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：黎嘉洛

学　　号：20213642

指导教师：李建宏

2024 年　6 月 15　日

目录

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 黎嘉洛

【摘要】

摘要：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。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。

关键词：UI；github；gitbash；软件开发；程序设计

1. 前言

随着互联网的普及和科技的飞速发展，Web UI 设计已经成为现代软件开发中不可或缺的一环。优秀的 Web UI 设计不仅能够提升用户体验，还能为企业带来更高的效益。本文将介绍 Web UI 设计的基本概念、设计原则以及相关技术和工具，深入了解并掌握 Web UI 设计。

Web UI 设计，即网页用户界面设计，是指通过一系列的设计元素和布局，使用户在使用网页时能够高效、便捷地完成任务。一个成功的 Web UI 设计应具备以下几个特点： 清晰的视觉层次：通过合理的布局和排版，让用户能够快速地找到自己需要的信息和功能；一致性：保持界面元素的一致性，包括颜色、字体、图标等，有助于用户快速熟悉和掌握使用方法； 简洁性：避免过多的装饰

和冗余信息，让用户专注于核心功能和内容；可访问性：考虑到不同用户的需求，提供易于操作和理解的界面，让更多人能够使用；为了实现优秀的 Web UI 设计，设计师需要掌握一定的设计原则和技巧，如：对比与重复、对齐与亲密性、空白与间距等。此外，设计师还需要了解一些常用的设计工具，如 Sketch、Adobe XD、Figma 等，以便更高效地完成设计工作。随着技术的发展，Web UI 设计也在不断演变。从最初的单页应用到如今的前端框架和组件 React、Vue、Angular 等，以及 Bootstrap、Ant Design 等 UI 组件库，都为设计师和开发者提供了丰富的资源和便捷的开发方式。Web UI 设计是现代软件开发中至关重要的一环。通过学习和实践，设计师可以不断提升自己的设计能力，为用户带来更好的使用体验。希望本文能够为您提供 Web UI 设计的入门指导和参考，让您在设计之路上一帆风顺。

1.1 毕设任务分析

毕业设计任务分析是一个系统的过程，需要从多个角度进行全面的考量。

首先要明确毕业设计的主题和目标，这通常由导师提出或与学生共同商定。理解任务的核心要求是后续所有工作的基础。对相关领域的现状、发展趋势、潜在用户需求等进行调研，以便更好地定位设计方向。根据任务理解和需求调研的结果，设定具体的研究目标和预期成果。评估项目的可行性，包括技术可行性、时间可行性、资源可行性等。

其次，需要有一定的知识储备，列出完成毕业设计所需的所有知识点，包括已有知识和需要新学习的知识。确定学习资源的获取途径，如书籍、在线课程、研讨会、实验室资源等。制定详细的学习时间表，包括每个知识点的学习时间和复习时间。建立机制定期检查学习进度，确保按计划进行。

最后，根据毕业设计的特点，选择合适的研究方法，如实验研究、理论研究、案例研究等。确定采用的技术路线，包括开发工具、实验设备、软件平台等。如果研究需要数据支持，确定数据来源、收集方法和分析工具。如果涉及实验，设计实验方案，包括实验步骤、控制变量、实验组与对照组的设置等。

通过上述分析，学生可以更清晰地理解毕业设计任务的要求，制定出合理的学习和研究计划，并采用恰当的方法来完成设计任务。这样的分析有助于提高毕

业设计的质量和效率，同时也能够为学生未来的学术或职业生涯打下坚实的基础。

1.2 研学计划

研学计划是学生在教师的指导下，通过实地考察、研究性学习等方式，对某一领域或主题进行深入学习的一种计划。

首先，对研学的任务有一定理解，明确研学主题和目标，理解研学活动的意义和目的。对研学需求进行调研，对相关领域进行调研，了解研学主题的背景和现状。需要设定目标，根据任务理解和需求调研，设定研学活动的具体目标和预期成果。

其次，需要有一定的知识储备，列出完成研学活动所需的所有知识点，包括已有知识和需要新学习的知识。利用好学习资源，确定学习资源的获取途径，如书籍、在线课程、研讨会等。对时间进行规划，制定详细的学习时间表，包括每个知识点的学习时间和复习时间。对进度进行跟踪，建立机制定期检查学习进度，确保按计划进行。

