试与运行 ..... 32
9. 用 git 工具管理代码的版本和项目发布 ..... 34
9.1. Git 介绍 ..... 34
9.2. 通过 GitHub 平台实现本项目的全球域名 ..... 34
9.3. 设置本地仓库和远程代码仓库的连接 ..... 35
参考文献 ..... 38

# 基于 Web 客户端技术的个性化 UI 设计和实现

**摘要：**近年来，HTML5 作为 Web 开发的核心技术，因跨平台与开源优势，在多领域软件开发中占据重要位置。本毕业设计深究任务需求，选择了 HTML5 Web 客户端技术，以深化软件开发与程序设计实践。初期，通过广泛研究技术文献与社区资源，我们着手开发个性化 UI 应用。开发中，我们利用 HTML 构建内容模型，CSS 美化界面，并以 JavaScript 实现界面交互，直接操纵底层 API，坚持手工编码，无外部框架依赖，展现了创新与定制化水平。应用还采纳响应式设计，适配不同移动设备屏幕，优化用户体验。程序设计上，采用面向对象策略，创新实现“指针”模型，统一操控鼠标和触摸输入，显著提升代码复用与维护效率。项目管理上，采取增量式开发，经历六次迭代（A-D-I-T 循环），细化项目，保障开发测试顺利。为促进代码共享，项目使用 Git 进行版本管理，历经六轮重构与周期内代码提交，测试期间两次修正。最终，借助 Git Bash，代码仓库部署到 GitHub，并利用其 HTTP 服务，实现了全球部署，用户可便捷跨平台访问，体现了项目的实用性和技术开放度。

# 1. 前言

## 1.1.毕设任务分析

### 1.1.1. 毕业设计的目的和意义

从学校的培养计划的角度看，毕业设计是完成专业教学计划、达到应用型本科专业培养目标的重要环节，也是教学设计规划中综合性最强的教学实践环节，它对培养我们独立分析问题和解决问题的能力、提高学生全面素质具有重要的意义。 毕业论文则体现的是学生对完成毕业设计整个过程的记录和总结，也包含毕业设计的理论支撑。对于我们计算机科学与技术专业的学生来说，毕业设计和论文是有机统一的，其包含多重目标:

(1) 综合运用所学专业的理论和技术，分析和解决实际问题的能力;

(2) 掌握文献索引、资料查询的基本方法及获取新知识的能力;

(3) 设计和开发计算机软件或运用系统的基本能力;

(4) 论文写作和语言表达的能力等。

### 1.1.2. 毕业设计/论文规划路线

计算机专业学生的毕业设计的选题大多为某一系统的设计与实现，完成毕业设计的首要前提是具有软件开发的专业核心能力，而此专业能力的培养需要学习和掌握大量的专业理论知识和技术，最主要的途径是学习大学的一系列课程，像“数据结构与算法”、“操作系统”、“计算机网络”、“软件工程”、 “数据库系统课程设计”、“Java 面向对象程序设计”等。单单掌握理论知识也是不可取的，完成系统设计还需学生具有工程的思维，将问题系统化模块化，将知识理论与工程有机结合。

毕业论文撰写阶段，内容方面要求学生规范阐述需求分析、概要设计、详细设计、测试等环节设计逻辑与实现方案，尤为重要的一点是要有创新意识和能力，在前人研究的基础上，能够提出新颖的观点或方法，进行一定程度的创新，即使是对于已有的问题提供新的视角或解决方案也是重要的创新体现。论文规范方面要求学生按照中华人民共和国国家标准 GB7713-87《科学技术报告、学位论文和学术论文的编写格式》标准对毕业论文进行撰写，写作方法及格式规范等均在其中做了详细规定。

## 1.2.研学计划

根据我校本科生毕业论文（设计）过程管理手册中的毕业论文（设计）进度计划来看，毕业论文（设计）的中心工作计划在 2024 年下半年开展，于 2025 年上半年末完成

收尾工作：

| 毕业论文（设计）进度计划 | | | |
|---|---|---|---|
| 序号 | 各阶段工作内容 | 起讫日期 | 备注 |
| 一 | 定选题 | 2024 年 10 月 | 指导教师向学生下达任务书 |
| 二 | 开题答辩 | 2024 年 10 月 | 进行毕业论文开题答辩，制作开题报告答辩 PPT |
| 三 | 文献综述 | 2024 年 10 月 | 完成文献综述的撰写，文献综述要求 3000 字。 |
| 四 | 论文一稿 | 2024 年 11 月 | 撰写论文框架 |
| 五 | 论文二稿 | 2024 年 12 月 | 撰写论文内容 |
| 六 | 中期检查 | 2025 年 1 月 | 项目作品完成，论文完成一半的进度 |
| 七 | 论文三稿 | 2025 年 1 月 | 论文格式按照学术规范 |
| 八 | 论文定稿 | 2025 年 2 月—2025 年 4 月 | 做好中国知网学术不端审核检测自查工作 |
| 九 | 毕业论文（设计）中国知网学术不端审核检测自查工作 | 2025 年 4 月 | 提交中国知网重复率检测报告，踩实本科毕业生毕业论文（设计）中国知网学术不端审核检测自查工作。 |

## 1.3.研究方法

本小节提出方法适用于选题为某一系统的设计与实现的计算机专业学生。

### 1.3.1. 毕业设计开发

要设计和开发出一个高效、稳定且友好的系统，除了需要掌握计算机科学与技术的基础课程，也应熟练掌握至少一种或几种编程语言，如 Java、Python、C++、JavaScript 等，具体取决于项目需求。除此之外，项目管理与软件工程知识也是必不可少的，要理解软件开发生命周期（A:Analysis、D:Design、I:Implementation、T:test）和项目管理的基本原则，能够使用版本控制工具（如 Git）进行团队协作。

另外，要有技术选型能力。根据项目需求，能够合理选择合适的技术栈、框架和开发工具。这要求学生对当前的技术趋势有所了解，并能评估不同技术方案的优劣。

### 1.3.2. 毕业论文撰写

在毕业论文撰写阶段，选择合适的研究方法是确保研究质量与深度的关键。其中模型研究法与文献研究是两种最常用的研究方法。

模型研究法是指通过构建理论模型、数学模型或计算机模拟模型等，对研究对象的内在机制、运行规律或未来趋势进行分析和预测的一种方法。这种方法的优势在于能够直观展现复杂系统的工作原理，帮助研究者在控制变量的条件下深入探究因果关系，验证理论假设。模型研究法要求研究者具备较强的抽象思维能力和数学、计算机技能，以确保模型的准确性和实用性。

文献研究法则是通过对现有文献资料的收集、整理、分析和综合，来探讨某一研究问题或理论框架的方法。它侧重于回顾、总结和批判性评价已有的研究成果，为新的研究提供理论依据、研究思路或指出研究空白。该方法的优势在于能够高效地利用前人的智慧，避免重复劳动，同时通过文献对比分析，促进理论创新和研究视角的多元化。进行文献研究时，关键在于全面、客观地筛选文献，采用科学的分类与编码技术，以及深入的批判性分析，以保证研究的深度和广度。

