
江西科技师范大学

课程设计（论文）

题目（中文）：基于 HTML 的 WebAppUI 设计开发

（外文）：WebAppUI design and developmentbased on HTML

院（系）：元宇宙产业学院

专 业：计算机科学与技术

学生姓名：袁飞宇

学 号：20213586

指导教师：李健宏

2024 年 6 月 19 日

目录

基于 Web 客户端技术的个性化 UI 的设计和编程.....	2
(Customized UI design and Programming based on Web client technology)	2
1. 前言	2
1.1 毕设任务分析	3
1.2 研学计划	3
1.3 研究方法	3
2. 技术总结和文献综述	5
2.1 Web 平台和客户端技术概述	5
2.2 项目的增量式迭代开发模式	5
3. 内容设计概要	6
3.1 分析和设计	6
3.2 项目的实现和编程	7
3.3 项目的运行和测试	8
3.4 项目的代码提交和版本管理	8
4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计	9
5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI	12
6. 个性化 UI 设计中对鼠标交互的设计开发	14
7. 对触屏和鼠标的通用交互操作的设计开发	16
8. UI 的个性化键盘交互控制的设计开发	21
9. 谈谈本项目中的高质量代码	23
10. 用 gitBash 工具管理项目的代码仓库和 http 服务器	25
10.1 经典 Bash 工具介绍	25
10.2 通过 gitHub 平台实现本项目的全球域名	26
10.3 创建一个空的远程代码仓库	26
10.4 设置本地仓库和远程代码仓库的链接	27
参考文献:	31
写作指导:	31

基于 Web 客户端技术的个性化 UI 的设计和编程

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

科师大元宇宙产业学院 2021 级 XXX

【摘要】

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，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（操作）。通过集成上述技术，结合本科的相关课程知识，本项目实现了一个个性化的用户界面（UI: uer interface），该用户界面可以动态适用于 PC 端和移动端设备，以响应式技术为支撑做到了最佳适配用户屏幕，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，用工程思想管理项目，该项目使用了软件工程的增量式增强的开发模式，一共做了 6 次 ADIT (A:D: I: T) 开发，逐步实现了整个 UI 应用编写。为了与互联网上的同行合作，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路径，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。

1. 前言

在软件工程的视角中的开发过程包括四个阶段：分析、设计、实施和测试，本文下面简称为 ADIT, 此开发过程有两种经典模型：瀑布模型 (The waterfall model) 和增量模型 (The incremental model)。

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。这对于我们大多数普通开发者是非常不方便的，比如在实施过程中，开发者发现设计段存在问题，开发者是不能纠正设计阶段的问题，只能按设计法案实施。我们大多数小微开发者，在软件的开发中总是在不断地优化设计，不断地重构代码，持续改进程序的功能和代码质量，这种方式与增量开发模式非常匹配，因此在本项目的开发中我们采用了增量模型的开发模式。在本项目中我们一共做了六次项目的开发迭代，

1.1 毕设任务分析

计算机科学与技术专业的毕业设计是对学生所学知识和技能的综合检验。

通常有软件开发、系统设计、算法、等多个领域。例如，开发一款具有特定功能的应用软件，如在线学习平台或企业资源管理系统；设计并实现高效的算法，如图像识别算法或数据加密算法；研究人工智能在某个领域的应用，如自然语言处理在智能客服中的应用等。

在设计过程中，学生需要运用多种技术和工具。包括编程语言（如 Java、Python 等）、数据库管理系统（如 MySQL、Oracle 等）、开发框架（如 Spring、Django 等），以及版本控制工具（如 Git）等。

毕业设计还注重学生对系统性能、安全性、可扩展性等方面的考虑。学生需要进行需求分析，明确系统的功能和性能要求；进行详细设计，包括架构设计、模块划分和接口定义；然后通过编码实现，并进行严格的测试和调试，以确保系统的稳定性和可靠性。

最后，毕业设计的成果展示和论文撰写也非常重要。学生需要清晰地阐述设计思路、技术实现、遇到的问题及解决方案等，以展示自己的专业能力和研究成果。总之，计算机科学与技术专业的毕业设计是一个全面、系统且具有挑战性的项目，对于学生未来的职业发展和学术深造都具有重要意义。

。

1.2 研学计划

研学计划是一种有组织、有目的、有规划的学习活动方案。

它通常结合旅行和实地考察的方式，旨在让参与者在亲身经历和实践中获取知识、提升能力、培养素养。一份完整的研学计划包括明确的主题，例如历史文化、自然科学、艺术人文等。它会确定具体的研学目标，如增强对某一领域的了解、培养团队协作能力、提高观察和分析问题的能力等。在行程安排方面，会详细规划访问的地点、活动内容以及时间分配。还会涉及交通、住宿、餐饮等后勤保障的安排。同时，研学计划会明确指导教师或导师的职责，制定学习成果的评估方式，以及为可能出现的问题准备应对措施和安全预案。总之，研学计划是一个综合性的方案，旨在为参与者提供丰富、有意义且安全的学习体验。

1.3 研究方法

1. 文献综述：广泛查阅国内外相关文献，了解该领域的研究现状和发展趋势。
2. 理论研究：深入学习和分析相关的理论知识和算法，为实际研究提供理论支持。
3. 实验研究：使用公开的医疗影像数据集进行实验，通过编程实现算法和模型，并进行训练和测试。
4. 数据分析：对实验结果进行数据分析，评估模型的性能和准确性，并找出影