最后，根据研学主题的特点，选择合适的研究方法，如实验研究、调查研究、案例研究等。确定采用的技术路线，包括开发工具、实验设备、软件平台等。确定数据来源、收集方法和分析工具，确保数据的可靠性和有效性。如果涉及实验，设计实验方案，包括实验步骤、控制变量、实验组与对照组的设置等。

1.3 研究方法

研究方法是毕业设计任务中实现研究目标的关键部分。

首先，对研学的任务有一定理解，明确研学主题和目标，理解研学活动的意义和目的。对研学需求进行调研，对相关领域进行调研，了解研学主题的背景和现状。需要设定目标，根据任务理解和需求调研，设定研学活动的具体目标和预期成果。

其次，需要有一定的知识储备，列出完成研学活动所需的所有知识点，包括已有知识和需要新学习的知识。利用好学习资源，确定学习资源的获取途径，如书籍、在线课程、研讨会等。对时间进行规划，制定详细的学习时间表，包括每

个知识点的学习时间和复习时间。对进度进行跟踪，建立机制定期检查学习进度，确保按计划进行。

最后，根据研学主题的特点，选择合适的研究方法，如实验研究、调查研究、案例研究等。确定采用的技术路线，包括开发工具、实验设备、软件平台等。确定数据来源、收集方法和分析工具，确保数据的可靠性和有效性。如果涉及实验，设计实验方案，包括实验步骤、控制变量、实验组与对照组的设置等。

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。

2.1.1 历史

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。

2.1.2 万维网联盟

1994 年，在许多公司将越来越多的资源投入网络的敦促下，万维网联盟决定成立。蒂姆·伯纳斯-李爵士开始领导网络联盟团队的基本工作，以促进一个一致的架构，以适应快速发展的网络标准，用于构建网站、浏览器、设备，以体验网络所提供的一切。在创建万维网联盟时，蒂姆·伯纳斯-李爵士创建了一个同行社区。Web 技术已经发展得如此之快，以至于组建一个组织来协调 Web 标准变得至关重要。Tim 接受了麻省理工学院的邀请，他有与联盟打交道的经验，负

责托管 W3C。他从一开始就要求 W3C 具有全球性的足迹。

2.1.3 Web 平台与 Web 编程

让我们先简单介绍一下 Web，也就是万维网的缩写。大多数人说“Web”而不是“World Wide Web”，我们将遵循这一惯例。网络是文档的集合，称为网页，由世界各地的计算机用户共享(大部分)。不同类型的网页做不同的事情，但至少，它们都在电脑屏幕上显示内容。所谓“内容”，我们指的是文本、图片和用户输入机制，如文本框和按钮。[2]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。所有的工具都使用核心语言 HTML，所以几乎所有的 web 编程书籍都在某种程度上描述了 HTML。这本教科书涵盖了 HTML5, CSS 和 JavaScript，所有的深度。这三种技术被认为是客户端 web 编程的支柱。使用客户端 web 编程，所有网页计算都在最终用户的计算机(客户端计算机)上执行。[3]