# 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。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。 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）[1]；

HTML（Hyper Text Markup Language，超文本标记语言）是一种用来告知浏览器如何组织页面的标记语言。HTML 可复杂、可简单，一切取决于 web 开发者。HTML 由一系列的元素组成，这些元素可以用来包围或标记不同部分的内容，使其以某种方式呈

现或者工作。

CSS（层叠样式表）用于设置和布置网页——例如，更改内容的字体、颜色、大小和间距，将其拆分为多个列，或添加动画和其他装饰功能。这个模块为你掌握 CSS 的过程提供了一个温和的开端，包括它如何工作的基础知识，语法是什么样的，以及如何开始使用它来为 HTML 添加样式。

JavaScript 是一种脚本编程语言，它可以在网页上实现复杂的功能，网页展现给你的不再是简单的静态信息，而是实时的内容更新——交互式的地图、2D/3D 动画、滚动播放的视频等等——JavaScript 就在其中[2]。

## 2.2.软件开发的过程管理

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）[3]。

### 2.2.1. 瀑布模型

在瀑布模型中，开发过程仅沿一个方向流动。 这意味着在完成前一个阶段之前，无法启动一个阶段。换句话说，整个项目的分析阶段应该在设计阶段开始之前完成。在实施阶段开始之前，应完成整个设计阶段，如下图所示。 瀑布模型各有优缺点。一个优点是每个阶段都在下一阶段开始之前完成。例如，在设计阶段工作的小组确切地知道该做什么，因为他们拥有分析阶段的完整结果。测试阶段可以测试整个系统，因为正在开发的整个系统都已准备就绪。然而，瀑布模型的一个缺点是难以定位问题：如果部分过程出现问题，则必须检查整个过程。

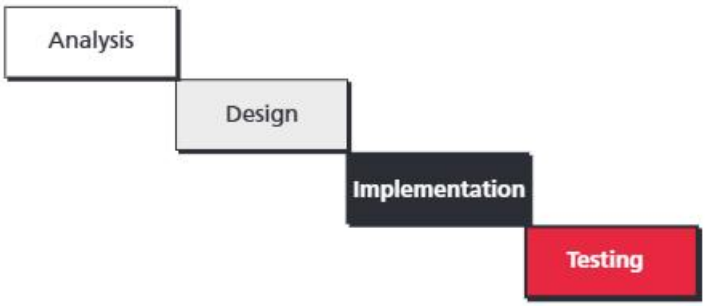


图 1 瀑布模型示例图

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。

### 2.2.2. 增量模型

当今开源的软件开发环境背景下，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。

在增量模型中，软件是分一系列步骤开发的。开发人员首先完成整个系统的简化版本。此版本代表了整个系统，但不包括详细信息，如下图显示的增量模型概念。 在第二个版本中，添加了更多细节，而一些细节尚未完成，并再次测试系统。如果有问题，开发人员知道问题出在新功能上。在现有系统正常工作之前，它们不会添加更多功能。此过程将持续进行，直到添加所有必需的功能。

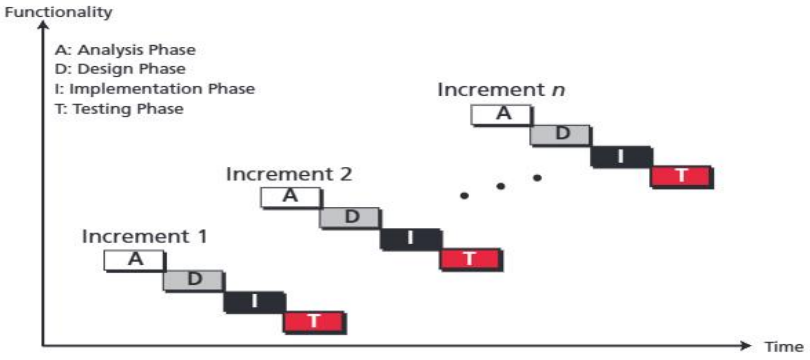


图 2 增量模型示例图

# 3. 内容设计概要

## 3.1.项目分析与设计

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”式简洁方式开展界面设计，首先用一个标题性信息展示 logo 或文字标题，吸引用户的注意力，迅速表达主题；然后展现主要区域，也就是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的重点；最后则是足部的附加信息，用来显示一些用户可能关心的细节变化。如下面用例图所示：

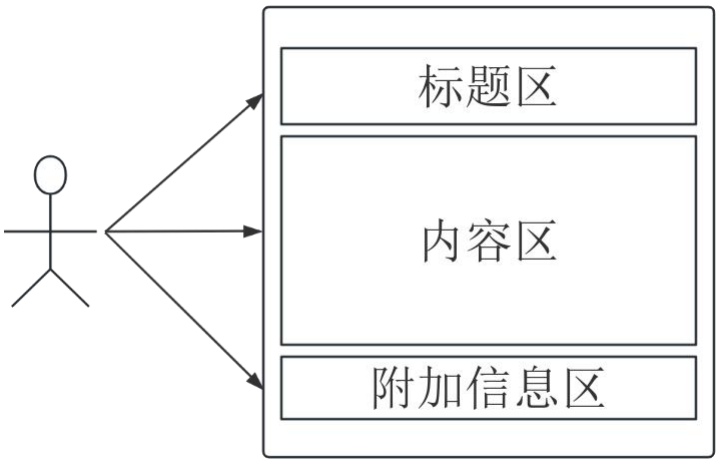


图 3 用户界面用例图

项目 Dom 树结构如下所示：

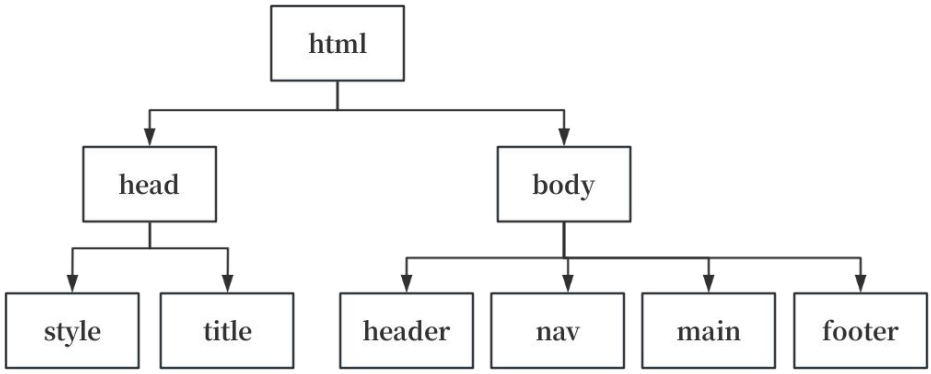


图 4 项目 Dom 树结构

## 3.2.项目编程与实现

项目 HTML 片段：

```
<!doctype html>
```

```
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width , initial-scale=1">
  <title>WebUI1</title>
 </head>
 <body>
  <header>
    <p>
        《计算机思维和语言学习》
    </p>
  </header>
  <nav>
    <div id="buttons">
      <a href="https://zhuhyan.github.io/WebUI1.2.html"><button>1.2</button></a>
      <a href="https://zhuhyan.github.io/WebUI1.3.html"><button>1.3</button></a>
      <a href="https://zhuhyan.github.io/WebUI1.4.html"><button>1.4</button></a>
      <a href="https://zhuhyan.github.io/WebUI1.5.html"><button>1.5</button></a>
      <a href="https://zhuhyan.github.io/WebUI1.6.html"><button>1.6</button></a>
      </div>
    </div>
  </nav>
   <main>
    <p>
        ‘读好书、练思维、勤编程’ 计算思维系列课程
    </p>
   </main>
  <footer>
    <p>
      CopyRight from  朱红燕  江西科技师范大学  2024--2025
```

```
    </p>
  </footer>
 </script>
 </body>
</html>
```