响模型性能的关键因素。

5. 对比研究：将所提出的方法与现有先进方法进行对比，分析其优势和不足。

2. 技术总结和文献综述

2.1 Web 平台和客户端技术概述

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

Web 应用的程序设计体系由三大语言有机组成：**HTML**，**CSS**，**JavaScript**。这三大语言的组合也体现了人类社会化大生产分工的智慧，可以看作用三套相对独立体系实现了对一个信息系统的描述和控制，可以总结为：**HTML** 用来描述结构（**Structure**）、**CSS** 用来描述外表（**presentation**）、**Javascript** 用来描述行为（**Behavior**）^[3]；这也可以用经典的 MVC 设计模式来理解 Web 平台架构的三大基石，**Model** 可以理解为 **HTML** 标记语言建模，**View** 可以理解为用 **CSS** 语言来实现外观，**Controller** 则可理解为用 **JavaScript** 结合前面二个层次，实现了在微观和功能层面的代码控制。

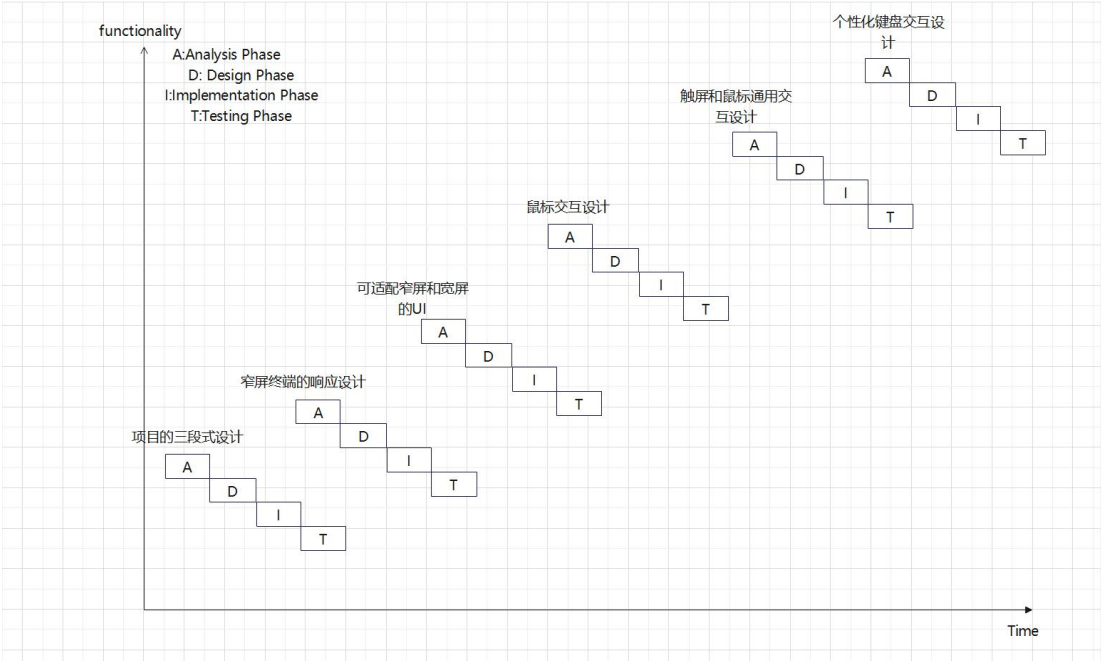
2.2 项目的增量式迭代开发模式

本项目作为一个本科专业学生毕业设计的软件作品，与单一用途的程序相比较为复杂，本项目所涉及的手写代码量远超过简单一二个数量级以上，从分析问题的到初步尝试写代码也不是能在几天内能落实的，可以说本项目是一个系统工程，因此需要从软件工程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开发过程管理模式有两种经典模型：瀑布模型（The waterfall model）和增量式迭代模型（The incremental model）。而任何开发模式则都必须同样经历四个阶段：分析（Analysis）、设计（Design）、实施（Implementation）、测试（test）。

瀑布模型需要专业团队完美的配合，从分析、设计到实施，最后到测试，任何阶段的开始必须基于上一阶段的完美结束。而这对于我们大多数普通开发者是不太现实的，作为小微开发者由于身兼数职，其实无法 1 次就能完美完成任何阶段的工作，比如在实施过程中，开发者会发现前面的设计存在问题，则必须在下一次迭代项目时改良设计。在当今开源的软件开发环境中，开发者在软件的开发中总

是在不断地优化设计、重构代码，持续改进程序的功能和代码质量。因此在本项目的开发中，也采用了增量模型的开发模式^[5]。本项目中我一共做了六次项目的开发迭代，如下图所示：



3. 内容设计概要

3.1 分析和设计

这一步是项目的初次开发，本项目最初使用人们习惯的“三段论”式简洁方式开展内容设计，首先用一个标题性信息展示 logo 或文字标题，吸引用户的注意力，迅速表达主题；然后展现主要区域，也就是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的重点；最后则是足部的附加信息，用来显示一些用户可能关心的细节变化。如图 4-1 用例图所示：

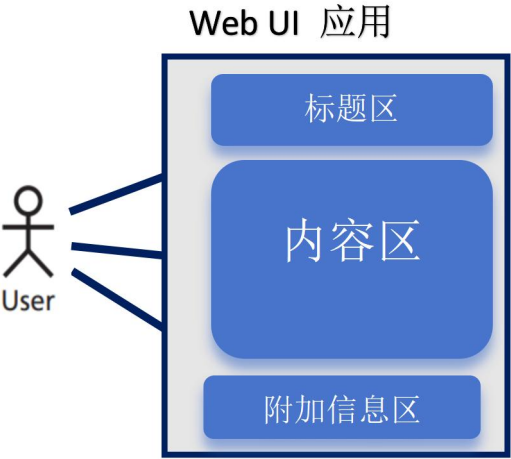


图 4-1 用例图

3.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
    《 我的毕设题目 》
</header>
<main>
    我的主题内容： ‘读好书、练思维、勤编程’ @masterLijh 计算思维系列课程
</main>
<footer>
    Copyright XXX 江西科技师范大学 2024-2025
</footer>
```

二、CSS 代码编写如下：