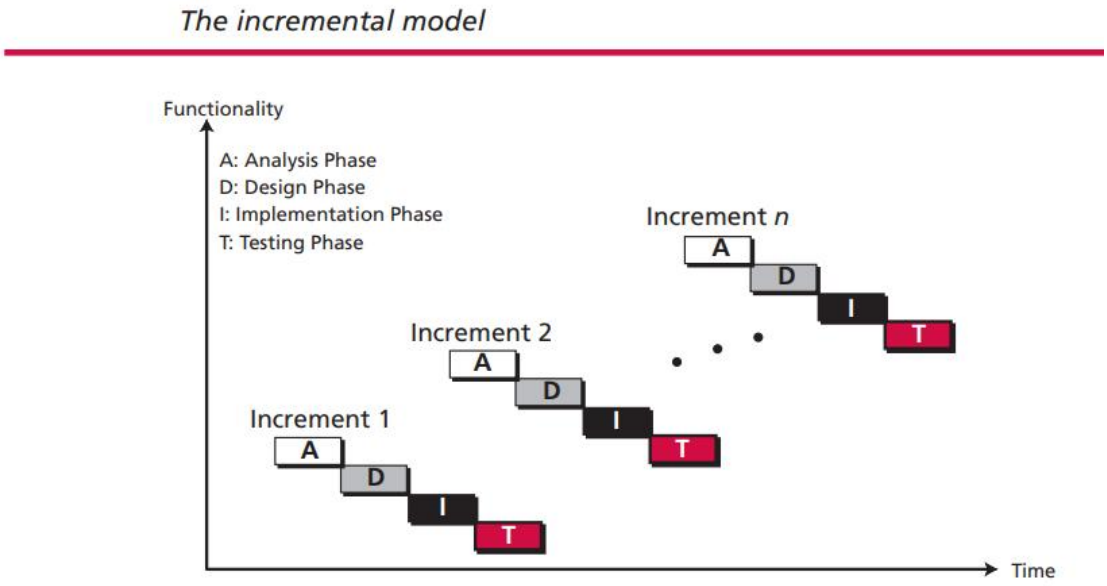
Web 应用的程序设计体系由三大语言有机组成：HTML, CSS, JavaScript。这三大语言的组合也体现了人类社会化大生产分工的智慧，可以看作用三套相对独立体系实现了对一个信息系统的描述和控制，可以总结为：HTML 用来描述结构（Structure）、CSS 用来描述外表（presentation）、Javascript 用来描述行为（Behavior）[3]；这也可以用经典的 MVC 设计模式来理解 Web 平台架构的三大基石，Model 可以理解为 HTML 标记语言建模，View 可以理解为用 CSS 语言来实现外观，Controller 则可理解为用 JavaScript 结合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

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 2-1 所示：



增量模型

在增量模型中，软件是按一系列步骤开发的。开发人员首先完成整个系统的简化版本。这个版本代表整个系统，但不包括细节。图中显示了增量模型概念。在第二个版本中，添加了更多的细节，而一些未完成，并再次测试系统。如果有问题，开发人员就会知道问题出在新功能上。在现有系统正常工作之前，它们不会添加更多的功能。这个过程一直持续到添加了所有需要的功能。[5]

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：

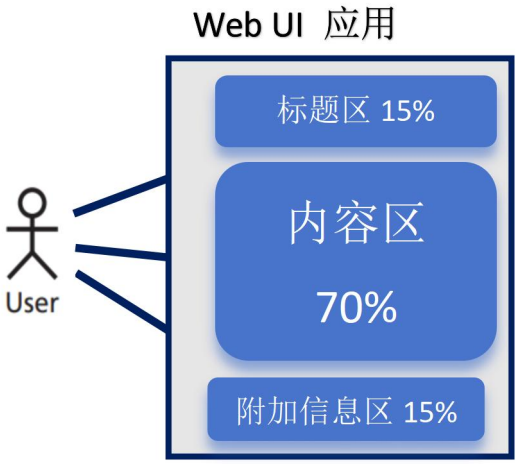


图 4-1 用例图

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。

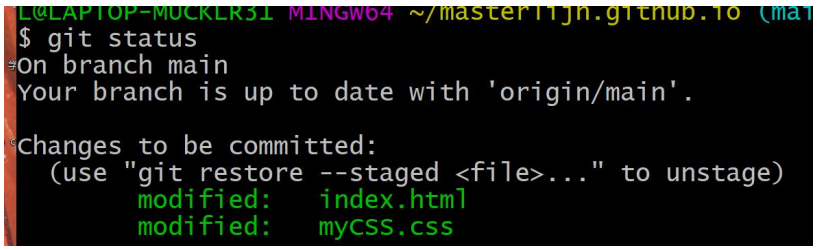
进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd /
$ mkdir webUI
$ cd webUI
$ git init
$ git config user.name 江科师大黎嘉洛
$ git config user.email 281404428@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 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：



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：

4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计

响应式设计——适应显示硬件

计算机所使用的显示硬件千差万别，显示器的大小和分辨率取决于成本。设计师们选择让网页给出一般的布局准则，并允许浏览器选择如何在给定的计算机上显示页面，而不是为每种类型的显示提供每个网页的版本。因此，一个网页不能提供很多细节。例如，网页的作者可以指定一组句子组成一个段落，但作者不能指定诸如一行的确切长度或是否缩进段落开头等细节。[1]