项目 CSS 片段：

```
 <style>
    header{
      border: 2px solid green;
      height: 100px;
    }
    nav{
      border: 2px solid green;
      height: 100px;
    }
    main{
      border: 2px solid green;
      height: 400px;
    }
    footer{
      border: 2px solid green;
      height: 100px;
    }
    body{
      font-size: 30px;
    }
    p{
      text-align: center;
      display: flex;
      justify-content: center;
```

```
        align-items: center;
    }
    #buttons{
        margin: 25px;
        text-align: center;
        display: flex;
        justify-content: center;
        align-items: center;
    }
    button{
        font-size: 45px;
        margin-right: 30px;
        height: 60px;
        width: 80px;
    }
</style>
```

### 3.3.项目测试与运行

用户界面（以 PC 端为例）如下图所示。由于本项目已上传至 github 网站，移动端用户可以扫描下方二维码查看本项目。

图 5 PC 端用户界面

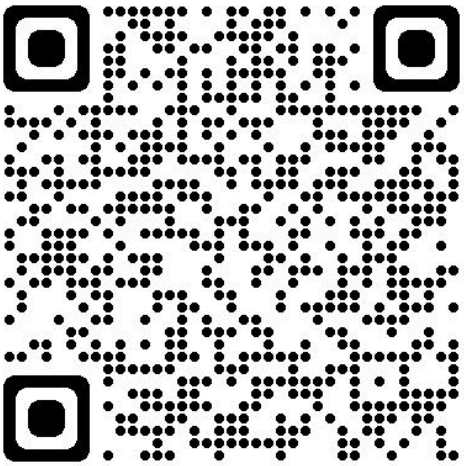

图 6 项目二维码

## 3.4.项目代码提交与版本管理

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

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ mkdir webUI
$ cd webUI
$ git init
$ git config user.name 朱红燕
$ git config user.email 2942503667@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 “建立应用的文件夹结构并做第一次代码提交”
```

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 6bb16b73a3adb9b3a5eb66b319ec4a692ea3e9ac
Author: 朱红燕 <2942503667@qq.com>
Date:   Tue May 28 08:51:20 2024 +0800

    建立应用的文件夹结构并做第一次代码提交
```

# 4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计

## 4.1.分析与设计

计算机使用的显示器硬件差别很大，显示器的尺寸和分辨率取决于成本。设计师没有为每种显示类型设定每个网页的版本，而是选择让网页提供一般布局指南，并允许浏览器选择如何在给定计算机上显示页面。 因此，一个网页不会给出很多细节。如，一个网页的作者可以指定一组句子组成一个段落，但作者不能指定一行的确切长度或是否缩进段落的开头等细节。网页给出了关于所需表现形式的一般准则：浏览器在显示页面时选择细节。因此，相同的网页在两台不同的计算机上或通过不同的浏览器显示时，其显示效果可能略有不同，这就是响应式设计。而如今针对窄屏终端的响应式设计成为了 UI 设计师和前端开发者关注的焦点[4]。窄屏设备，如智能手机和平板电脑，其屏幕尺寸、分辨率及纵横比的多样性远超传统桌面显示器，这要求 UI 设计不仅要保持视觉上的吸引力和信息的清晰传达，还需具备高度的灵活性和自适应性。本项目设计窄屏终端的用户界面用例图和 Dom 树结构如下所示：

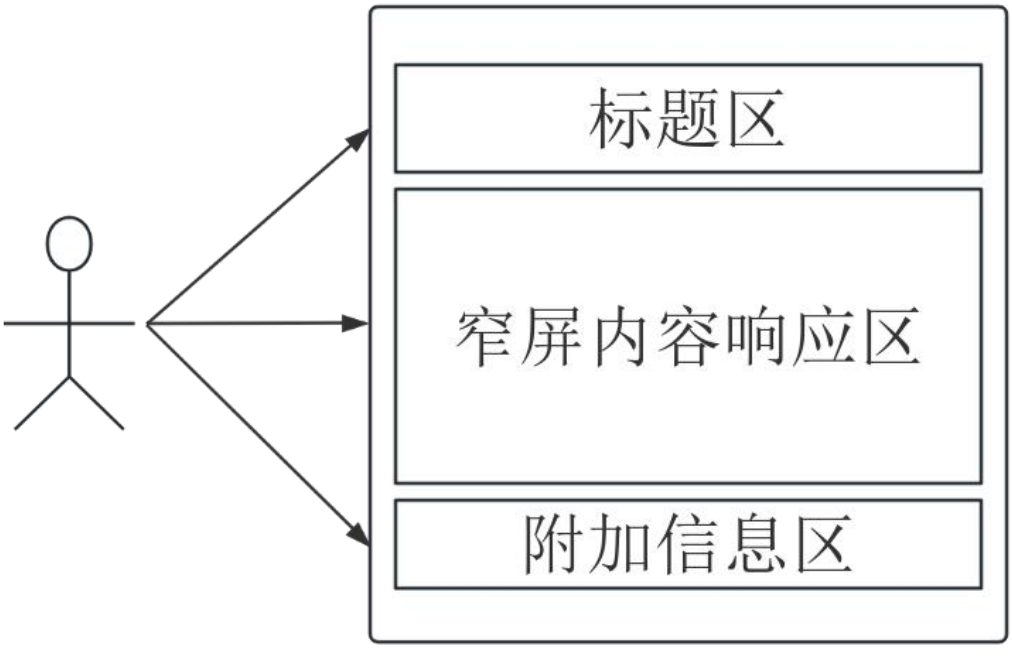

图 7 窄屏用户界面用例图

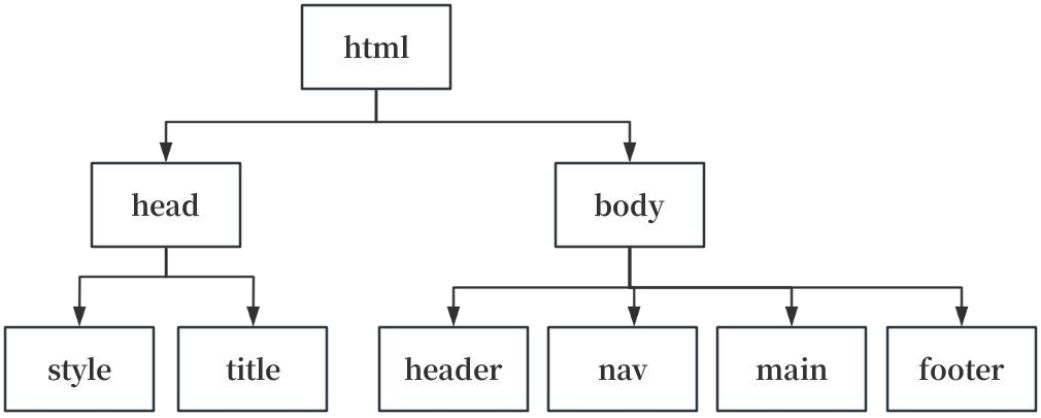


图 8 窄屏项目 Dom 树结构

## 4.2.编程与实现

实现响应式设计的代码逻辑如下：用 JavaScript 来动态读取显示设备的信息，然后按设计，使用 Javascript + CSS 来部署适配当前设备的显示的代码。