```
*{
    margin: 10px;
    text-align: center;
    font-size:30px ;
}
header{
    border: 2px solid blue;
    height: 200px;
}

main{
    border: 2px solid blue;
    height: 400px;
}
footer{
    border: 2px solid blue;
    height: 100px;
}
a{
    display: inline-block ;
    padding:10px ;
    color: white;
    background-color: blue;
    text-decoration: none ;
}
```


3.3 项目的运行和测试

项目的运行和测试至少要通过二类终端，本文此处仅给出 PC 端用 Chrome 浏览器打开项目的结果，如下图 4-2 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 4-3 的二维码，运行测试本项目的第一次开发的阶段性效果。



图 4-2 PC 端运行效果图



图 4-3 移动端二维码

3.4 项目的代码提交和版本管理

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

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /  
$ mkdir webUI  
$ cd webUI  
$ git init  
$ git config user.name 江科师大李健宏  
$ git config user.email marsterlijh@jxstnu.edu.cn  
$ touch index.html myCss.css
```

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

```
$ git add index.html myCss.css
```

```
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
```

成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m 项目第一版：“三段论”式的内容设计概要开发
[master (root-commit) 32de024] 项目第一版：“三段论”式的内容设计概要开发
2 files changed, 46 insertions(+)
create mode 100644 index.html
create mode 100644 myCss.css
```

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，

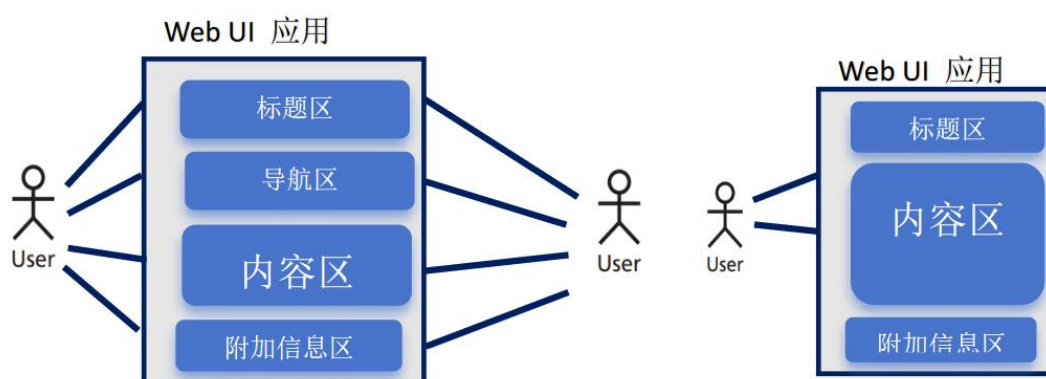
```
$ git log
```

gitbash 反馈代码的仓库日志如下所示：

```
$ git log
commit 32de0243f71ec4396f54175d83c805c999d473ca (HEAD -> master)
Author: 江科师大李健宏 <marsterlijh@jxstnu.edu.cn>
Date: Wed May 29 15:26:56 2024 +0800

    项目第一版：“三段论”式的内容设计概要开发
```

4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计



分析移动互联时代的多样化屏幕的需求。

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

实现代码

用汉语言来描述我们是如何实现的，与上一阶段比较，本阶段初次引入了 em 和 %，这是 CSS 语言中比较高阶的语法，可以有效地实现我们的响应式设计。如代码块 4-1 所示：

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

```

header{
  border: 2px solid blue;
  height: 15%;
  font-size: 1.66em;

}
main{
  border: 2px solid blue;
  height: 70%;
  font-size: 1.2em;

}
nav{
  border: 2px solid blue;
  height: 10%;
  }
nav button{
  font-size: 1.1em;
}
footer{
  border: 2px solid blue;
  height: 5%;
}
</style>

```

代码块 4-1

用汉语言来描述我们是如何实现的：与上一阶段比较，本阶段首次使用了 JavaScript，首先创建了一个 UI 对象，然后把系统的宽度和高度记录在 UI 对象中，又计算了默认字体的大小，最后再利用动态 CSS，实现了软件界面的全屏设置。如代码块 4-2 所示：

```

<script>
  var UI = {};
  UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;
  UI.appHeight = window.innerHeight;
  const LETTERS = 22 ;
  const baseFont = UI.appWidth / LETTERS;

  //通过更改 body 对象的字体大小，这个属性能够遗传其子子孙孙
  document.body.style.fontSize = baseFont + "px";
  //通过把 body 对象的宽度和高度设置为设备/屏幕的宽度和高度，实现全屏。
  //通过 CSS 对子对象百分比（纵向）的配合，从而实现响应式设计的目标。
  document.body.style.width = UI.appWidth - 2*baseFont + "px" ;
  document.body.style.height = UI.appHeight - 4*baseFont + "px";
</script>

```

代码块 4-2

《 我的毕设题目 》

我的主题内容： ‘读好书、练思维、勤编程’ @masterLijh 计算思维系列课程

Copyright XXX 江西科技师范大学 2024-2025

其他章节：

1.21.31.41.51.6

《 我的毕设题目 》

我的主题内容： ‘读好书、练思维、勤编程’ @masterLijh 计算思维系列课程

Copyright XXX 江西科技师范大学 2024-2025

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI



A: 宽屏窄屏 UI 设计需求分析: 需考虑不同屏幕尺寸对布局的影响。宽屏可展示更多内容, 元素排列可更宽松, 注重信息丰富度与整体视觉效果。窄屏则空间有限, 要突出关键信息, 简洁明了, 操作便捷。还需确保在两种屏幕下, 界面的可读性、交互性与美观度都能良好呈现, 保持风格一致, 适应不同用户场景, 以提升用户体验, 满足多样化使用需求。

D: 通过运用灵活的布局、媒体查询和动态调整元素大小等手段, 确保界面在不同屏幕尺寸下都能呈现出最佳效果。它能根据屏幕宽度自动调整页面结构和元素排列, 使得无论是宽屏的高清显示器还是窄屏的移动设备, 用户都能获得一致且舒适的交互体验。开发过程中, 精心构建响应式框架, 对各种组件进行巧妙设计, 实现了界面在不同屏幕条件下的无缝适配, 满足现代用户对多设备使用的需求, 提升了产品的可用性和用户满意度。

I: 实施阶段, 将设计转换为代码实现, 如下图