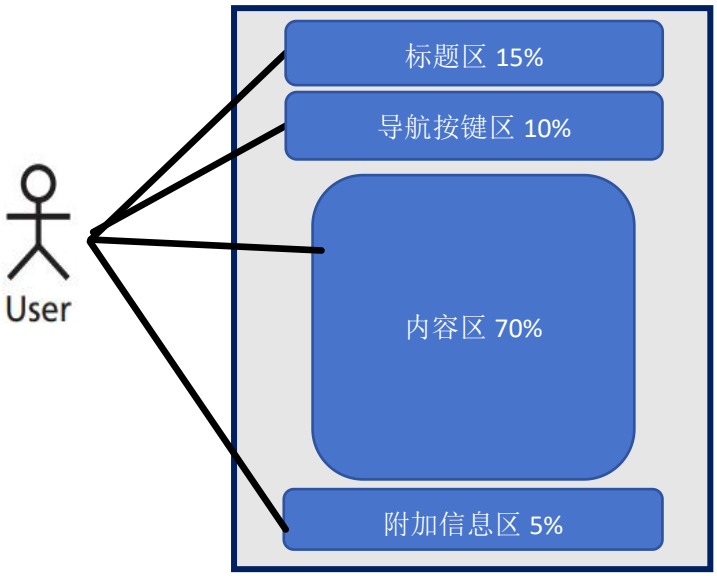
允许浏览器选择显示细节会产生一个有趣的结果:当通过两个浏览器或在硬件不同的两台计算机上浏览时，网页可能会显示不同的内容。如果一个屏幕比另一个屏幕宽，则可以显示的文本行的长度或图像的大小不同。重点是:网页给出了关于期望呈现的一般指导方针;浏览器在显示页面时选择详细信息。因此，同一网页在两台不同的计算机或不同的浏览器上显示时可能会略有不同。[1]

分析移动互联时代的多样化屏幕的需求。

4.1 分析与设计

本项目在上一次的“三段论”方式开展内容设计的基础上，用一个标题性信息展示 logo 或文字标题，吸引用户的注意力，接着在标题下方添加了一个按键区域，

按键区设计了三个导航按钮，可以为用户提供导航的便利，然后展现的主要区域是内容区，“内容为王”是项目必须坚守的理念，也是整个 UI 应用的主题，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。如下图所示：



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。

4.2 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

4.3 项目的运行和测试

本文通过 PC 平台，用 chrome 浏览器打开本阶段上传至 http 服务器的代码，网址（https://20216342.github.io/1.2.html），效果图如图 4-3 所示。还可以通过手机平台，用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址（https://20216342.github.io/1.2.html），效果图如图 4-4 所示。表明目前项目运行正常，达到了项目初步的目标。移动端可以使用图 4-5 的二维码，查看代码运行效果。

代码的运行效果图：

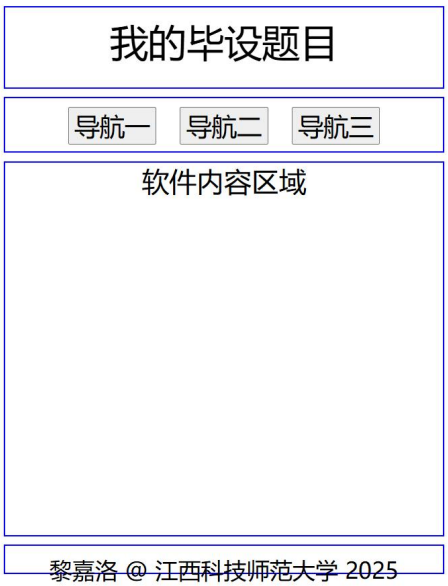


图 4-3 PC 平台运行效果图

图 4-4 手机平台效果运行图

图 4-5 移动端二维码入口

4.4 代码提交

本阶段完成的代码存放在 1index.html 和 1myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 4-6 所示：

```
$ git add 1myCSS.css

L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   1index.html
        new file:   1myCSS.css
```

图 4-6 本次代码提交的文件列表

命令行如下：

```
git add 1index.html  1myCSS.css
git commit -m 移动互联网时代的响应式设计和窄屏代码实现
```