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 。

```
*{
    margin: 5px;
    text-align: center;
 }
 header{
    border: 3px solid green;
    height: 10%;
    font-size: 0.7em;

 }
 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:0.5em;
    }
```

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。

```
var UI = {};
  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
document.body.style.width = UI.appWidth  + "px";
document.body.style.height = UI.appHeight - 62 + "px";
```

## 4.3.测试与运行

窄屏用户界面（以 iPhone12 pro 用户为例）如下图所示。由于本项目已上传至 github 网站，移动端用户可以扫描下方二维码查看本项目。

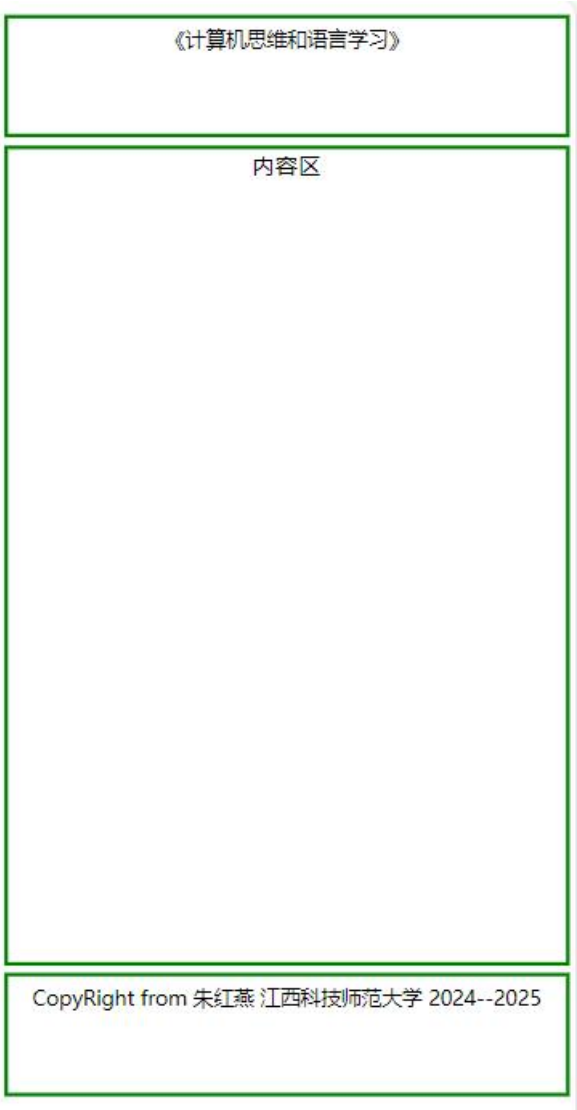


图 9 iPhone12 pro 用户界面

图 10 项目二维码

# 5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI

## 5.1.分析与设计

随着市场上出现越来越多的设备，针对桌面和移动设计不同样式的方法已过时。将桌面和移动设备之间的强制二分法导致的问题多于解决的问题。 更好的方法是为所有用户提供相同的 HTML 和 CSS。通过使用一些关键技术，用户可以根据用户的浏览器视口大小（或偶尔基于屏幕分辨率）以不同的方式呈现内容。这样，开发人员就不需要两个不同的网站。开发人员创建了一个适用于智能手机、平板电脑或用户扔给它的任何其他东西的网站。这种方法由网页设计师 Ethan Marcotte 推广，称为响应式设计[5]。本项目应用响应式设计开发技术开发了一个可适配窄屏和宽屏的 UI，窄屏用户和宽屏用户界面用例图如下所示：