```
var UI = {};  
    UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth ;  
    UI.appHeight = window.innerHeight;  
    const LETTERS = 22 ;  
    const baseFont = UI.appWidth / LETTERS;  
  
    //通过更改 body 对象的字体大小, 这个属性能够遗传其子子孙孙  
    document.body.style.fontSize = baseFont + "px";  
    //通过把 body 对象的宽度和高度设置为设备/屏幕的宽度和高度, 实现全屏。  
    //通过 CSS 对子对象百分比(纵向)的配合, 从而实现响应式设计的目标。  
    document.body.style.width = UI.appWidth - 2*baseFont + "px" ;  
    document.body.style.height = UI.appHeight - 4*baseFont + "px";
```

T: 测试阶段从以下几个方面

字体大小测试:

在宽屏模式下, 字体大小显示适中, 能够清晰阅读, 无模糊、重叠或过小难以辨认的情况。

切换至窄屏时，字体自动进行了合理调整，依然保持了良好的可读性，未出现因屏幕变窄而导致字体严重变形或看不清的问题。

文本清晰测试：

无论是宽屏还是窄屏状态，文本的清晰度都很高，边缘锐利，没有出现锯齿或模糊现象。

不同字号的文本在各种屏幕尺寸下均能准确呈现，未发现有文本模糊影响理解的区域。

空间布局合理性测试：

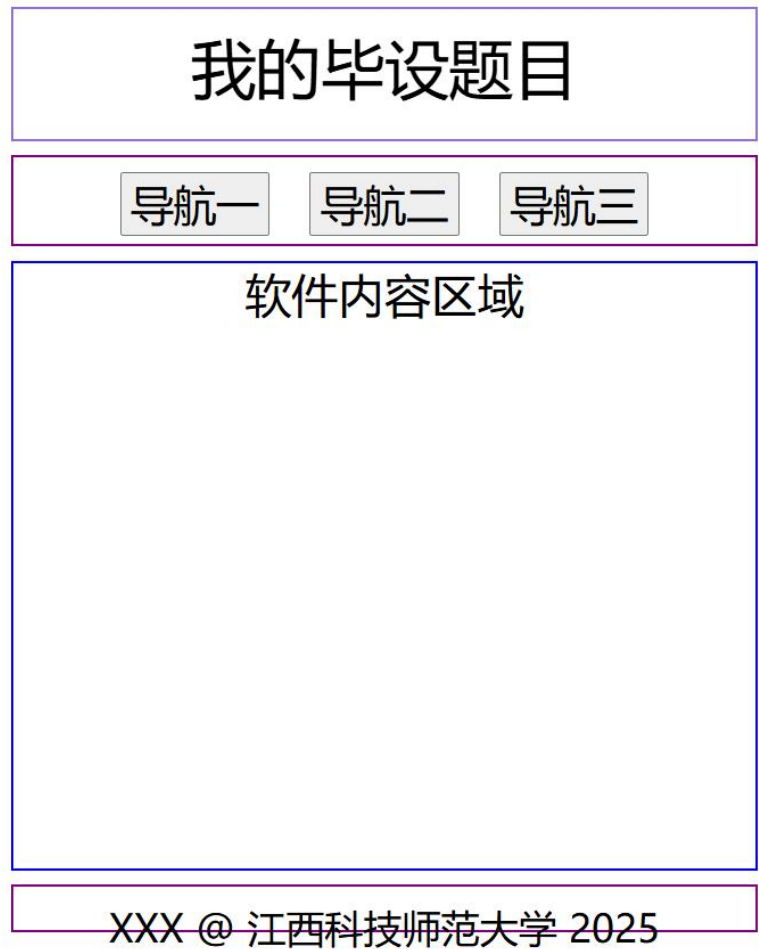
在宽屏上，各元素之间的空间布局合理，有足够的留白，视觉上不拥挤。

窄屏状态下，元素自动重新排列，空间利用得当，没有出现元素重叠、遮挡或布局混乱的情况。

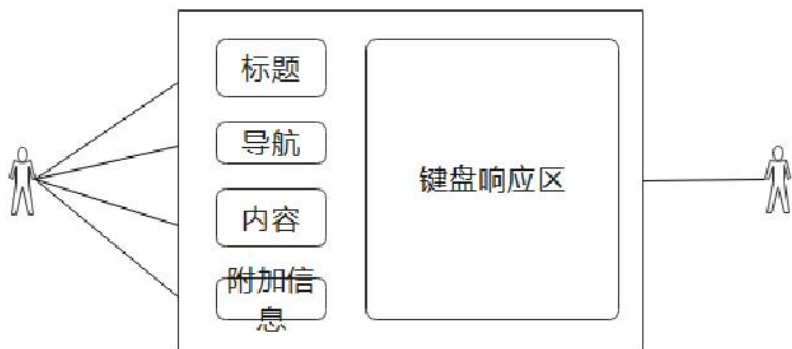
过渡平滑测试：

从宽屏到窄屏以及反之的切换过程中，界面过渡非常平滑，没有出现卡顿、闪烁或突然的变形。

所有元素的缩放和位置调整都很自然流畅，提供了连贯一致的用户体验。如下图



6. 个性化 UI 设计中对鼠标交互的设计开发



图表 4-1 响应用例图

A: 设计中，对鼠标交互的设计开发需求至关重要。需考虑鼠标悬停时元素的视觉反馈，如颜色变化、阴影效果等，以增强提示性。点击动作应具备明确响应，反馈及时且直观。滚动操作要流畅自然，与页面内容联动良好。同时要兼顾不同鼠标设备的兼容性，确保在各种情况下都能实现精准交互，从而提升用户体验，让用户与界面的互动更加便捷、愉悦。

D: 个性化 UI 设计，当鼠标悬停在界面元素上时，会出现微妙而吸引人的视觉反馈，出现实时鼠标位置，及时更新鼠标位置，给予用户明确的提示。点击操作伴有清晰的触感反馈，增强交互的真实感。鼠标滚动时，页面的响应流畅自然，内容过渡顺滑。让用户清晰感知操作进程。在不同的场景下，为用户打造极具沉浸感和便捷性的交互体验，使整个 UI 系统更具活力与亲和力。

I: :实施阶段，将设计转换为代码实现，如下图