本次代码提交成功，结果如图 4-7 所示：

```
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git config user.name 黎嘉洛

L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git config user.email 2814044280@qq.com

L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git commit -m移动互联网时代的响应式设计和窄屏代码实现
[main de7c402] 移动互联网时代的响应式设计和窄屏代码实现
 3 files changed, 91 insertions(+)
 create mode 100644 1index.html
 create mode 100644 1myCSS.css
 create mode 100644 index.1
```

图 4-7 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：

```
commit de7c40271a9b953695df609b7f50e21f5c0e4ece
Author: 黎嘉洛 <2814044280@qq.com>
Date:   Sat Jun 15 14:34:31 2024 +0800

    移动互联网时代的响应式设计和窄屏代码实现
```

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.1 移动互联网用户终端

移动互联网时代的特点之一就是用户终端的多样性。用户可以使用各种设备访问互联网，包括手机、电脑等智能设备。这些设备的屏幕尺寸、分辨率、输入方式等有所不同，因此网页和应用的设计需要能够适应这些多样化的设备。响应式设计（Responsive Design）是一种网页设计和开发的方法，旨在使网页能够响应用户的行为和环境，包括屏幕尺寸、平台和方向。这种设计方式确保了网页在不同设备上都能提供良好的用户体验，无论用户使用的是智能手机、平板电脑、笔记本电脑还是台式电脑。

CSS 语言和 JavaScript 语言实现响应式设计，加入了#aid 和#bookface，首次使用了 position 用法。如码块 5-1 所示：

```
*{
    text-align: center;
    box-sizing: border-box ;
  }
    header,main,div#bookface,nav,footer{
     margin:1em;
    }
    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%;
  }
  body{
   position:relative ;
  }
#aid{
  position: absolute;
  border: 3px solid blue;
  top: 0.5em;
  left: 600px;
}
#bookface{
  width: 80%;
  height: 80%;
  border:1px solid red;
  background-color: blanchedalmond;
  margin:auto;
}
```

码块 5-1

用 JavaScript 实现了软件界面的全屏设置。如代码块 5-2 所示：

```
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 - 8*baseFont + "px";

if(window.innerWidth < 900){
    $("aid").style.display='none';
  }
  $("aid").style.width=window.innerWidth - UI.appWidth - 2*baseFont + 'px';
  $("aid").style.height= document.body.clientHeight + '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+")";
});
$("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 $
```

码块 5-2

5.2 项目的运行和测试

本文通过 PC 平台，用 chrome 浏览器打开本阶段上传至 http 服务器的代码，网址（https://20216342.github.io/1.3.html），效果图如图 5-3 所示。还可以通过手机平台，用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址（https://20216342.github.io/1.3.html），效果图如图 5-4 所示。表明目前项目运行正常，达到了项目初步的目标。移动端可以使用图 5-5 的二维码，查看代码运行效果。

代码的运行效果图：

图 5-3 PC 端运行效果图

图 5-4 手机端运行效果图

图 5-5 移动端二维码入口

5.3 代码提交

本阶段完成的代码存放在 2index.html 和 2myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 5-6 所示：

