基于 Web 客户端技术的个性化 UI 的设计和编程

（CustomizedUIdesignandProgrammingbasedonWebclienttechnology）

科师大元宇宙产业学院 2021 级张文宇

摘要：近十年来，基于 HTML5 核心的 Web 标准软件开发技术因其跨平台和开源的优势，已广泛应用于各类领域的应用软件开发中。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，我们初步掌握了 HTML 内容建模、CSS 样式设计和 JavaScript 功能编程的基本技术和技巧。本项目基于 Web 客户端技术开发，以适应移动互联网时代 Web 应用的前端需求并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 能够较好地适配各种屏幕尺寸。在功能实现方面，项目通过 DOM 技术和事件监听，实现了对鼠标、触屏和键盘底层事件的支持，并确保了响应的流畅性。基于面向对象的思想，我们为鼠标和触屏设计了对象模型，并通过代码模拟了这些指向性设备的操作。为了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，共进行了七次项目迭代开发，每次迭代都经历了 ADIT（分析、设计、实施、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。为了分享和共享代码，并与其他开发者合作，我们使用了 Git 工具进行代码和开发过程日志记录。总共进行了七次代码提交操作，详细记录并展示了开发思路和代码优化的过程。然后，通过 Git Bash 将项目上传到 GitHub 并建立了自己的代码仓库，将该代码仓库设置为 HTTP 服务器，实现了全球便捷访问。通过采用 HTML5 技术进行全面开发和优化，本项目展示了跨平台和响应式设计的优势，同时通过严格的工程管理和版本控制，实现了高质量的代码和广泛的应用部署。

关键词：Web 客户端技术；Git；UI；

1. 引言

UI 设计对于计算机行业至关重要，因为它直接影响到用户体验、产品功能实现和整体软件质量。UI 设计通过合理布局、清晰的操作流程和友好的交互设计，使得用户能够更加轻松、高效地使用软件，从而提升用户满意度。优秀的 UI 设计不仅注重外观上的美观，更重要的是能够有效支持软件功能的实现。合理的布局、明确的功能入口和交互设计有助于用户快速找到所需功能，提高软件的实用性和功能性。良好的用户界面设计可以降低用户的学习成本。通过直观的操作界面、清晰的提示信息和一致的交互逻辑，用户可以更容易上手，减少学习和适应的时间，提高使用效率。UI 设计是软件质量的重要组成部分。合理的交互设计、兼容性考虑和用户反馈机制有助于发现和解决潜在问题，提升软件的稳定性、安全性和可靠性。精心设计的用户界面可以成为产品推广的有力工具。优秀的界面设计不仅能吸引用户注意力，还能传达产品的核心价值和特色，提升产品竞争力，促进产品的推广和营销。随着技术领域的扩展和产品生产的人性化趋势增强，UI 设计师已成为人才市场上的紧缺职位。据市场调查显示，中国在 UI 设计领域存在大量的工作岗位空缺，显示出 UI 设计在计算机行业中的重要性和广阔的发展空间。

1.1 项目概述

本次课程设计的目标是基于 HTML5 技术，设计和实现一个个性化的用户界面（UI）应用程序。通过分析项目需求，我们选择 HTML5 为核心技术路线，结合 CSS 进行 UI 设计，利用 JavaScript 实现交互功能。为保证代码质量和可维护性，项目采用面向对象设计思想和响应式设计，以适应不同设备屏幕的需求。整个开发过程中，我们手工编写每一行代码，深入理解 Web 客户端技术的核心原理。项目采用增量式开发模式，经过多次迭代和重构，不断优化代码，实现了设计、开发和测试的有机结合。为了更好地进行版本控制和代码管理，我们使用 Git 工具，将项目托管在 GitHub 平台上，并通过 GitHub 的 HTTP 服务器实现全球互联网部署，使用户能够便捷地访问和体验该程序。本论文将详细介绍本次课程设计的技术路线、开发过程、功能实现及代码管理方法。

1.2 研学计划

提交次数	目标
第一次	实现网页窄屏代码，并进行个性化 UI 设计中的鼠标模型 1.0 的探究。利用鼠标监听事件，显示鼠标的实时坐标。
第二次	实现个性化 UI 设计中的鼠标模型 2.0，支持触屏功能。利用鼠标监听事件，实现页面元素的拖动，并显示拖动距离。
第三次	实现个性化 UI 设计中的键盘控制 1.0，利用 keydown 和 keyup 事件，显示按下键的字符以及 key 和 keyCode 值。
第四次	实现个性化 UI 设计中的键盘控制 2.0，利用 for 循环和条件判断进行筛选，只显示单个字符。
第五次	实现个性化 UI 设计中的键盘控制 3.0，处理特殊键 Back 和 Enter，通过监听事件实现文本删除与换行功能。
第六次	对整个应用程序进行全面的测试，发现并修复 BUG，优化代码性能，确保程序的稳定性和高效性。