```
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";
});
$("#body").addEventListener("keypress",function(ev) {
    let k = ev.key;
    let c = ev.keyCode;
    $("#keyboard").textContent = "您的按键："+k+"， "+ "字符编码："+c;
```



```

});

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 $

```

T: 测试阶段至关重要，用于验证功能是否按预期工作。

测试应该包括：

检查鼠标按下、移动、移出事件是否被正确捕获和处理。

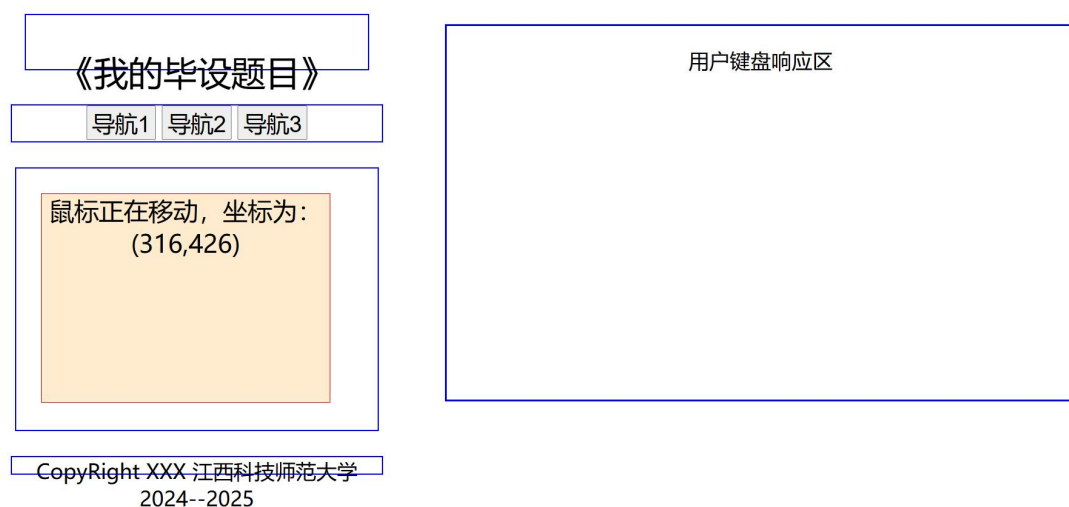
测试键盘按键事件是否准确识别按键和键码。

检查自定义函数在不同元素 ID 和选择器下的表现。

确保所有事件处理逻辑在各种浏览器环境下均能稳定运行。

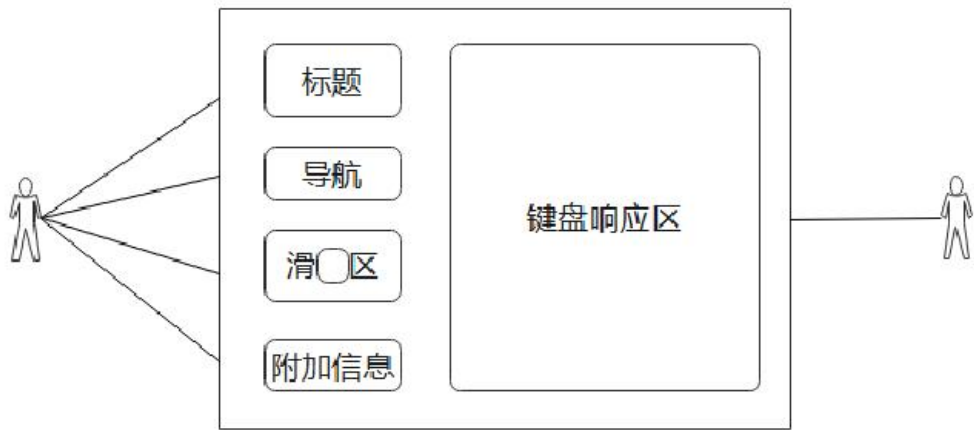
通过全面测试，我们可以确保代码的质量和稳定性，满足设计阶段设定的需求。

以下为实现图：



7. 对触屏和鼠标的通用交互操作的设计开发

阐述用一套代码逻辑同时为触屏和鼠标建立对象模型



图表 6-6 鼠标触屏交互用例图

A: 对于触屏和鼠标的通用交互操作设计开发，需考虑两者特性。要确保界面元素大小适宜，方便触屏点击和鼠标精准定位。手势操作与鼠标动作应对应合理，如滑动、缩放等。反馈机制需明确，触屏的触觉反馈和鼠标的视觉反馈都要清晰直观。同时，要保证交互的流畅性和响应速度，兼顾不同场景下的使用体验。使无论是触屏还是鼠标操作，用户都能轻松完成各种任务，提升可用性和便捷性。

D: 在通用交互操作设计开发中，针对触屏和鼠标进行了巧妙融合。对于触屏，设计了直观的手势操作，如轻触对应鼠标点击，滑动对应页面滚动或元素切换。长按可触发特定功能或弹出菜单。而对于鼠标，保持传统的点击、滚动等操作流畅性。同时，确保两种交互方式在功能响应上保持一致，让用户无论是使用触屏还是鼠标都能获得相似的体验。在界面元素的设计上，尺寸和布局适应触屏的直接触摸和鼠标的精准指向，兼顾两者的便利性。通过精心设计，实现了触屏和鼠标通用交互的和谐统一，提升用户体验的连贯性和便捷性。

I: 在交互操作设计开发中，实现触屏和鼠标的通用。对于点击操作，无论是触屏的直接触碰还是鼠标的左键点击，都能触发相同响应。滑动操作在触屏上通过手指滑动与鼠标滚轮滚动实现统一效果。元素的选中在触屏上通过点击并长按与鼠标的左键按住类似。同时，设计合理的反馈机制，让触屏的触感反馈

实施阶段，将设计转换为代码实现，如下图