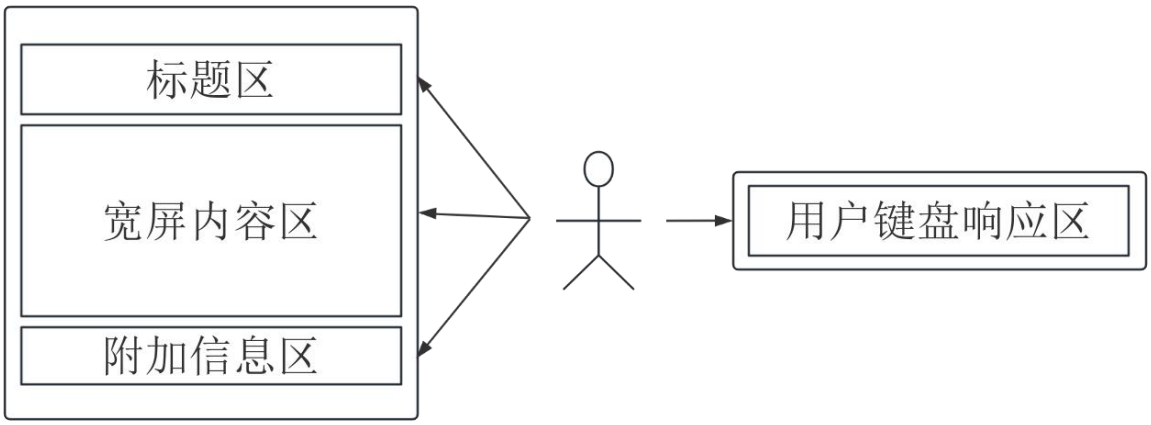


图 11 宽屏用户界面用例图

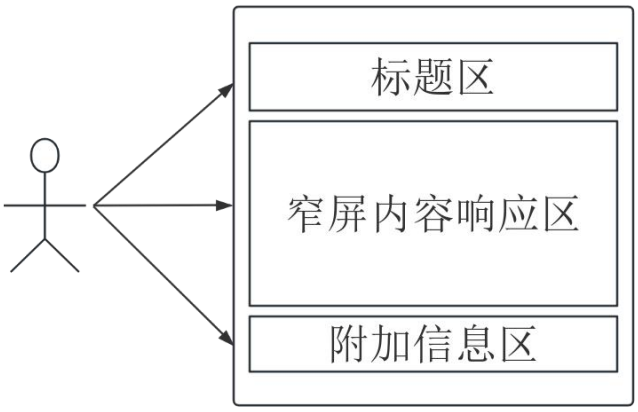


图 12 窄屏用户界面用例图

项目 Dom 树如下所示：

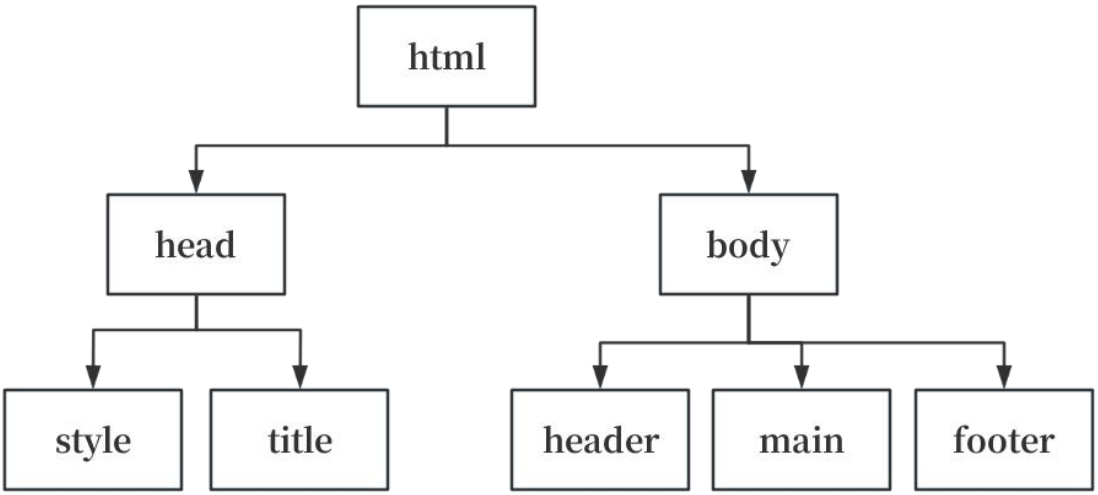


图 13 宽屏窄屏项目通用 Dom 树

## 5.2.编程与实现

本项目代码实现了一套响应式设计逻辑，旨在适应不同宽度的屏幕，特别是宽屏和窄屏，主要通过 JavaScript 来动态调整页面布局和交互。CSS 部分减小 body 的高度以适应屏幕顶部和底部留白边距，优化视觉效果。代码部分如下：

```
*{
  margin: 5px;
  text-align: center;
}
header{
```

```
    border: 3px solid green;
    height: 10%;
    font-size: 0.7em;

}
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:0.5em;
  }
  #aid{
    position: absolute;
    border: 3px solid blue;
    top:0px;
    left:600px;
  }
```

JavaScript 部分首先通过 window.innerWidth 获取当前视口宽度，判断是否大于 600px，决定 UI.appWidth 的值。如果窗口宽度超过 600px，则固定为 600px，否则取当前窗口宽度，保证了在宽屏下的适配。根据 UI.appWidth 和 UI.appHeight 动态设置 body`的字体大小（基础字体大小与屏幕宽度成比例）和高度，确保文字大小适应屏幕。如下所示：

```
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 + "px";
  document.body.style.height = UI.appHeight - 62 + "px";
  if(window.innerWidth<1000){
    document.getElementById("aid").style.display='none';
}
```

```
    document.getElementById("aid").style.width=window.innerWidth-UI.appWidth-30+'px';
    document.getElementById("aid").style.height= UI.appHeight-62+'px';
```

## 5.3.测试与运行

窄屏和宽屏的用户界面如下图所示。由于本项目已上传至 github 网站，移动端用户可以扫描下方二维码查看本项目。

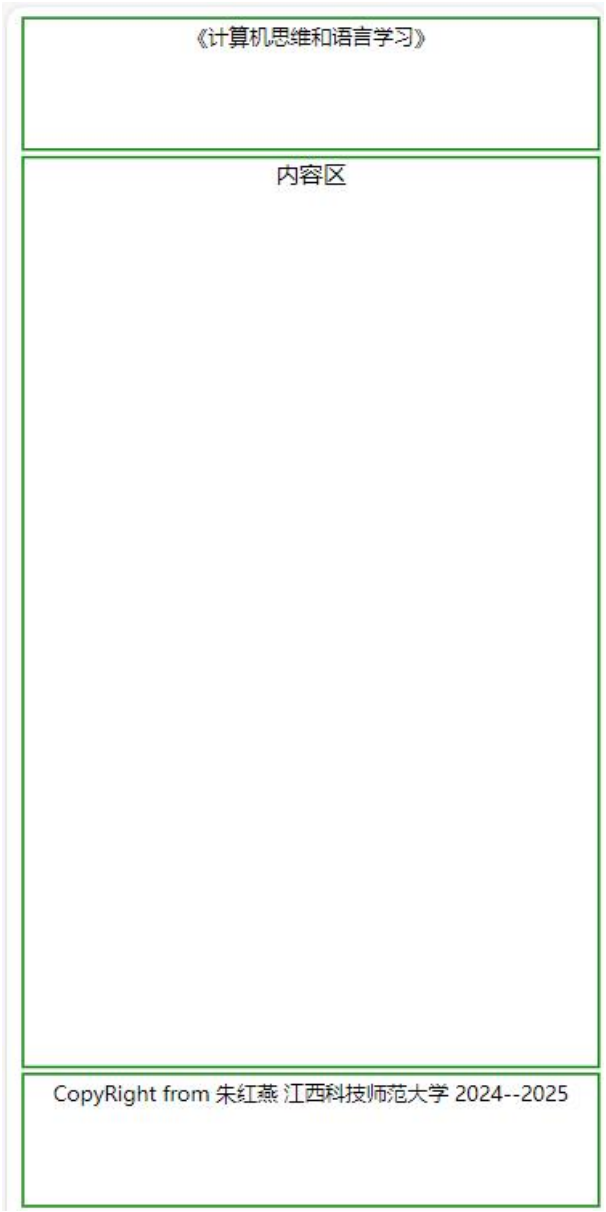


图 14iPhone XR 用户界面


图 15 iPad mini 用户界面


图 16 PC 端用户界面

图 17 项目二维码

# 6. 个性化 UI 设计中对鼠标交互的设计开发

## 6.1.分析与设计

在 JavaScript 中，处理鼠标事件是实现用户交互的重要部分。一个基本的鼠标模型包括监听鼠标的各种动作（如点击、移动、滚轮等），并根据这些动作来改变网页元素的状态或执行特定的功能。在本项目中，需要监听 mousedown，mousemove，和 mouseup 事件，并在适当的时候更新元素的位置，用户界面用例图如下所示。

