

成绩:

江西科技师范大学

课程设计（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）：Personalized UI design and implementation based on Web
client technology

院（系）：元宇宙产业学院

专 业：计算机科学与技术

学生姓名：张紫阳

学 号：20213570

指导教师：李健宏

2024 年 6 月 19 日

目录

1. 前言	1
1.1 毕设任务分析	1
1.2 研学计划	1
1.3 研究方法	1
2. 技术总结和文献综述	2
2.1 Web 平台和客户端技术概述	2
2.2 项目的增量式迭代开发模式	2
3. “三段论”式的内容设计概要	3
4. 移动互联时代的响应式设计（窄屏）	5
4.1 分析和设计	5
4.2 项目的实现和编程	6
4.3 项目的运行和测试	7
5. 响应式设计（宽屏和窄屏）	8
5.1 分析和设计	8
5.2 项目的实现和编程	9
5.3. 项目的运行和测试	11
6. UI 设计中鼠标模型的控制	13
6.1 分析和设计	13
6.2 项目的实现和编程	13
6.3. 项目的运行和测试	16
7. 通用的 UI 设计，用一套代码同时为触屏和鼠标建模	17
7.1 分析和设计	17
7.2 项目的实现和编写	17
7.3 项目的运行和测试	20
8. UI 的个性化键盘控制——巧妙应用 keydown 和 keyup 键盘底层事件	21
8.1 分析和设计	21
8.2 项目的实现和编写	22
8.3 项目的运行和测试	24
9. 用 git 工具开展代码版本管理	24
10. 编写高质量代码	27
参考文献	27

摘要:Web 平台是一个基于超文本和 HTTP 的全球性、动态交互的且跨平台的分布式图形信息系统，提供易于访问的直观界面方便用户在互联网浏览信息，在本项目中通过以 Web 客户端技术为研究对象，并且以 HTML 为内容建模、CSS 语言样式设计和 JavaScript 功能实现，完成一个 UI 界面设计，通过响应式设计实现了对 PC 端以及手机端屏幕的适应，本程序还用 Dom 技术和事件驱动模式通过控制 JavaScript 操作网页元素实现了页面动态交互，完成程序对鼠标、触屏、键盘的底层事件的响应和流畅支持，还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，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。本程序使用了软件工程的增量式增强的开发模式处理好设计和开发的关系，逐步实现本项目 UI 编写，为了分享本代码还与网上的开发者共同合作，在此基础上用 git 工具进行代码和开发过程的日志记录，详细的展示制作思路，采用 gitbash 将项目上传至 GitHub 的代码仓库当中，通过将代码仓库设置成为 http 服务器实现 UI 应用全球便捷访问。

关键词: Web；响应式设计；模型；git；UI 设计

1. 前言

1.1 毕设任务分析

毕设任务是一个复杂且系统的过程，涉及多个步骤和环节，首先必须明确毕设的目标以及预期的成果，运用计算机科学技术知识训练编写代码能力，系统性落实分析问题、建立模型、软件设计、系统实施、测试调试等传统流程，根据毕设任务研究方法，进行数据收集和分析，包括文献归纳法和数据分析法等，总结项目进行撰写论文，从实践和理论进行有机结合。

1.2 研学计划

本项目通过六周时间逐步对代码进行完善补充。

时间	任务
第一周	UI 建模
第二周	Web 窄屏响应式设计
第三周	Web 宽屏与窄屏响应式设计
第四周	UI 设计中的鼠标模型分析和控制
第五周	为触屏和鼠标建模的通用 UI 设计可以与跨平台 UI 框架通过抽象和封装底层平台的细节使得一套代码来构建适不同平台的用户界面
第六周	UI 个性化键盘，巧妙应用 keydown 和 keyup 这两个键盘底层事件

1.3 研究方法

计算机科学与技术专业研究方法多样化,涵盖理论研究、实验研究、系统研究等多个方面,理论研究包括定理、证明以及结果等,实验研究是本专业第一手段,包括实验设计、数据分析等,通过分析论证理论,系统研究则注重算法基础理论研究,本专业具有较强的交叉性,需要掌握相关知识且有逻辑思维,

2. 技术总结和文献综述

2.1 Web 平台和客户端技术概述

Web 平台主要涉及到 Web 浏览器、服务器以及它们之间的通信协议。Web 浏览器如 Chrome、Firefox 等,提供了与服务器交互的界面。服务器则负责处理业务逻辑和存储数据,如 Java Servlet 和 JSP 技术所示,这些技术支持跨平台可移植性和灵活性。Web 平台的一个关键组成部分是 HTTP 协议,它是万维网的基础,用于在客户端和服务端之间传输数据。客户端,也称为用户端,是指与服务端相对应,为客户提供本地服务的程序。客户端技术的主要职责是实现用户界面的显示和用户输入的处理。随着技术的发展,客户端不仅仅局限于简单的页面显示,还包括了复杂的交互功能,如动态内容加载、多媒体处理等。Web 编程是一个庞大的领域,不同类型的 Web 编程由不同的工具实现。所有工具都使用核心语言 HTML,众所周知,HTML5、CSS 和 JavaScript 这三种技术是客户端网络编程的支柱,在客户端网页编程中,所有网页计算都在最终用户的计算机(客户端计算机)上进行^[1]。经典的 MVC 设计模式理解 Web 平台架构的三大基石,Model 可以理解为 HTML 标记语言建模,View 可以理解为用 CSS 语言来实现外观,Controller 则可理解为用 JavaScript 结合前面二个层次,实现了在微观和功能层面的代码控制。