```
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(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("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 $

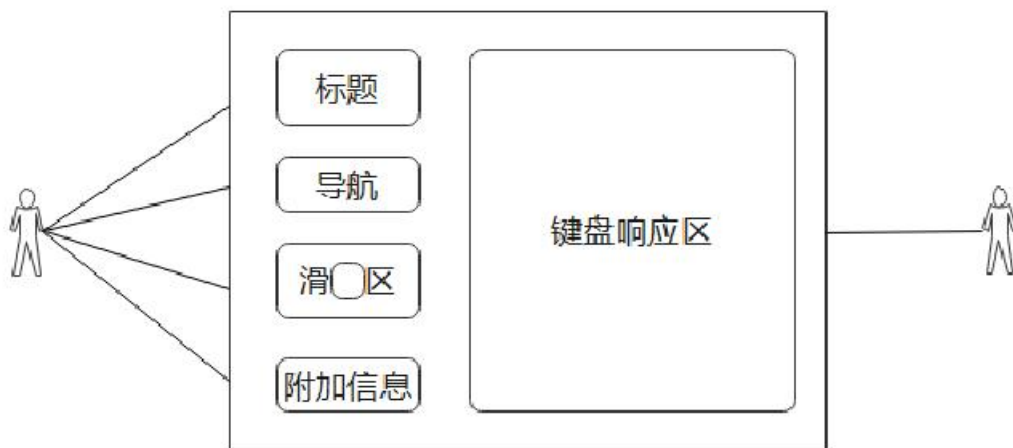
```

T: 在对触屏和鼠标的通用交互操作设计开发测试中，我们着重关注两者的兼容性和一致性。对于触屏操作，手势的识别精准且迅速，如滑动、缩放等动作能流畅地触发相应功能。而鼠标操作时，点击、拖动等动作也能达到同样效果，且反馈及时。在界面元素的选中与交互上，无论是触屏点击还是鼠标点击，都能准确响应。测试中还注重过渡效果的自然性，无论是触屏滑动切换还是鼠标滚轮滚动，页面转换都顺滑无卡顿。通过多场景、多方式的反复测试，确保了这套通用交互操作设计在触屏和鼠标环境下都能为用户提供优质、统一的体验，提升了可用性和便捷性。以下是效果图



8. UI 的个性化键盘交互控制的设计开发

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



图表 8-1 键盘交互控制用例图

:

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 ;
    }
}

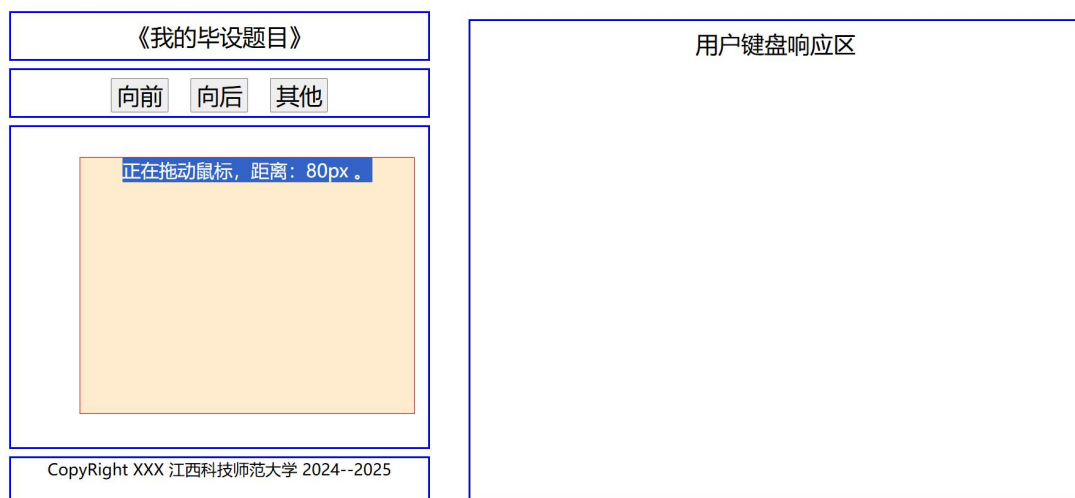
```

```

    }
    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)
});

```

以下是效果图



9. 谈谈本项目中的高质量代码

创建一个 **Pointer** 对象，践行 **MVC** 设计模式，设计一套代码同时对鼠标和触屏实现控制。面向对象思想，封装，抽象，局部变量，函数式编程，逻辑。


```

var Pointer = {};
Pointer.isDown= false;
Pointer.x = 0;
Pointer.deltaX =0;
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' ;
        }
    }
};