1.3 研究方法

使用原型设计法，使用工具绘制了界面原型，确保设计方案的可行性。其次，我们利用增量开发法，采用迭代增量式开发模式，将整个项目划分为若干小模块，逐步开发和测试，且每次都对已有的功能进行优化，确保项目的质量的进度。最后，我们采用实验测试法和版本控制法，在浏览器的开发者面板中进行调试修改，并使用 Git 进行版本控制和代码管理，通过 GitHub 平台进行项目的托管和全球互联网部署。为了确保本项目的科学性和实用性，本论文采取了六种研究方法。

2.Web 平台和客户端技术概述

2.1Web 平台

Web 平台的历史可以追溯到 1989 年，当时 Tim Berners-Lee 在欧洲粒子物理研究所（CERN）提出了万维网（World Wide Web）的概念。1990 年，他开发了世界上第一个网页浏览器，并在 1991 年 8 月 6 日通过 alt.hypertext 新闻组向全世界介绍了这个项目，并邀请广泛的参与与合作。 Web 平台的发展经历了几个重要阶段，包括静态网页时代、动态网页技术的出现、CSS 的引入、AJAX 技术的普及，以及 Web 2.0 和 Web 3.0 的概念提出。这些技术的发展极大地丰富了 Web 平台的功能，使得 Web 平台从一个简单的信息发布和浏览工具转变为一个

强大的交互和应用平台。至今，Web 平台仍然在不断进化，新的技术和理念不断涌现，推动着 Web 平台向着更加智能化、个性化和互动化的方向发展。

2.2Web 编程

Web 编程是一个很大的领域，不同类型的 Web 编程由不同的工具实现。但是，基本上所有的工具都使用核心语言 HTML。其中，HMTL5，CSS 和 JavaScript 这三种技术被认为是 Web 编程的主要支柱。

2.3 软件开发的过程管理——增量式开发模式

增量式开发模式是一种软件工程方法，它强调将大型项目分解成一系列小的、可管理的部分，然后逐步完成这些部分。这种方法的关键在于每个小部分都能独立工作，并且可以在整个项目完成之前被测试和验证。

2.3.1 瀑布模型

瀑布模型（Waterfall Model）是一种经典的软件开发方法，它将软件开发过程视为一系列顺序排列的阶段，每个阶段都有特定的活动和产出物。这些阶段包括需求分析、系统设计、实现编码、集成和测试、以及维护。每个阶段的输出成为下一个阶段的输入，类似于瀑布流水的连续过程。

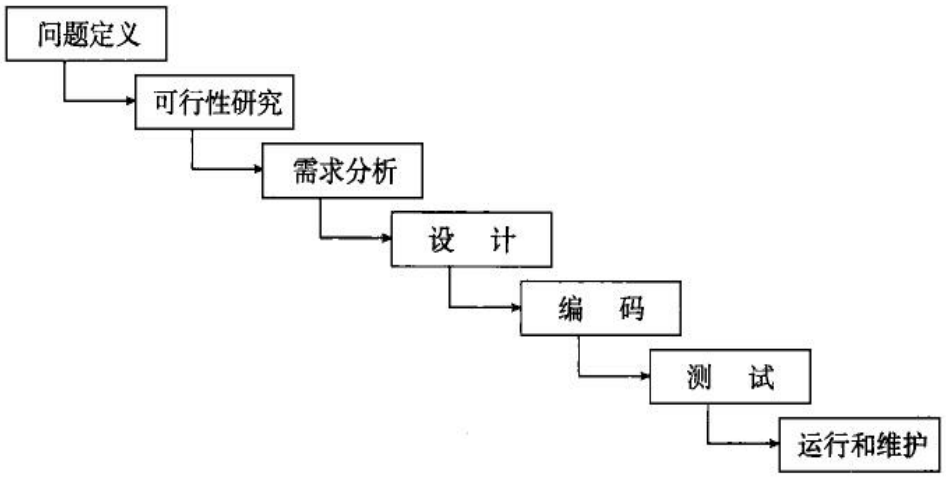

图 2.1 瀑布模型

2.3.2 增量模型

增量模型（Incremental Model）是一种软件开发方法，它将软件产品视为一系列的增量构件来设计、编码、集成和测试。每个构件由多个相互作用的模块构成，并且能够完成特定的功能。使用增量模型时，第一个增量构件往往实现软件的基本需求，提供最核心的功能。把软件产品分解成增量构件时，唯一必须遵守的约束条件是，当把新件集成到现有构件中时，所形成的产品必须是可测试的。