2.2 项目的增量式迭代开发模式

本项目为本科学生毕业设计的软件作品,其中手写代码数量较多,需要分步骤逐步完成,可将其视为一个系统工程,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,在软件开发过程管理模式有两种模型:瀑布模型和增量式迭代模型,任何开发模式则都必须同样经历四个阶段:分析(Analysis)、设计(Design)、实施(Implementation)、测试(test)。

瀑布模型将软件生命周期划分几个线性和有序的阶段,这种顺序性确保了项

目的每个部分按计划进行，然而其局限性是高度依赖文档驱动，可能导致产品不能满足需求，然而增量式迭代模型可以将每个模块作为一个增量组件，分批次地分析、设计、编码和测试这些增量组件，本项目也采用了增量模型的开发模式，共做了六次项目的开发迭代。

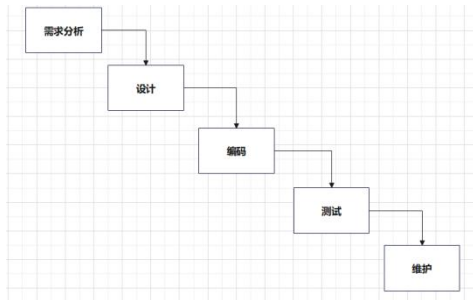


图 2-1. 瀑布模型图

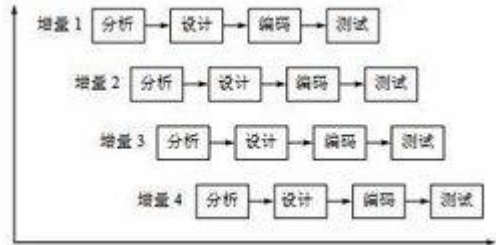


图 2-2. 增量模型图

3. “三段论”式的内容设计概要

UI 内容建模是用户界面设计中通过使用各种工具和技术来创建和管理 UI 元素的过程，他也是一个涉及多个方面的复杂过程，包括但不限于视觉设计、交互方式的整合、排版策略、使用专业工具以及适应新技术的能力。通过综合考虑这些因素，可以创建出既美观又实用的用户界面，本次设计用 HTML 语言实现了内容模型的建立，并且把应用分成<header>、<main>和<footer>三部分，header 页头部分是用户访问时看到的第一个部分，具有较为强烈的吸引力，main 主内容区他包裹着文档中唯一的中心内容，可以清晰展示与页面主题相关或拓展的信息，保证了网页结构的清晰性和语义性，footer 页脚通常包含着网页相关的联系信息与声明可以增强用户的体验，构建用户与网站部分的桥梁。为了使网页具有美观性，本设计采用了 css 语言对项目实现了大致的 UI 外观，图 1 为 css 语言设计外观，采用 border 属性设置元素边框样式、高度和颜色，增强了页面的视觉效果和用户体验。在此架构设计中，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功能设计，lesson 文件夹和 myface 文件夹分别放入项目的各种资源文件和各种图片。

```

<head>
<style>
*{
font-size:30px ;
}
header{
border: 2px solid blue;
height: 200px;
}

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

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

```

图 3-1. 程序的 css 设计

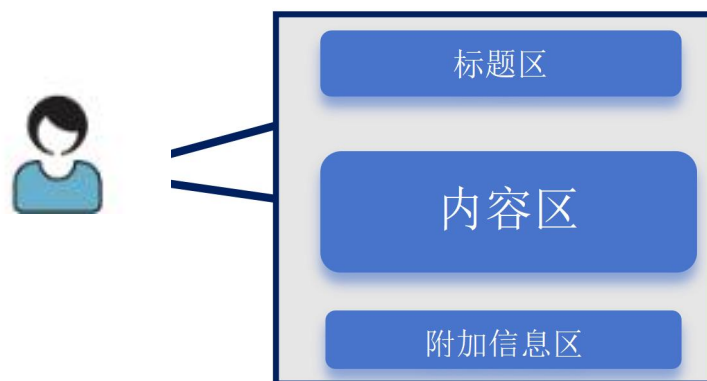


图 3-2. 三段式用例图



图 3-3. PC 端页面展示



图 3-4. 手机端页面展示

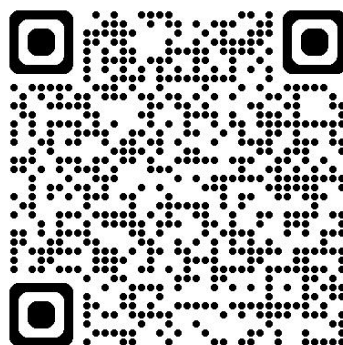


图 3-5. 二维码展示

4. 移动互联时代的响应式设计（窄屏）

4.1 分析和设计

在移动互联时代，响应式设计已成为前端开发的核心技术之一，其主要目的是为了适应不同设备和屏幕尺寸，提供一致且优化的用户体验。计算机使用的显示硬件千差万别，显示屏的大小和分辨率取决于成本。设计者并没有为每种类型的显示器设计不同版本的网页，而是选择让网页提供总体布局指南，并允许浏览器选择如何在特定计算机上显示网页^[2]。因此，网页不会提供太多细节。响应式设计的关键特点包括使用弹性网格布局、媒体查询以及流式布局等技术，随着智