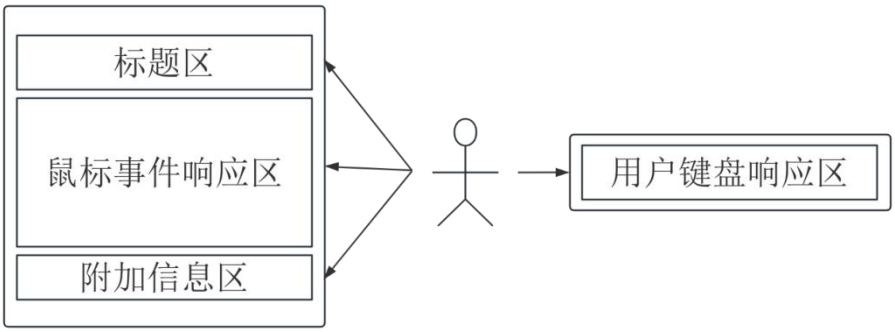


图 18 个性化 UI 实现鼠标交互用户界面用例图

项目 Dom 树如下图所示：

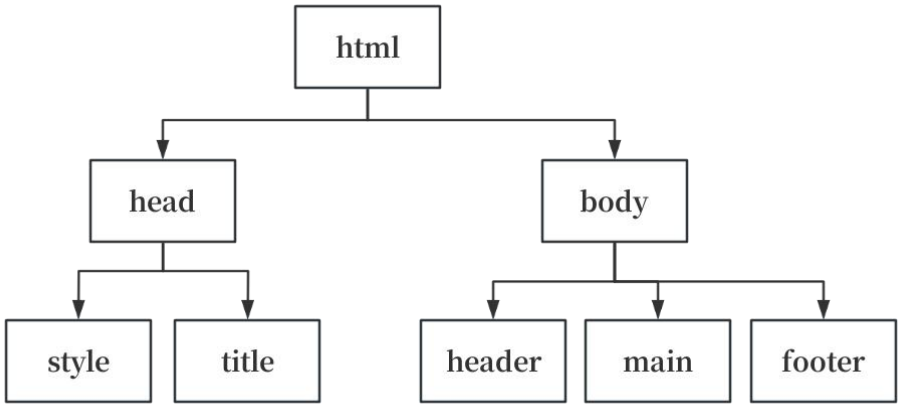


图 19 个性化 UI 实现鼠标交互项目 Dom 树

## 6.2.编程与实现

本项目实现鼠标监听页面元素的代码逻辑如下：定义 Pointer 对象，其属性 isDown 标记鼠标是否按下，x 和 deltaX 分别记录初始位置和位移。当用户在鼠标事件响应区移动鼠标时，显示鼠标正在移动并显示鼠标的位移；当用户在鼠标事件响应区按下鼠标时，显示鼠标按下，并记录鼠标按下时的位移；当用户将鼠标从键盘响应区移开时，显示鼠标已离开。代码如下：

```
//尝试对鼠标设计 UI 控制
var mouse={};
mouse.isDown= false;
mouse.x= 0;
mouse.deltaX=0;
$("bookface").addEventListener("mousedown",function(ev){
    let x= ev.pageX;
    let y= ev.pageY;

    console.log("鼠标按下了，坐标为："+"("+x+","+y+")");
    $("bookface").textContent= "鼠标按下了，坐标为："+"("+x+","+y+")";
});
$("bookface").addEventListener("mousemove",function(ev){
    let x= ev.pageX;
```

```
        let  y= ev.pageY;
        console.log("鼠标正在移动，坐标为："+"("+x+","+y+")");
        $("bookface").textContent= "鼠标正在移动，坐标为："+"("+x+","+y+")";
    });
    $("bookface").addEventListener("mouseout",function(ev){
        console.log(ev);
        $("bookface").textContent="鼠标已经离开";

    });
```

## 6.3.测试与运行

个性化 UI 实现鼠标交互的用户界面如下图所示。由于本项目已上传至 github 网站，移动端用户可以扫描下方二维码查看本项目。


图 20 个性化 UI 实现鼠标交互的用户界面

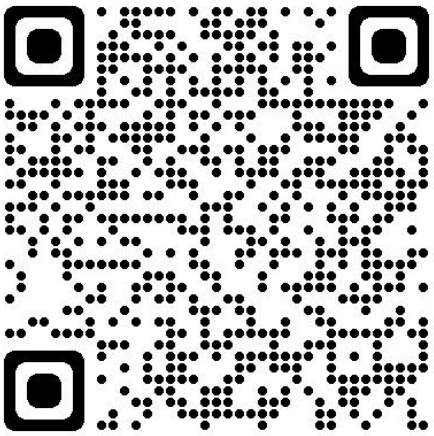

图 21 项目二维码

# 7. 对触屏和鼠标的通用交互操作的设计开发

## 7.1.分析与设计

触屏和鼠标控制的统一处理有助于实现更完善的响应式设计，确保在不同屏幕尺寸和输入方式下，UI 布局和功能均能正确响应和适应，符合现代网页设计标准。本项目用户界面用例图如下所示：