```

10. 用 gitBash 工具管理项目的代码仓库和 http 服务器

10.1 经典 Bash 工具介绍

Bash（Bourne Again Shell）是许多类 Unix 操作系统中常用的命令行 shell 环境，拥有众多实用工具。

首先是“grep”，用于在文件或输入流中搜索匹配指定模式的文本行。它能快速筛选出所需信息。

“awk”是一种强大的文本处理工具，能对文本进行复杂的格式化和数据提取操作。

“sed”可对文本进行非交互式的编辑，实现替换、删除、插入等多种修改。

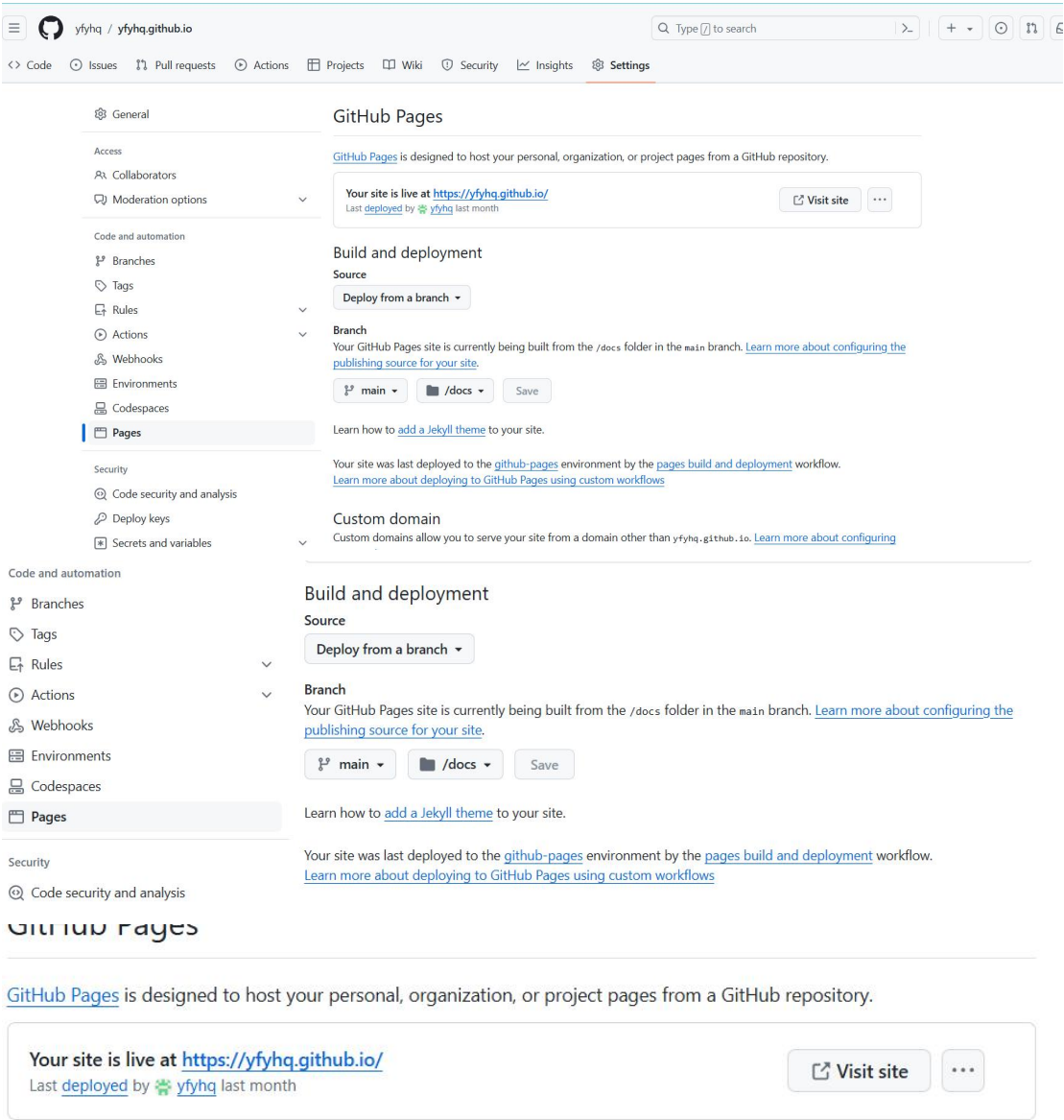
“find”用于在指定目录及其子目录中查找符合条件的文件。

“sort”能对文本行进行排序，可按照数字、字母等规则进行。

“uniq”用于去除连续重复的行，方便对数据进行去重处理。

这些经典的 `Bash` 工具，相互配合使用，能够高效地处理各种文本和文件操作任务，极大地提高了系统管理员和开发者在命令行环境下的工作效率。

10.2 通过 gitHub 平台实现本项目的全球域名




10.3 创建一个空的远程代码仓库

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 masterLijh

 /

Repository name *

✓ userName.github.io is available.

Great repository names are short and memorable. Need inspiration? How about [expert-rotary-phone](#) ?