能手机和平板电脑的普及，移动设备已经成为人们访问网站和应用的的首选工具。因此，移动优先的设计原则在前端开发中占据了重要地位。这种设计理念强调的是，网站应当根据访问设备的屏幕尺寸、分辨率和操作系统自动调整布局和视觉效果，以提供最佳的用户体验，在 HTML 中，通过设置元素 height 属性为百分比值，使得元素的高度能够根据其父元素的大小自适应调整，通过把设备的高度全部分配给 body 对象，再结合前面高度的分配实现了各个区域响应式设计，同时为应用计算了基础字体的大小，并且利用了 body 的遗传机制再同时通过 font-size 属性控制改变每个板块的字体大小，实现了字体的响应式设计。交互设计在窄屏环境下有着重要响应式，在此添加<nav>元素，设计导航一、导航二、导航三的三个按钮功能，更加使得页面具有可操控性，提高屏幕空间的利用率，为了提升美感，在本程序中，设置 UI 全局变量，为属性 appWidth 和 appHeight 设置视图的宽度和高度，同时为了更好体验将字体设置为宽度的二十分之一，保证每行最多 20 字。

4.2 项目的实现和编程

一. HTML 字体响应式设计编写

```
<script>
var UI = {};
UI.appWidth = window.innerWidth;
UI.APPHeight = window.innerHeight;
let baseFont = UI.appWidth / 20 ;
//更改 body 字体大小影响子子孙孙
document.body.style.fontSize = baseFont + "px";
document.body.style.height = UI.APPHeight + 170 + "px"
```

二. CSS 编写

```
*{
    margin: 10px;
    text-align: center;
}
header{
    border: 3px solid red;
    height: 15%;
    font-size: 1.66em;
}
main{ border: 3px solid red;
    height: 150%;
```



```

font-size: 1.2em;
background-image: url(../lesson/CSS.jpg);
background-size: cover;
}
nav{
border: 3px solid red;
height: 10%;
font-size: 1.1em;
}
footer{
border: 3px solid red;
height: 5%;
}

```

4.3 项目的运行和测试

项目的运行和测试通过 PC 端和手机端，本项目阶段性文件上传至 GitHub 网站，移动端可扫描二维码查看。



图 4-1. PC 端显示



图 4-2. 手机端显示

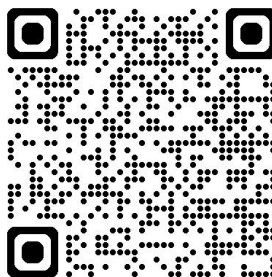


图 4-3

5. 响应式设计（宽屏和窄屏）

5.1 分析和设计

Web 响应式设计是一种网页设计方法，旨在使网站能够适应不同设备和屏幕尺寸的显示需求，它通过一系列技术和策略，确保网站能够在任何设备上都能提供良好的用户体验。这不仅提高了网站的可访问性，也增强了用户满意度和互动性，随着移动设备的普及，掌握响应式设计的最佳实践对于任何前端开发者来说都是非常重要的。在本程序 js 中，通过 `UI.appWidth` 和 `UI.appHeight` 变量中，使得获取窗口的初始宽度和高度，在 `UI.appWidth` 变量中设置窗口宽度大于或等于 600 像素，如若不是则设置为窗口的实际宽度，可以较好的适应窄屏环境问题，在本程序中设置 `setBodyStyle` 和 `setAidStyle` 函数分别用于设置 body 和键盘响应区域的宽度和高度，在键盘响应区域的窗口宽度小于 1000 像素时，显示属性设置为 none，当处于小屏则隐藏。为了实现响应式设计还要注意鼠标事件，设置全局变量 `mouse`，在程序 `bookface` 区域移动鼠标将会显示其坐标位置。

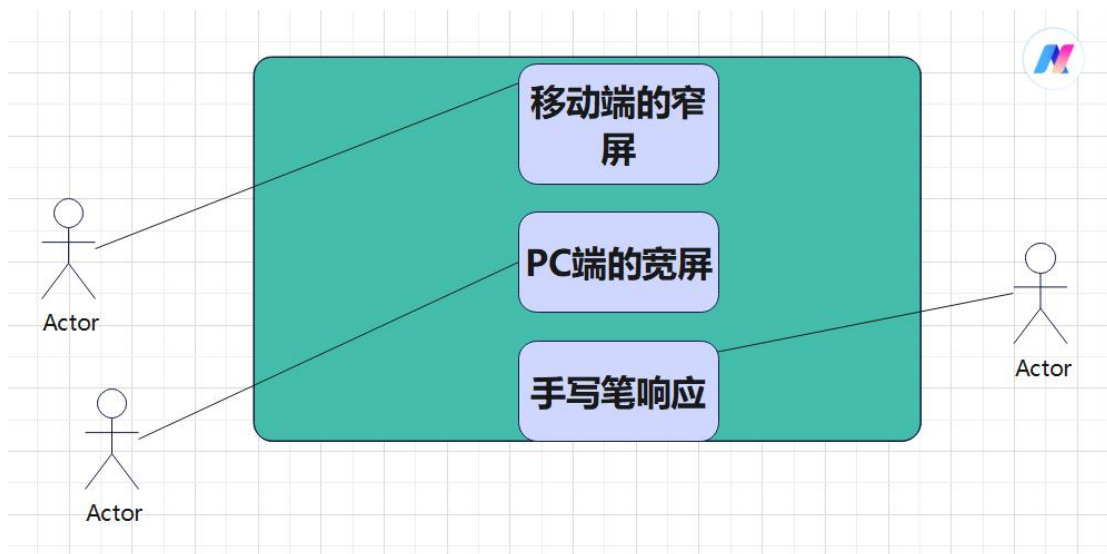


图 5-1. 宽屏窄屏用例图

5.2 项目的实现和编程

一. HTML 响应式设计编写

```
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){
    $("aid").style.display=' none' ;
}
$("aid").style.width=window.innerWidth-UI.appWidth-30+' px' ;
$("aid").style.height= UI.appHeight-62+' px' ;
//尝试对鼠标设计 UI 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+")";
});
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 $