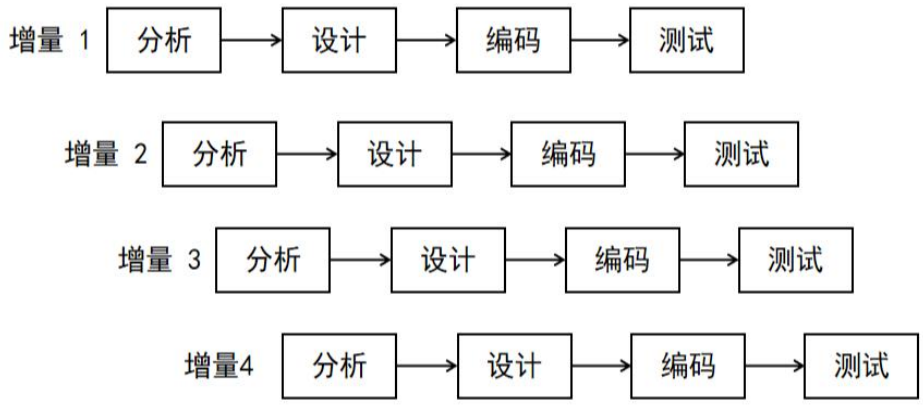

图 2.2 增量模型

3. 页面结构设计

首先，我利用 HTML5 创建了页面的结构模型，然后再分别创建子元素 header、main、footer 用于分别存储标题，作品内容以及个人版权等信息。其中，实现方式如代码块 3.1 所示。

```
<header>
Design Works Display
</header>
<main>
Poster Content
</main>
<footer>
&copy; 2024 张文宇 1518273854qq.com
</footer>
```

代码块 3.1

4.移动互联时代的响应式设计

本阶段初次引入了 CSS 中比较高阶的语法，如 em 和%单位，这些单位在响应式设计中非常有效。通过利用%单位，我给 header，body，footer 分别赋予了相对于父容器的比例，如 15%，75%，10%。而通过 em 单位，继承 body 的属性，灵活地设置了字体的大小。如代码块 4.1 和 4.2 所示。

```
<style>
* {
margin: 0;
color: grey;
text-align: center;
align-content: center;
justify-content: center;
box-sizing: border-box;
```

```
background-color: ghostwhite;
font-family: Karla, sans-serif;
}
header {
height: 15%;
display: flex;
border: 5px solid black;
border-bottom: none;
}
11
#title {
font-size: 2em;
}
main {
height: 75%;
font-size: 0.8em;
border: 5px solid black;
display: flex;
}
.poster {
width: 75%;
font-size: 2em;
background-color: bisque;
}
footer {
height: 10%;
font-size: 1em;
border: 5px solid black;
border-top: none;
}
</style>
```

代码块 4.1

```
// 创建一个名为 UI 的对象，用于存储系统的宽度和高度信息
var UI = {};
// 记录系统窗口的宽度和高度，并限制宽度不超过 600px
UI.deviceWidth = window.innerWidth > 600 ? 600 : window.innerWidth;
UI.deviceHeight = window.innerHeight;
// 计算默认字体大小
const Letters = 33; // 字母数量
const baseFont = UI.deviceWidth / Letters; // 基准字体大小
// 设置页面的字体大小为默认字体大小
document.body.style.fontSize = baseFont + 'px';
// 通过动态 CSS 设置页面全屏显示
document.body.style.width = UI.deviceWidth + 'px';
```

```
document.body.style.height = UI.deviceHeight + 'px';
```