Create repository

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

10.4 设置本地仓库和远程代码仓库的链接

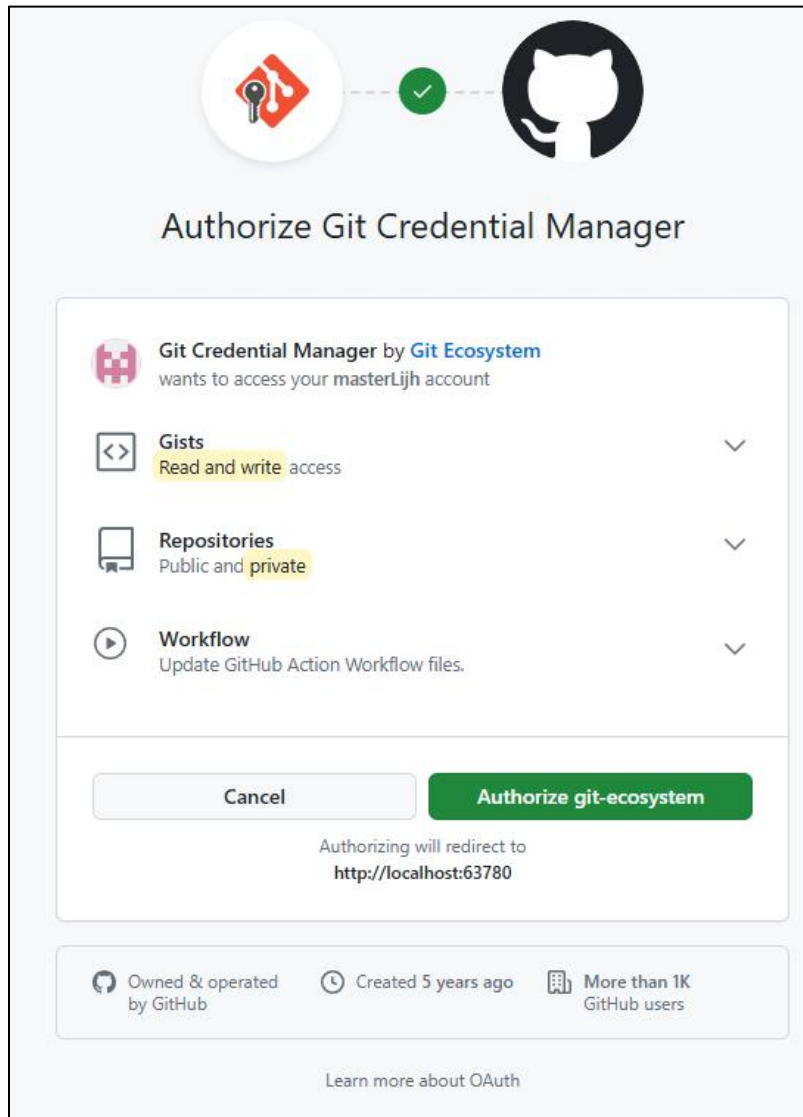
进入本地 webUI 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接

```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md
$ git init
$ git add README.md
$ git commit -m "这是我第一次把代码仓库上传至 gitHub 平台"
$ git branch -M main
$ git remote add origin
    https://github.com/masterLijh/userName.github.io.git
$ git push -u origin main
```

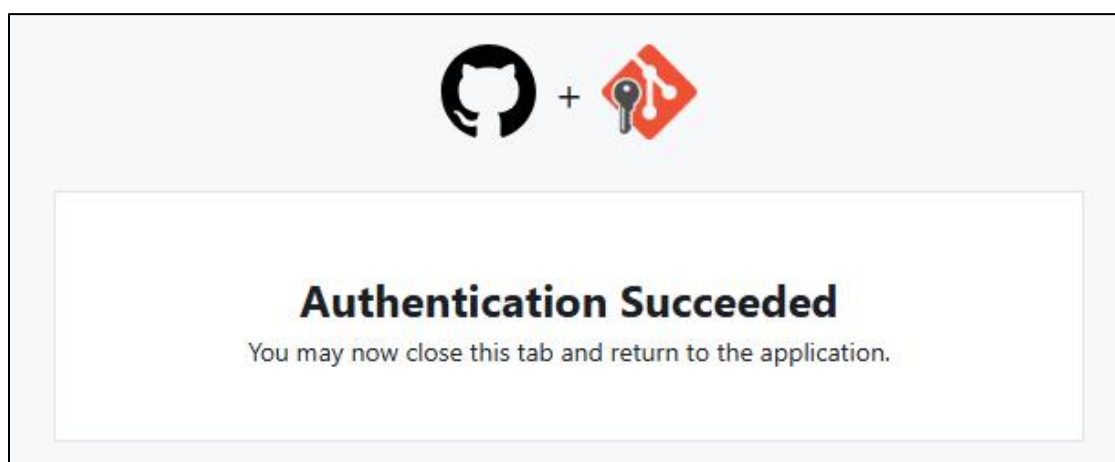
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



再次确认授权 `gitBash` 拥有访问改动远程代码的权限，如下图所示：

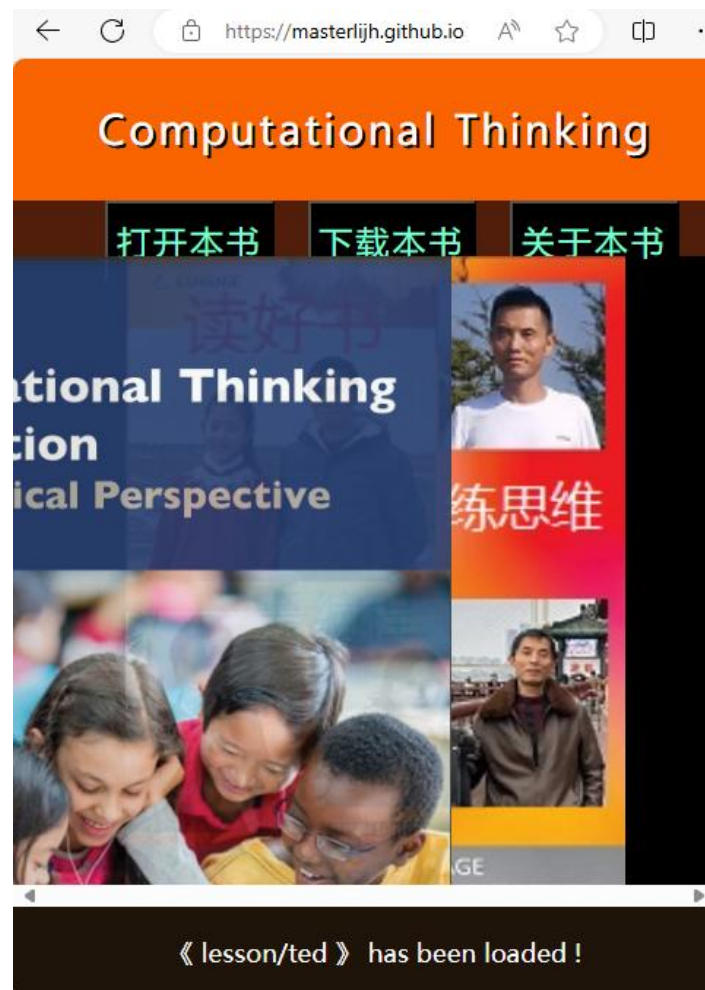


最后，GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。



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

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



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

写作指导:

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。