```

二 CSS 编写

```

*{
    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;
}
#aid{
  position: absolute;
  border: 3px solid blue;
  top:0px;
  left:600px;
}
#bookface{
  width: 80%;
  height: 80%;
  border:1px solid red;
  background-color: blanchedalmond;
  margin:auto;
}
```

5.3. 项目的运行和测试

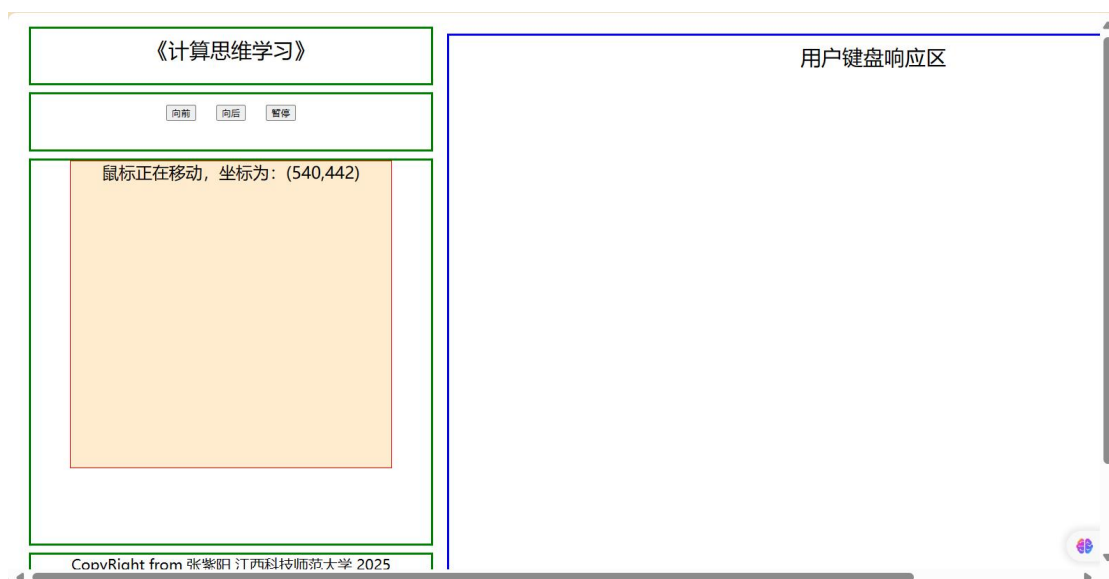


图 5-2. PC 端 body 与键盘响应

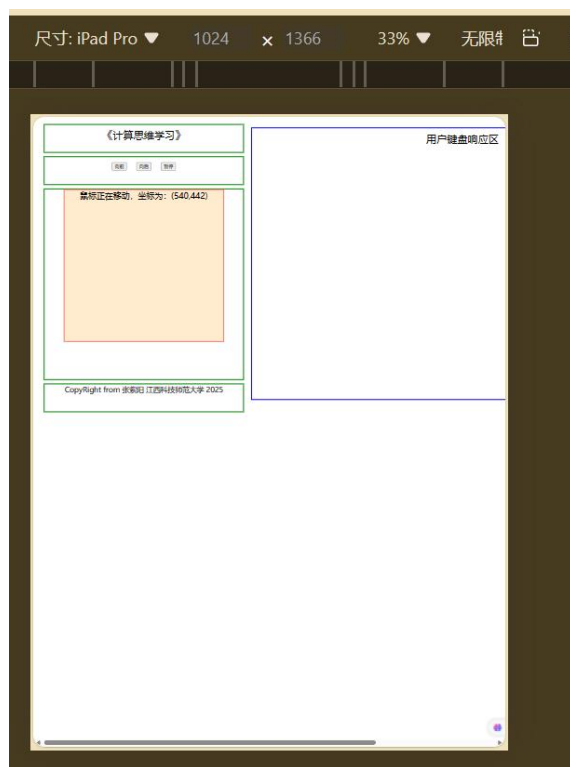


图 5-3.手机端 body 与键盘响应

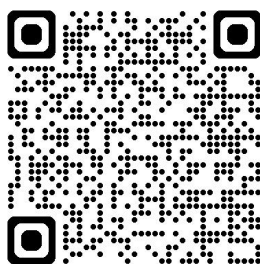


图 5-4

6. UI 设计中鼠标模型的控制

6.1 分析和设计

UI 设计中的鼠标模型分析和控制是一个复杂但至关重要的方面。鼠标作为用户与计算机交互的主要工具，其控制方式直接影响用户体验和界面的可用性，此程序设置模型 mouse 来模拟鼠标的操作数据，并根据鼠标模型和用户操作设计 UI 反馈程序，首先 isDown 属性标记鼠标是否按下，x 与 y 记录鼠标的坐标位置，而 deltaX 则追踪鼠标在水平方向上的移动距离，通过 addEventListener 方法，代码标记了四个鼠标事件：mousedown、mousemove、mouseup 和 mouseout，当鼠标按下时 mousedown 触发事件，记录鼠标的当前位置，当鼠标在区域内移动时触发 mousemove 事件，位置随鼠标移动而移动，当区域内鼠标施行拖拽时，触发 mouseup 事件检测是否超过区域值，mouseout 事件用于在鼠标离开元素时，恢复元素到其初始位置，此程序还增加了底层函数\$，用于快速抓取页面上的对象在代码中反复使用。