代码块 4.2

5. 个性化交互 UI 初步——鼠标模型的设计

考虑到信息量的多少，我决定将鼠标的坐标信息显示在 main 元素的子元素.poster14 中。我首先创建了一个 mouser 对象，用于存储鼠标的部分信息如 mouseisDown，mouse.x 和 mouse.deltaX 等。然后，我添加三个监听事件，mousedown、mousemove、mouseout 用于获取鼠标按下以及移动时的坐标。其中，代码实现方式如代码块 5.1 所示。

```
// 监听鼠标在 .poster 区域的按下事件
$(.poster).addEventListener(mousedown, function (ev) {
// 获取鼠标按下时的坐标，并在 .poster 区域显示鼠标按下的信息及坐标
let x = ev.pageX;
let y = ev.pageY;
$(.poster).textContent = 鼠标按下了，坐标为： + ( + x + , + y + );
});
// 监听鼠标在 .poster 区域的移动事件
$(.poster).addEventListener(mousemove, function (ev) {
// 获取鼠标当前坐标，并在 .poster 区域显示鼠标移动的信息及坐标
let x = ev.pageX;
let y = ev.pageY;
$(.poster).textContent = 鼠标正在移动，坐标为： + ( + x + , + y + );
});
```

代码块 5.1 鼠标模型实现

6. 探索 UI 对鼠标拖动的操作模式

首先，我定义了一个名为 mouse 的对象，用于存储与鼠标相关的状态和信息，例如鼠标是否按下、鼠标的位置等。然后，我为.poster 元素添加了四个事件监听器。当鼠标按下时，记录下鼠标的位置，并将 mouse.isDown 设置为 true；当鼠标松开时，将 mouse.isDown 设置为 false，并根据水平移动距离判断拖动是否有效；当鼠标移出元素时，同样将 mouse.isDown 设置为 false，并根据移动距离判断拖动是否有效。其中，具体实现如代码块 6.1 和 6.2 所示。

```
//尝试对鼠标设计 UI 控制
var mouse={};
mouse.isDown= false;
mouse.x= 0;
mouse.y= 0;
mouse.deltaX=0;
$(.poster).addEventListener(mousedown,function(ev){
mouse.isDown=true;
mouse.x= ev.pageX;
mouse.y= ev.pageY;
console.log(mouseDown at x: +(+mouse.x +, +mouse.y +) ) ;
$(.poster).textContent= 鼠标按下，坐标：+(+mouse.x+,+mouse.y+);
});
```

```
$(.poster).addEventListener(mouseup,function(ev){
mouse.isDown=false;
$(.poster).textContent= 鼠标松开!;
if(Math.abs(mouse.deltaX) > 100){
$(.poster).textContent += ，这是有效拖动！ ;
}else{
$(.poster).textContent += 本次算无效拖动！ ;
}
});
$(.poster).addEventListener(mouseout,function(ev){
ev.preventDefault();
mouse.isDown=false;
$(.poster).textContent= 鼠标松开!;
if(Math.abs(mouse.deltaX) > 100){
$(.poster).textContent += 这次是有效拖动！ ;
17
}else{
$(.poster).textContent += 本次算无效拖动！ ;
}
});
$(.poster).addEventListener(mousemove,function(ev){
ev.preventDefault();
if (mouse.isDown){
console.log(mouse isDown and moving);
mouse.deltaX = parseInt( ev.pageX - mouse.x );
$(.poster).textContent= 正在拖动鼠标，距离： + mouse.deltaX +px 。;
$(.poster).style.left = mouse.deltaX + 'px' ;
}
});
```

代码块 6.1

```
// 触屏事件处理
var touch = {
startX: 0,
startY: 0,
deltaX: 0
};
$(".poster").addEventListener("touchstart", function (ev) {
var touchEvent = ev.touches[0];
touch.startX = touchEvent.pageX;
touch.startY = touchEvent.pageY;
$(".poster").textContent = "触屏开始，坐标：" + "(" + touch.startX
+ "," + touch.startY + ")";
});
$(".poster").addEventListener("touchmove", function (ev) {
```

```
ev.preventDefault();
var touchEvent = ev.touches[0];
touch.deltaX = parseInt(touchEvent.pageX - touch.startX);
$(".poster").textContent = "正在触屏滑动，距离：" + touch.deltaX +
"px 。";
$(".poster").style.left = touch.deltaX + 'px';
});
18
$(".poster").addEventListener("touchend", function (ev) {
$(".poster").textContent = "触屏结束!";
if (Math.abs(touch.deltaX) > 100) {
$(".poster").textContent += "，这是有效滑动！";
} else {
$(".poster").textContent += " 本次算无效滑动！";
}
});
```