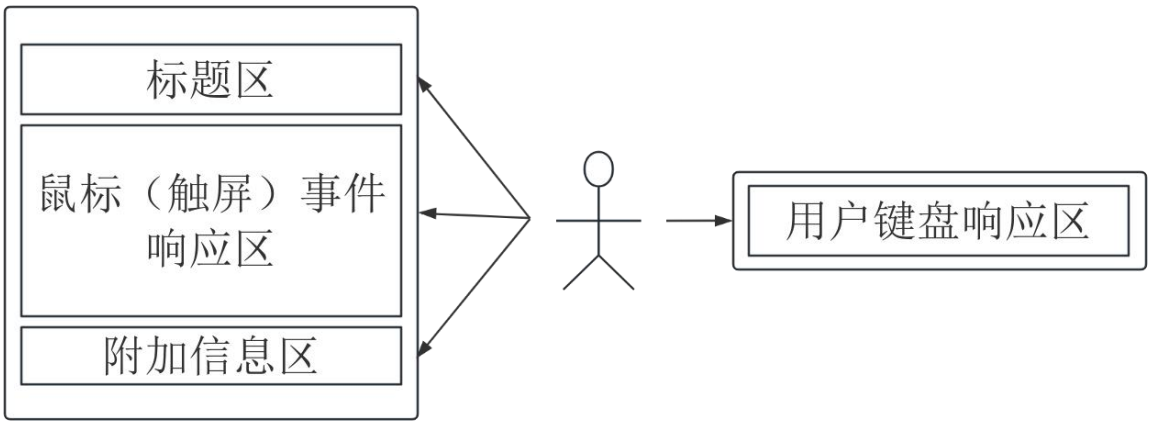


图 22 触屏和鼠标的通用交互项目用户界面用例图

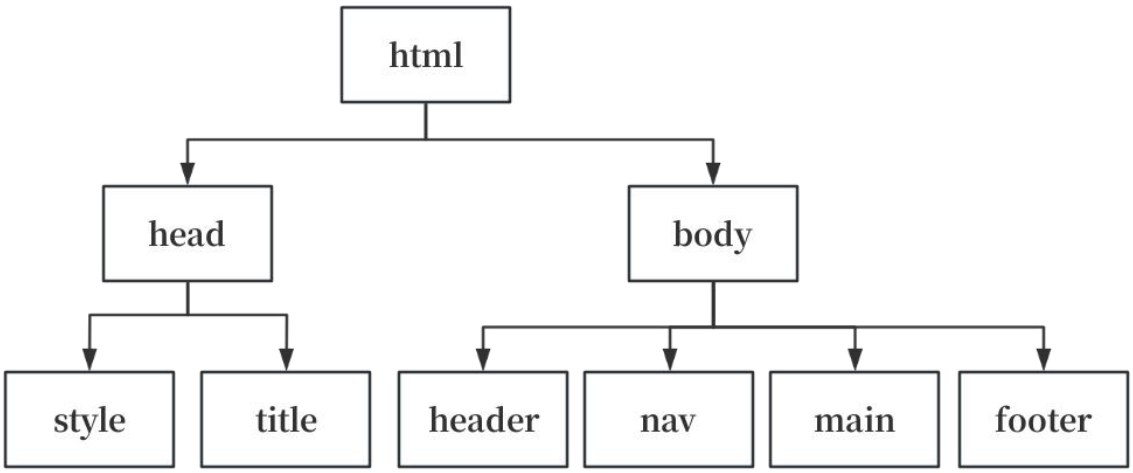


图 23 触屏和鼠标的通用交互项目 Dom 树

## 7.2.编程与实现

本项目首先通过检查窗口的宽度 window. innerWidth 来设定一个应用宽度 UI. appWidth，确保界面在不同设备上的适配性。如代码块 4-2 所示；再通过 ev. touches 判断触屏事件和鼠标事件，再进行相应处理。如处理触屏开始或鼠标按下事件：如果事件包含触控信息，则处理触控事件；否则处理鼠标事件。记录按下时的位置信息，并更新界面上的提示文本，如以下代码所示：

```
//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 ;
});
```

## 7.3.测试与运行

触屏和鼠标的通用交互的用户界面如下图所示。由于本项目已上传至 github 网站，移动端用户可以扫描下方二维码查看本项目。


图 24 触屏和鼠标的通用交互项目用户界面

图 25 项目二维码

# 8. UI 的个性化键盘交互控制的设计开发

## 8.1.分析与设计

Keydown 事件在用户按下按键时触发，允许程序即时响应，捕捉按键序列，实现快捷键操作、连续动作输入等功能，提升了操作效率与流畅度。而 Keyup 事件则在按键释放时触发，这对于判断动作完成、避免按键粘连误判至关重要。结合二者，开发者能创建复杂的键盘交互逻辑，如长按重复动作、组合键功能激活等，为用户带来更贴近自然操作习惯的界面体验。本项目在宽屏界面中设计了用户键盘响应区，用户界面用例图如下图所示：