图 6-1 用例图

6.2 项目的实现和编程

一. HTML 鼠标模型编写

```
var mouse={};
mouse.isDown= false;
mouse.x= 0;
mouse.y= 0;
mouse.deltaX=0;
$("bookface").addEventListener("mousedown",function(ev){
    mouse.isDown=true;
    mouse.x= ev.pageX;
    mouse.y= ev.pageY;
    console.log("mouseDown at x: "+"("+mouse.x +"," +mouse.y +")" );
```

```

    $("bookface").textContent=    "    鼠 标 按 下 ，    坐 标  :
    "+"("+mouse.x+", "+mouse.y+")";
});
$("bookface").addEventListener("mouseup",function(ev){
    mouse.isDown=false;
    $("bookface").textContent= "鼠标松开!";
    if(Math.abs(mouse.deltaX) > 100){
        $("bookface").textContent += "，这是有效拖动！" ；
    }else{
        $("bookface").textContent += " 本次算无效拖动！" ；
        $("bookface").style.left = '7%' ；
    }
});
$("bookface").addEventListener("mouseout",function(ev){
    ev.preventDefault();
    mouse.isDown=false;
    $("bookface").textContent= "鼠标松开!";
    if(Math.abs(mouse.deltaX) > 100){
        $("bookface").textContent += " 这次是有效拖动！" ；
    }else{
        $("bookface").textContent += " 本次算无效拖动！" ；
        $("bookface").style.left = '7%' ；
    }
});
$("bookface").addEventListener("mousemove",function(ev){
    ev.preventDefault();
    if (mouse.isDown){
        console.log("mouse isDown and moving");
        mouse.deltaX = parseInt( ev.pageX - mouse.x );
        $("bookface").textContent= "正在拖动鼠标, 距离: "+ mouse.deltaX+"px 。
";
        $('bookface').style.left = mouse.deltaX + 'px' ；
    }
});
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 $

```

二. CSS 编写

```

*{
    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% ;
}

```

6.3. 项目的运行和测试



图 6-2 PC 端鼠标事件响应



图 6-3. PC 端鼠标事件响应

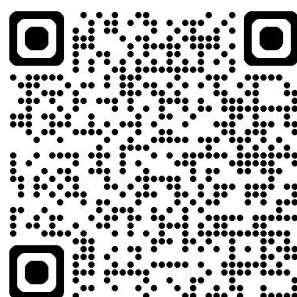


图 6-4

7. 通用的 UI 设计，用一套代码同时为触屏和鼠标建模

7.1 分析和设计

在开发中实现一套代码同时为触屏和鼠标建模的通用 UI 设计可以与跨平台 UI 框架通过抽象和封装底层平台的细节使得一套代码来构建适应不同平台的用户界面，再通过 CSS、JavaScript 根据设备动态 Dom 元素设置高度、宽度以及位置，定义一个 pointer 对象实现鼠标以及触屏的互动，在 pointer 对象中添加三个属性如 isDom、x 以及 deltaX，他们分别表示是否触发鼠标和触屏、x 代表鼠标以及触屏的横坐标而 deltaX 表示鼠标或触屏最后移动的位置与 x 之间的差值，为了处理交互事件，定义三个方法 handleBegin、handleMove、handleEnd，他们分别表示处理交互的开始、交互的移动以及交互的结束，<body>元素添加 6 个事件分别监视着鼠标的开始、移动和结束以及触屏的开始、移动和结束。通过上述操作实现鼠标触屏的交互，提高了开发效率确保应用能够在不同设备上的兼容性和统一性。

7.2 项目的实现和编写

一. HTML 触屏和鼠标建模编写

```

var Pointer = {};
Pointer.isDown= false;
Pointer.x = 0;
Pointer.deltaX =0;
{ //Code Block begin
let handleBegin = function(ev){
  Pointer.isDown=true;
  //console.log(ev.touches);
  if(ev.touches){
    Pointer.x = ev.touches[0].pageX ;
    Pointer.y = ev.touches[0].pageY ;
    console.log("Touch begin : "+"("+Pointer.x +"," +Pointer.y +")" ) ;
    $("bookface").textContent= " 触 屏 事 件 开 始 , 坐 标 :
"+"("+Pointer.x+","+Pointer.y+")";
  }else{
    Pointer.x= ev.pageX;
    Pointer.y= ev.pageY;
    console.log("PointerDown at x: "+"("+Pointer.x +"," +Pointer.y
+" )" ) ;
    $("bookface").textContent= " 鼠 标 按 下 , 坐 标 :
"+"("+Pointer.x+","+Pointer.y+")";
  }
};
let handleEnd = function(ev){
  Pointer.isDown=false;
  if(ev.touches){
    $("bookface").textContent= "触屏事件结束!";
    if(Math.abs(Pointer.deltaX) > 100){
      $("bookface").textContent += "，这是有效触屏滑动！" ;
    }else{
      $("bookface").textContent += " 本次算无效触屏滑动！" ;
      $("bookface").style.left = '7%' ;
    }
  }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' ;
        }
    }
};