代码块 6.2

7. 个性化 UI 对键盘控制-单个字符输入

首先，我创建了一个 元素，并给它分配了一个特定的 ID，命名为 aid，方便我在后续代码中引用。接下来，便是在这个大的 div 中填充内容。首先，我添加了一个标题区域，用一个<p>元素来显示，对标题进行了简单的说明。然后，是文本输入内容的部分。我创建了另一个 元素，其 ID 设置为 keyboard，用于容纳后续实现换行功能的内容。在这个 中，我添加了一个<p>元素，用于存储用户输入字符的第一行内容。这个<p>元素将在后续的交互中不断更新以显示用户的输入。最后，我添加了一个<p>元素，用于显示与输入字符相关的信息，作为尾部。通过这样的设置，就实现了一个具有清晰结构的键盘响应区域，为后续的功能扩展做好了准备，它的具体实现如代码块 7.1、7.2 和 7.3 所示。

```
<div id=aid>
<p id=respond>用户键盘响应区域</p>
<div id=keyboard>
<p id=firstLine>
 
</p>
</div>
21
<p id=status>
 
</p>
</div>
```

代码块 7.1 添加文本编辑区

```
$(body).addEventListener(keypress, function(ev){
$(firstLine).textContent += ev.key ;
});
```

```
$(body).addEventListener(keydown, function (ev) {
let k = ev.key;
let c = ev.keyCode;
$(status).textContent = 您已按下键 ： + k + ， + 字符编码 ： + c;
});
```

代码块 7.2 字符的即时显示

```
/*判断是否是单个字符*/
function printLetter(k) {
//判断字符串长度是否大于 1
if (k.length > 1) {
return false;
}
let puncs = ['~', '`', '!', '@', '#', '$', '%', '^', '&', '*', '(',
')', '-', '_', '+',
'=', ',', '.', '<', '>', '?', '/', ', ' '];
/*字母输出*/
if ((k >= 'a' && k <= 'z') || (k >= 'A' && k <= 'Z') || (k >= '0'
&& k <= '9')) {
return true;
}
/*符号输出*/
22
for (let p of puncs) {
if (p === k) {
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？
}
});
```