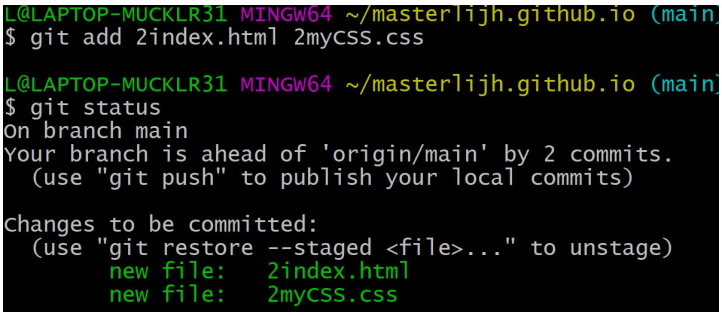


图 5-6 本次代码提交的文件列表

命令行如下：

```
git add 2index.html  2myCSS.css
git commit -m 适用移动互联网时代的响应式设计
```

本次代码提交成功，结果如图 5-7 所示：

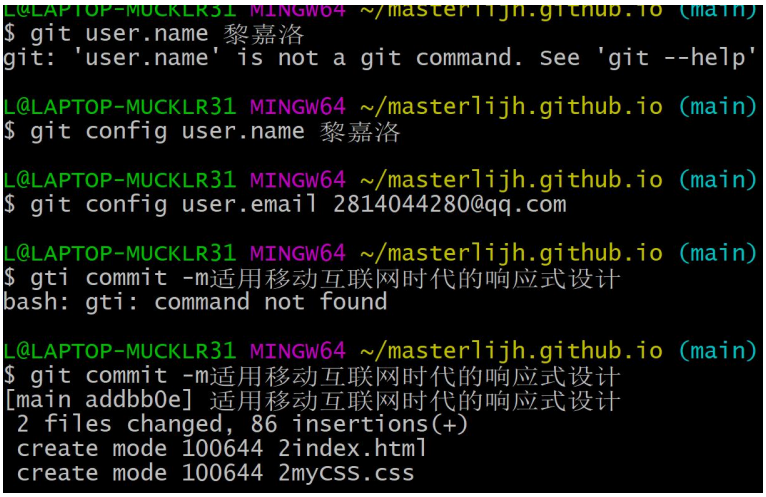
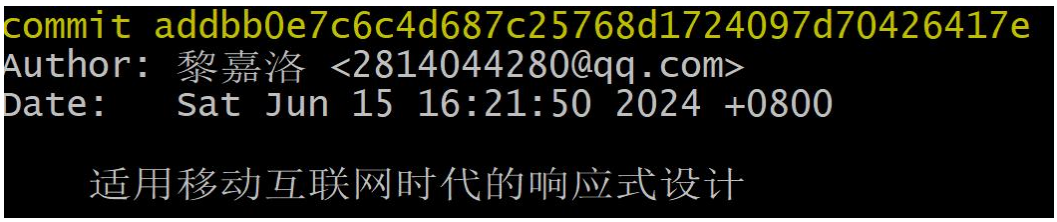


图 5-7 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：



6. 个性化 UI 设计中对鼠标交互的设计开发

6.1 个性化 UI 设计

在个性化 UI 设计中，为了确保用户体验的一致性，需要为触屏和鼠标交互设计一套共同的代码逻辑。这意味着无论是通过触摸屏还是鼠标，用户都能以相似的方式与 UI 进行交互。以下是如何使用一套代码逻辑同时为触屏和鼠标建立对象模型的方法，利用 CSS 语言和 JavaScript 语言完成设计，代码块如图 6-1 所示：

```
*{
      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% ;
  }
```

CSS 块码 6-1

用 JavaScript 实现了软件界面的全屏设置。如代码块 6-2 所示：

```
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 $
```

码块 6-2

6.2 项目的运行和测试

本文通过 PC 平台，用 chrome 浏览器打开本阶段上传至 http 服务器的代码，网址（https://20216342.github.io/1.4.html），效果图如图 6-3 所示。还可以通过手机平台，用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址（https://20216342.github.io/1.4.html），效果图如图 6-4 所示。表明目前项目运行正常，达到了项目初步的目标。移动端可以使用图 6-5 的二维码，查看代码运行效果。