```

二. CSS 编写

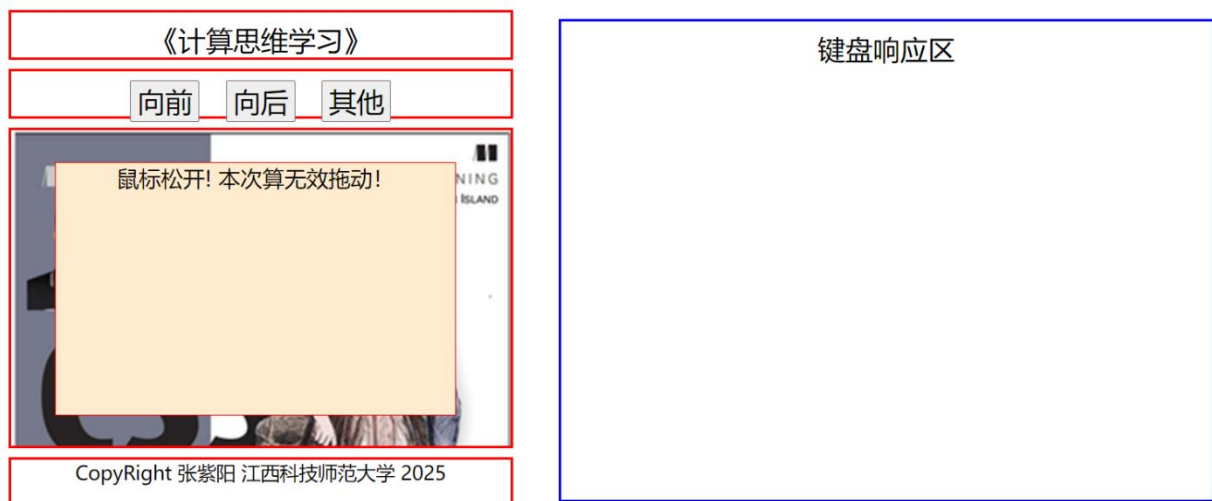
```

*{
    margin: 10px;
    text-align: center;
}
header{
    border: 3px solid red;
    height: 10%;
    font-size: 1em;
}
nav{
    border: 3px solid red;
    height: 10%;
}
main{
    border: 3px solid red;
    height: 70%;
    font-size: 0.8em;
    position: relative;
    background-image: url(../lesson/CSS.jpg);
    background-size:cover;
}
#box{
    position: absolute;

```

```
    right: 0;
    width: 100px;
}
footer{
    border: 3px solid gred;
    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% ;
}
footer{
    border: 3px solid red;
    height: 50px;
}
```

7.3 项目的运行和测试



7-1 PC 端

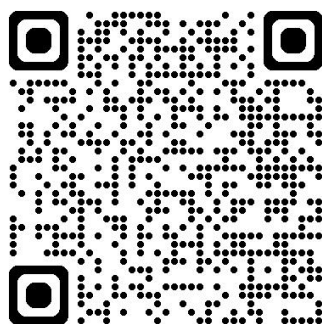


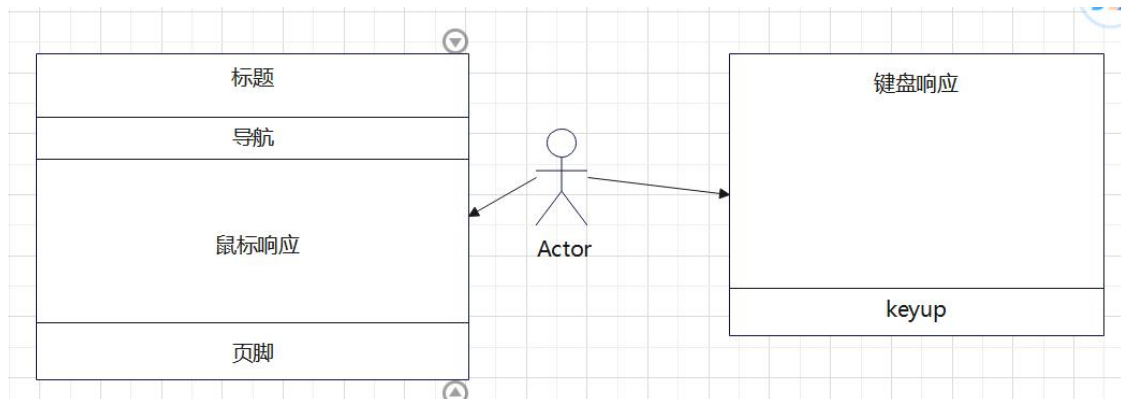
图 6-2

8. UI 的个性化键盘控制——巧妙应用 keydown 和 keyup 键盘底层事件

8.1 分析和设计

在 UI 设计中，个性化键盘控制是一个重要的功能，它可以提升用户体验和交互效率。通过巧妙应用 keydown 和 keyup 这两个键盘底层事件，我们可以实现对用户输入行为的精确控制，keydown 事件发生在用户按下键盘上的某个键时，而 keyup 事件则在用户释放该键时触发。这两个事件是处理键盘输入的基础，通过监听这些事件，我们可以捕获到用户的每一次按键操作，从而进行相应的处理。ev.preventDefault() 方法阻止 keypress 事件发生，实现了自定义的交互，在程序中两个底层事件分别在按键按下以及按键释放时触发，方便提供输入。在页面的键盘响应区域 keydown 事件可以输出所按键的值，keyup 事件则显示在区域内显示出弹起信息，总之，通过合理地使用 keydown 和 keyup 事件，我们可以在

UI 设计中实现丰富而灵活的个性化键盘控制，从而提升用户的操作体验和效率。在设计这些功能时，理解这些事件的触发条件和特性是非常重要的，这将帮助我们更好地满足用户的需求并优化用户界面。



8-1. 用例图

8.2 项目的实现和编写

一. HTML 的 keydown 和 keyup 事件编写

// 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";
    $("#outputText").textContent += key;
});
```

二. CSS 编写