代码块 7.3 判读是否为单个字符

8.UI 的个性化键盘交互控制

8.1 设计与分析

个性化 UI 对键盘对键盘的控制

图 8.1UI 的个性化键盘响应用例图

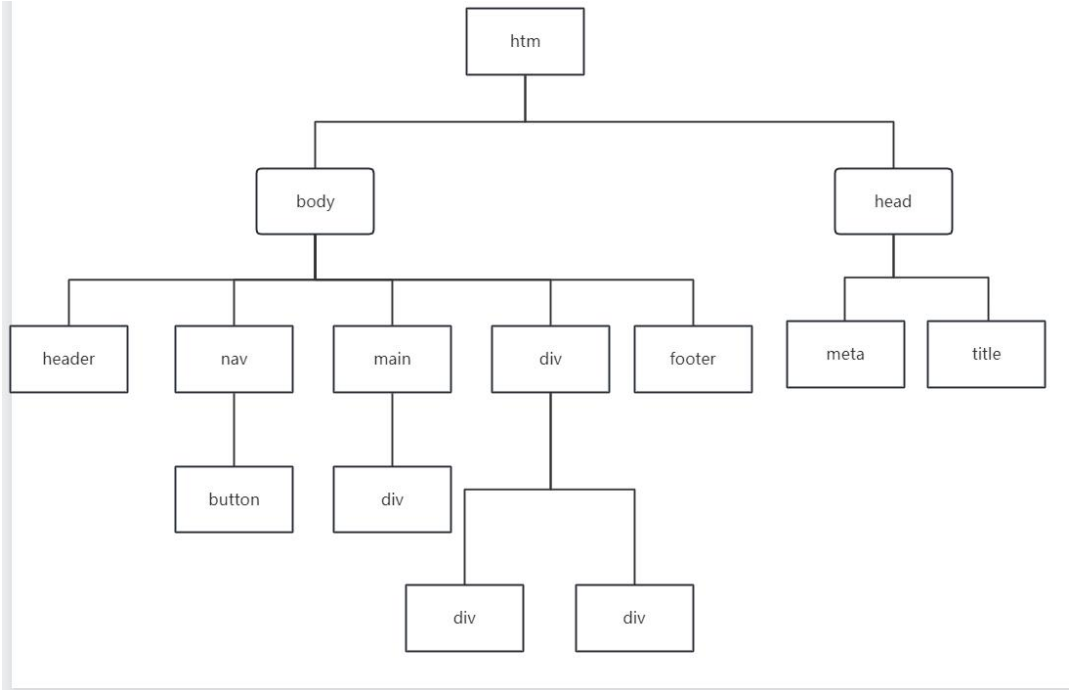

图 8.2 UI 的个性化键盘响应 DOM 树

8.2 编程与实现

基于图 8.1UI 的个性化键盘响应用例图而编写代码如下：

```
$(body).addEventListener(keyup, function (ev) {
let k = ev.key;
let c = ev.keyCode;
$(status).textContent = 松开按键 ： + k + ， + 字符编码 ： + c;

/*判断键盘按下的是否为 Enter 键，如果是添 p 元素实现换行*/

if (k === Enter) {//有且只能在 document 中创建子节点
```

```
let p = document.createElement(p);
                    //通过创建 p 元素，添加子节点来实现换行。
$(keyboard).appendChild(p);
}
else if (k === Backspace)
{

/*没有字符可以删除了，则将该子节点删除*/

if ($(keyboard).lastElementChild.textContent === ) {

/*删除前保证 keyboard 中至少有一个字节点*/

if ($(keyboard).childElementCount > 1) {
$(keyboard).removeChild($(keyboard).lastElementChild);
}
} else {
$(keyboard).lastElementChild.textContent =
$(keyboard).lastElementChild.textContent.slice(0, -1);
}
}
else if (printLetter(k))
{
$(keyboard).lastElementChild.textContent += k;
}
```

代码块 8.3 UI 的个性化键盘响应代码

8.3 测试与运行

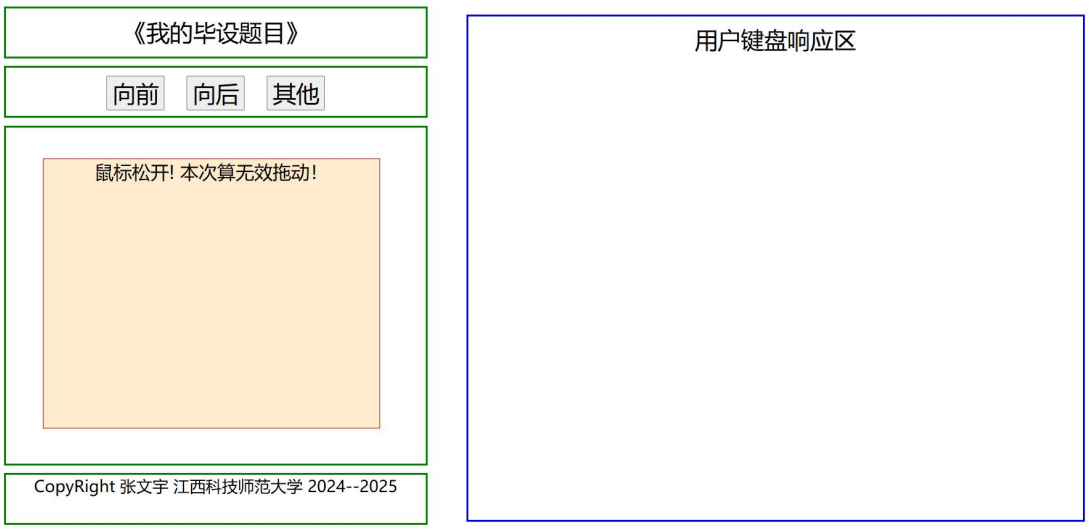

图 8.4

8.4 代码提交和版本管理

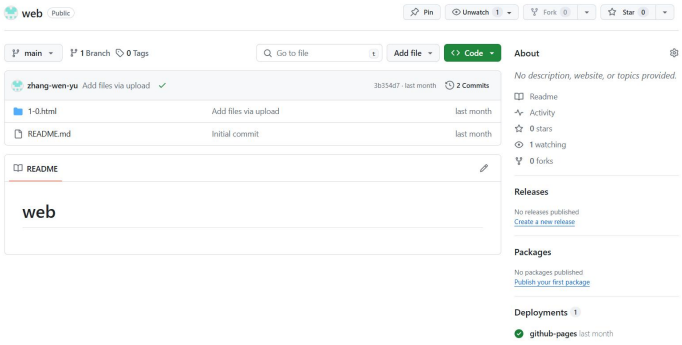

图 8.5 项目代码第六次提交

图 8.6 WebUI_v6.0 移动端二维码