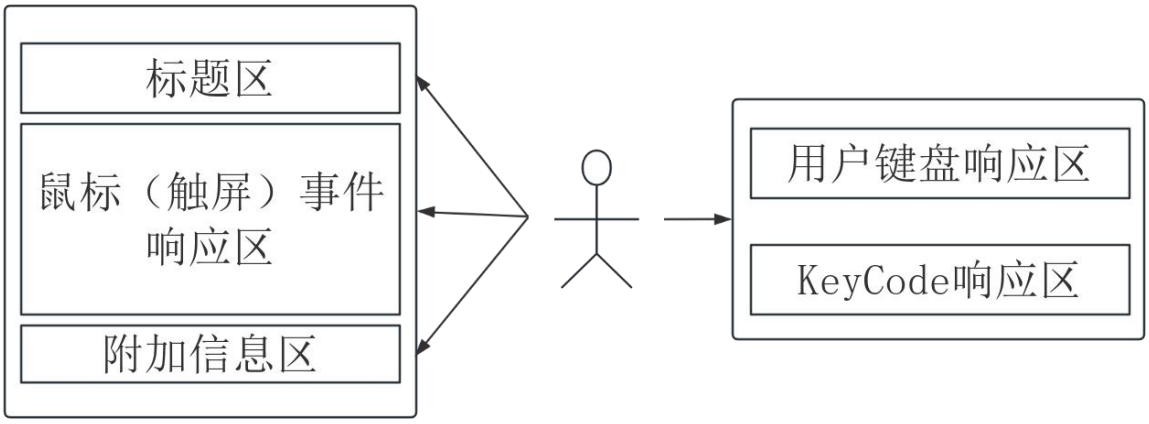


图 26 个性化键盘交互项目用户界面用例图

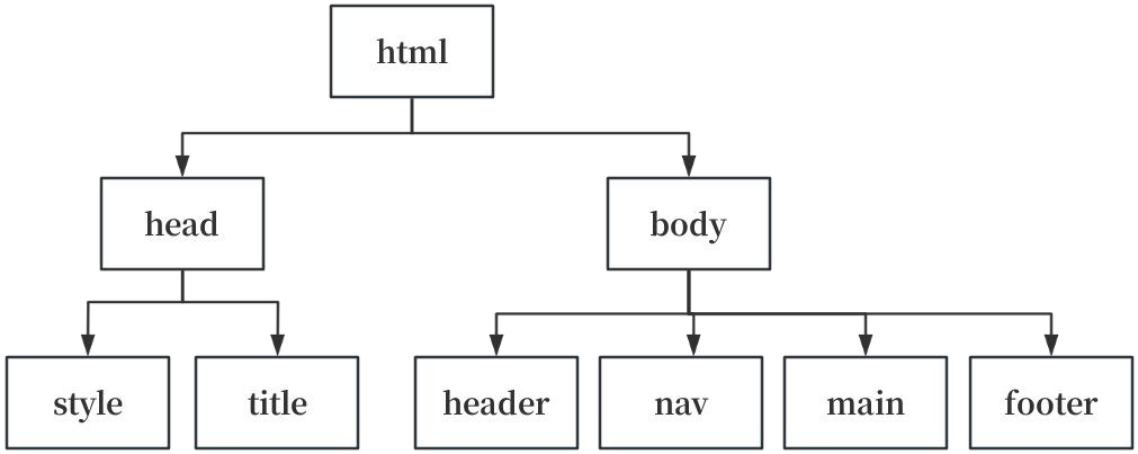


图 27 个性化键盘交互项目 Dom 树

## 8.2.编程与实现

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 ;
  }
```

可打印字符的处理：

```
  function printLetter(k){
  if (k.length > 1){ //确保函数只处理单个、有效的可打印字符
      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 ;
           }
    });
```

## 8.3.测试与运行

个性化键盘交互控制的用户界面如下图所示。由于本项目已上传至 github 网站，移动端用户可以扫描下方二维码查看本项目。


图 28 个性化键盘交互控制的用户界面

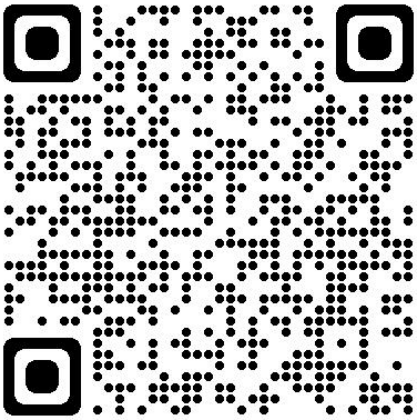

图 29 项目二维码

# 9. 用 git 工具管理代码的版本和项目发布

## 9.1.Git 介绍

作为专业开发软件人员，版本控制是必须要掌握的一种软件工程技术，它是指对软件开发过程中各种程序代码、配置文件及说明文档等文件变更的管理其一大特点是通过记录每一次文件的修改，使得团队成员能够协作开发，同时保持代码的可追溯性和历史记录。市场上已经有许多工具来实现这项技术，主要分为以下两类：集中式版本控制系统 （VCS） 和分布式版本控制系统 （DVCS），其中，Git 就是一种典型的分布式版本控制系统(DVCS)工具。

Git 和其他分布式版本控制系统 （DVCS） 与经典的集中式版本控制系统 （VCS）（如 Subversion）之间的最大区别在于，没有中央服务器可以托管代码仓库。但是，用户可以拥有许多远程服务器。这使用户可以将代码下载至本地从而拥有多个工作副本，以便在其中获取或提取修改，提高文件管理的灵活性[6]。

## 9.2.通过 GitHub 平台实现本项目的全球域名

Git 在构建时考虑到了合作的问题，它使得计算机之间不需要通过中央服务器就可以共享数据，因此 Git 远程就是另一台计算机，每台在共享相同代码仓库的计算机都可以成为其他计算机的远程，从这一角度来看，Git 远程仓库时用户在本地创建同一 Git 代码仓库的远程副本，通过网络的连接，用户可以保持修改与远程仓库同步。而 GitHub 提供了无限的免费公共存储库供用户使用，一个存储库就可以看做是一个远程仓库。

GitHub 是基于 Git 的一个代码托管平台，它为开发者提供了一个存储、分享、协作开发代码的云空间[7]。作为开源代码库以及版本控制系统，随着越来越多的应用程序转移到了云上，GitHub 已经成为了管理软件开发以及发现已有代码的首选方法。

GitHub 为所有用户提供免费的公共存储库，在注册了一个新账户后，随即创建个人的存储库，这就是一个远程仓库。

注册完成后，点击右上角的绿色按钮新建代码仓库：

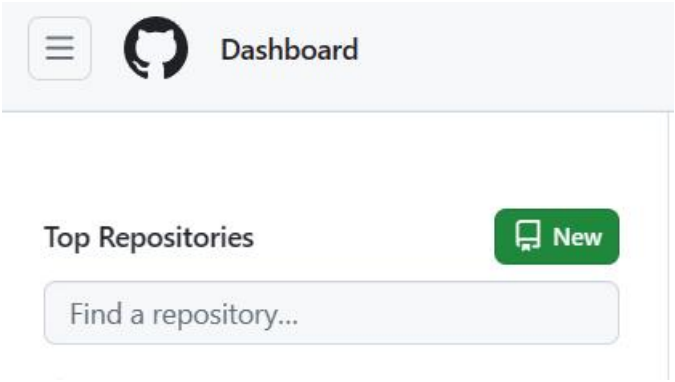


图 30 新建仓库

为了后续直接通过 GitHub 为每个用户分配的域名直接访问仓库里的文件，仓库命名最好要规范：


图 31 仓库命名示例

## 9.3.设置本地仓库和远程代码仓库的连接

分布式工作模式将远程服务器作为不同开发人员的联络点，因此建立远程与本地的连接是其中的核心环节。

进入本地 WebUI 项目的文件夹后，用 git remote add origin + 远程仓库地址命令建立本地仓库与目标远程仓库的关联：

```
86187@LAPTOP-MFOQMM5J MINGW64 /d/0327/abc (main)
$ git remote add origin https://github.com/zhuhyan/zhuhyan.github.io.git
```

图 32 建立本地仓库与远程仓库的关联

此命令也可在 GitHub 成功创建仓库的页面中复制：

...or create a new repository on the command line

```
echo "# zhuhyan1215.github.io" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/zhuhyan/zhuhyan1215.github.io.git
git push -u origin main
```

添加远程后，使用 git push -u origin main 将本地更改推送到远程：

```
86187@LAPTOP-MFOQMM5J MINGW64 /d/0327/abc (main)
$ git push -u origin main
Enumerating objects: 102, done.
Counting objects: 100% (102/102), done.
Delta compression using up to 12 threads
Compressing objects: 100% (68/68), done.
Writing objects: 100% (102/102), 4.25 MiB | 1004.00 KiB/s, done.
Total 102 (delta 32), reused 102 (delta 32), pack-reused 0
remote: Resolving deltas: 100% (32/32), done.
To https://github.com/zhuhyan/zhuhyan.github.io.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

图 33 将本地更改推送到远程

本地仓库与远程仓库连接后，用户在本地修改代码的工作完成后，可通过 git push 命令上传到远程仓库。需要注意的是，使用 git push 命令时，系统会检测仓库中的文件是否有改动，若无改动则上传失败，具体步骤如下：

(1) git add 命令告诉 Git 哪些文件的修改应该包含在下一次提交（commit）中：git add [file1] [file2] ...：添加一个或多个文件到暂存区；

git add [dir]：添加指定目录到暂存区，包括子目录；

git add . ：添加当前目录下的所有文件到暂存区；

(2) git commit -m 命令添加修改备注

git status 命令查看项目的当前状态

```
86187@LAPTOP-MFOQMM5J MINGW64 /d/0327/abc (main)
$ git add index.html

86187@LAPTOP-MFOQMM5J MINGW64 /d/0327/abc (main)
$ git commit -m "修改了标题"
[main c9ea696] 修改了标题
 1 file changed, 1 insertion(+), 1 deletion(-)

86187@LAPTOP-MFOQMM5J MINGW64 /d/0327/abc (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

图 34 本地修改代码

(3) git push 命令将代码提交到远程仓库中。（受网络因素影响可能导致提交失败，重复提交即可）

(4) git log 命令查看修改日志。

```
86187@LAPTOP-MF0QMM5J MINGW64 /d/0327/abc (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 312 bytes | 156.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/zhuhyan/zhuhyan.github.io.git
   46e2473..c9ea696  main -> main

86187@LAPTOP-MF0QMM5J MINGW64 /d/0327/abc (main)
$ git log
commit c9ea696bc882c9e2b50464213c4b0aa035e5d7f1 (HEAD -> main, origin/main)
Author: zhuhy <2942503667@qq.com>
Date:   Mon Apr 22 10:59:13 2024 +0800

    修改了标题
```

图 35 提交代码并查看日志

远程代码上传后，项目实现了在互联网的部署，用户可以通过域名或二维码打开并访问此项目，在仿真设备 iPhone12 pro 中打开本项目的效果如下图所示：


图 36 PC 端访问本项目

# 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] Mozilla Developer Network. (n.d.). MDN Web Docs. https://developer.mozilla.org/

[3] Roth, R. M. (2019). Foundations of Computer Science: From Algorithms to Data Structures. Pearson.

[4] John Dean,PhD. Web programming with HTML5,CSS,and JavaScript[M]. Jones & Bartlett Learning,LLC. 2019: 2

[5] Keith J. Grant. CSS in Depth[M]. Manning Publications, 2018: 3rd edition.

[6] William Shotts. The Linux Command Line, 2nd Edition [ M ]. No Starch Press, Inc, 245 8th Street, San Francisco, CA 94103, 2019：3-7

[7] Santacroce, F. (2023). Git mastering version control. Packt Publishing.