```
* {
    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/CSS.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;
}
#outputText {
    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%;  
}
```

8.3 项目的运行和测试

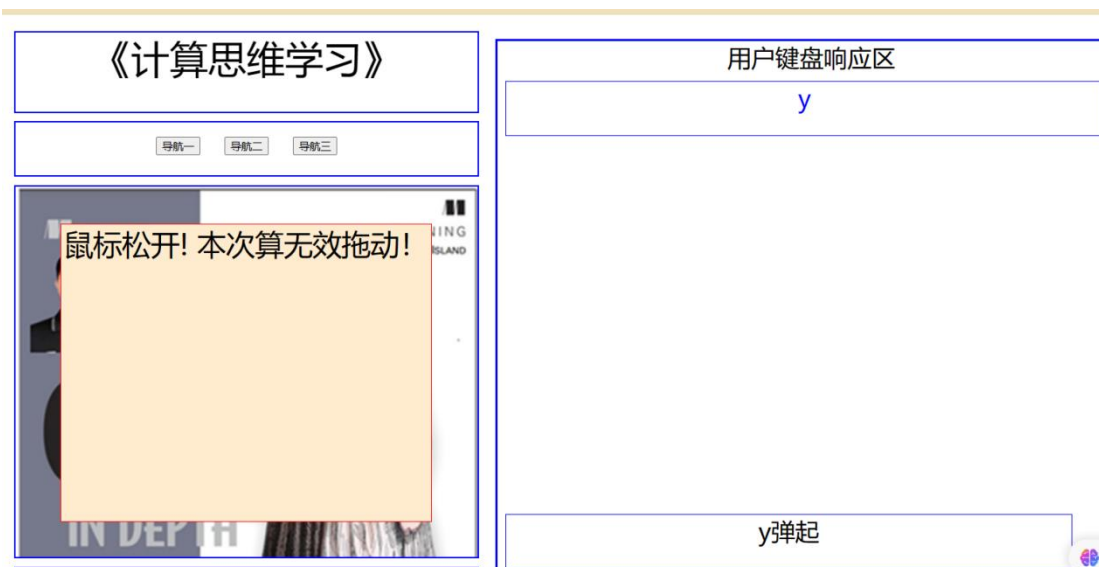


图 8-2. PC 端页面响应

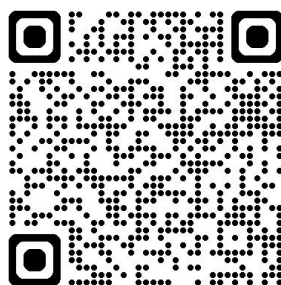
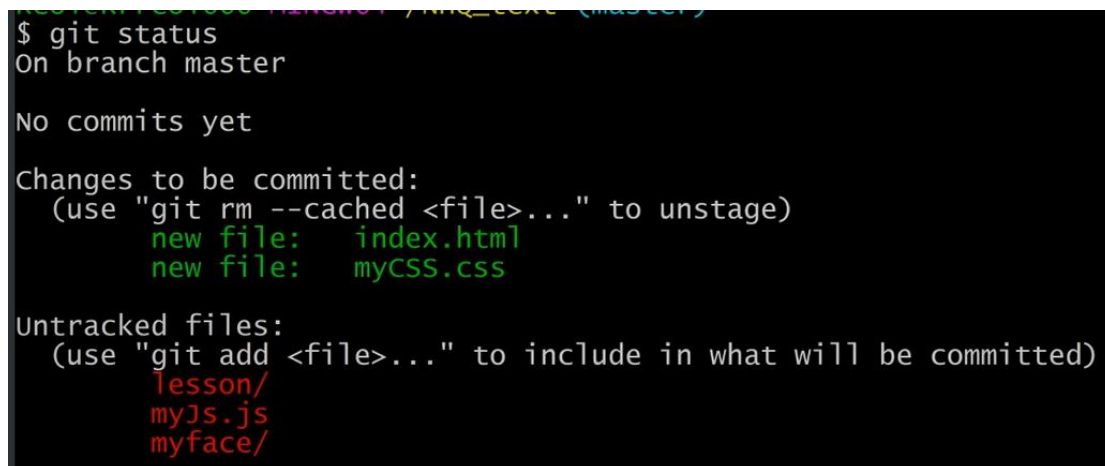


图 8-3

9. 用 git 工具开展代码版本管理

Git 是一个开源的分布式版本控制系统，由 Linus Torvalds 开发，最初用于管理 Linux 内核的开发。它是目前世界上最先进的版本控制系统之一，Git 的主要特点包括其高效性和灵活性，它支持快速的上下文切换和多重提交，这使得开发者可以在不同的分支之间自由切换，以尝试不同的功能或修复错误，而不会影响到主分支。当我们谈到命令行时，实际上指的是 shell。shell 是一个接收键盘命令并将其传递给操作系统执行的程序。几乎所有的 Linux 发行版都提供一个来自 GNU 项目的 shell 程序，名为 bash。这个名字是 bourne-again shell

的缩写，指的是 bash 是 sh 的增强版替代程序，而 sh 是由 Steve Bourne 编写的最初的 Unix shell 程序^[3]。此外，Git 采用的是分布式版本库的方式，这意味着每个用户都有一个完整的代码仓库副本，这与传统的集中式版本控制系统（如 SVN）形成对比，后者将版本控制信息集中存储在中央服务器上。GitHub 是基于 Git 的一个平台，它提供了代码托管、版本控制以及协作功能，是全球最大的开源项目托管平台。通过 GitHub，开发者可以更方便地管理他们的项目，同时也能利用 Git 的强大功能来提高开发效率和质量，本项目打开 gitbash 输入命令 `git init` 初始化仓库，通过命令 `git status` 来查看仓库初始化成功信息，`git add` 命令是 Git 版本控制系统中的一个基本且重要的命令。它的主要功能是将工作区中新建、修改或删除的文件内容添加到暂存区。完成文件修改后再通过命令 `git commit` 将暂存区中的变化提交到仓库区，本仓库包含多个 html 文件，通过 `git commit -m` 命令执行每次修改代码后的开发日志编辑。当本地仓库准备好时，可以将其推送到远程仓库 GitHub，这可以通过 `git push origin master` 命令完成。如果是从远程仓库拉取最新更改到本地，可以使用 `git pull` 命令，完成共享与协作。



```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   myCSS.css

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        lesson/
        myJs.js
        myface/
```

图 9-1. git status 命令示例

代码执行

```
$ cd/
$ mkdir webUI
$ cd webUI
$ git init
$ git config user.name 江科师大张紫阳
$ git config user.name 1482277817@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命令提交

```
$ git add index.html myCss.css
$ git commit -m '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功能设计。lesson文件夹和myface文件夹分别放项目的各种图片与资源文件。'
[master (root-commit) 60ea966] 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功能设计。lesson文件夹和myface文件夹分别放项目的各种图片与资源文件
```

图 9-2. git commit 命令示例

进入本地 webUI 项目文件后通过下面命令把本地代码仓库与远程建立密钥链接

```
$ echo "webUI"应用远程 http 服务器设置">>README.md
$ git init
$ git add README.md
$ git commit -m "第一次上传代码仓库到 GitHub 平台"
$ git branch -M main
$ git remote add origin
http://github.com/sunny-vit/zhangziyang.github.io
$ git push -u origin main
```

使用 window 平台，GitHub 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，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。

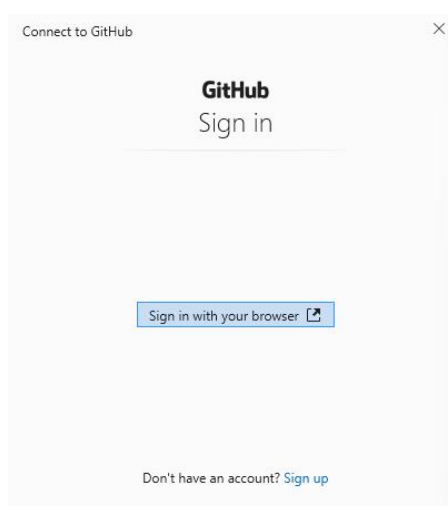
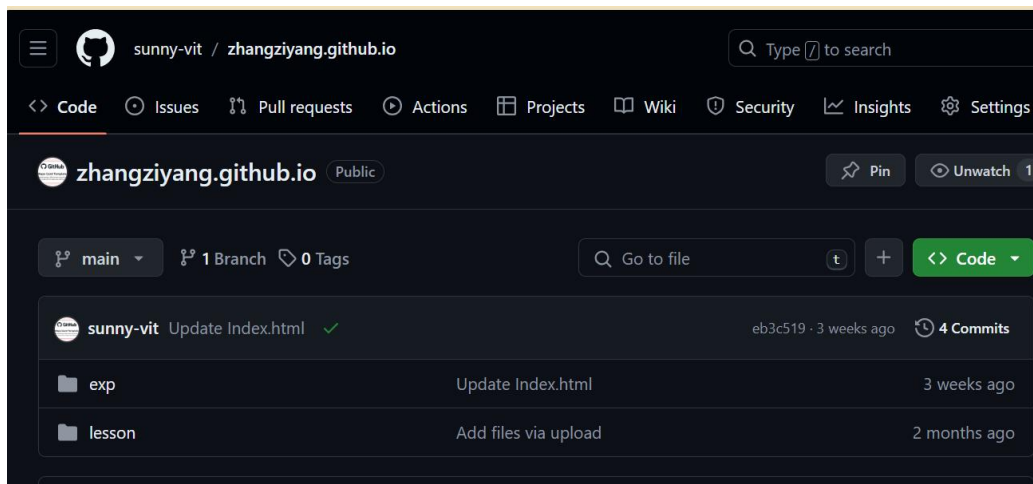


图 9-3. 授权界面



9-4. GitHub 个人仓库

10. 编写高质量代码

编写高质量代码是软件开发中的一个重要方面，它涉及到多个层面，包括函数定义、代码块的组织、模型设计以及全局变量的使用等。函数是实现程序功能的基本单位，编写高质量的函数需要注意函数的命名、参数和代码编写，好的函数应该具有清晰的命名，参数明确，并且代码易于理解和维护，函数式编程强调函数的纯粹性和不变性，这有助于提高代码的可读性和可维护性。合理的代码块组织可以提高代码的可读性和可维护性，例如，使用适当的注释和文档来解释代码的目的和行为，以及进行单元测试和集成测试，可以帮助开发者更好地理解和维护代码，在软件开发中，领域模型、设计原则和设计模式是提升代码质量的重要方法，通过分析需求，选择合适的领域模型和设计模式，可以有效地解决复杂问题，提高代码的结构性和可扩展性，全局变量虽然可以减少变量的个数，但过多使用会带来一系列问题，如耦合度高、难以理解和维护等。因此，应尽量减少全局变量的使用，可以通过封装状态到类或结构体中，或者使用命名空间来管理全局变量。

参考文献

- [1]John Dean,PhD. Web programming with HTML5,CSS,and JavaScript[M].Jones & Bartlett Learning,LLC. 2019
- [2]Marijn Haverbeke.Eloquent JavaScript 3rd edition. No Starch Press,Inc,2019.
- [3]William Shotts.The Linux Command Line, 2nd Edition [M]. No Starch Press, Inc,245 8th Street, San Francisco,CA 94103,2019