代码的运行效果图：

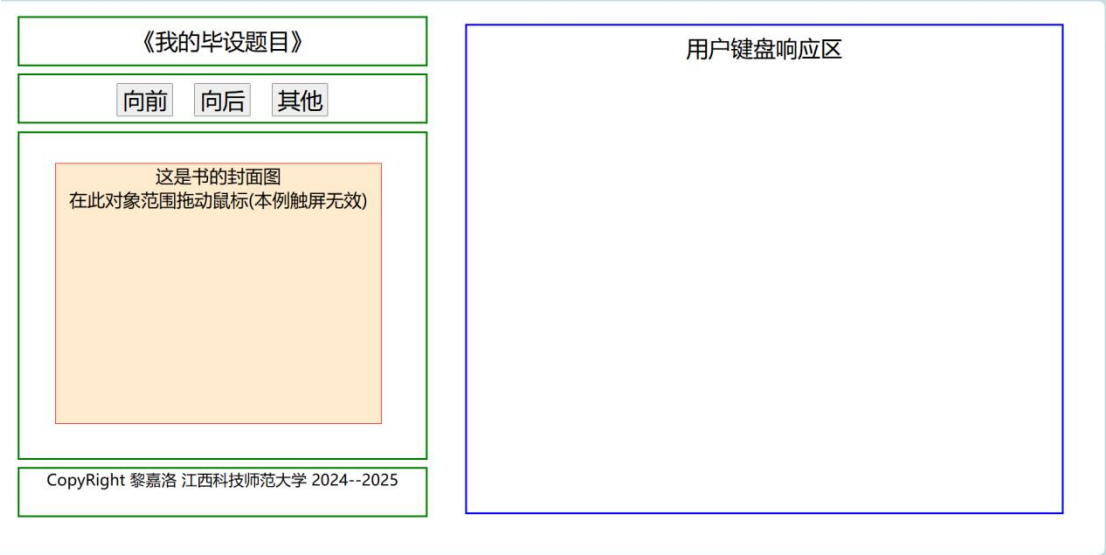


图 6-3 PC 端运行效果图

图 6-4 手机端运行效果图

图 6-5 移动端二维码入口

6.3 代码提交

本阶段完成的代码存放在 3index.html 和 3myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 6-6 所示：

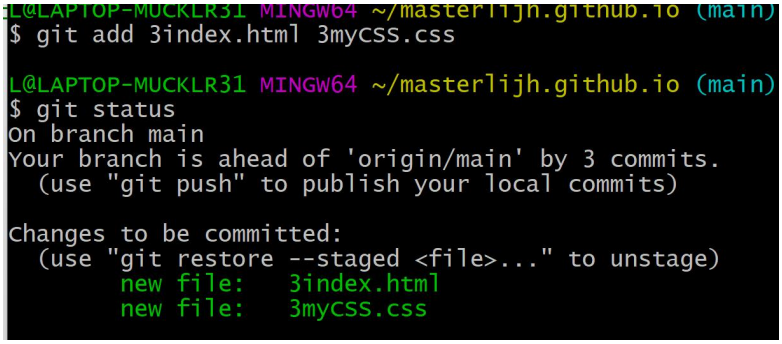


图 6-6 本次代码提交的文件列表

命令行如下：

```
git add 3index.html  3myCSS.css
git commit -m 个性化 UI 设计中鼠标模型
```

本次代码提交成功，结果如图 6-7 所示：

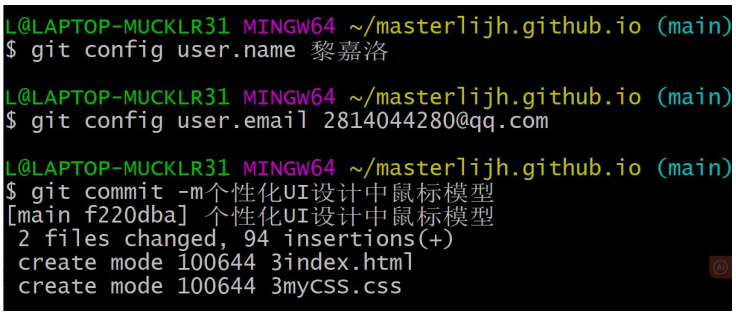


图 6-7 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：

7. 对触屏和鼠标的通用交互操作的设计开发

设计一套代码逻辑，使其能够同时适用于触屏和鼠标的交互操作，需要考虑两种输入设备的共性和差异。在软件工程中，这通常通过抽象层来实现，使得上层应用逻辑可以忽略底层的具体输入设备类型。以下是实现这一目标的基本步骤：首先，需要定义一个通用的交互接口（例如，IInputDevice），它包含了所有输入设备共有的基本操作，如点击、移动、拖拽等。这个接口应该足够抽象，以便能够适应不同类型的输入设备。接着，为每种输入设备（触屏和鼠标）实现这个通用接口。每种设备的具体实现会根据其特性有所不同。在实际应用中，需要能够识别用户当前使用的输入设备，并选择合适的模型来处理交互。这可以通过检测用户代理（User Agent）或其他方式来实现。在处理用户交互时，需要将不同设备的原始事件（如触屏的 touchstart、touchmove 和鼠标的 mousedown、mousemove）适配为通用接口中定义的方法。最后，需要对上述代码进行优化和测试，确保在不同设备和浏览器上都能正常工作。这可能包括对不同设备和操作系统的兼容性测试、性能优化等。。通过这种方式，我们可以用一套代码逻辑同时为触屏和鼠标建立对象模型，从而实现跨设备的交互操作。这样的设计不仅提高了代码的复用性，还使得应用更加易于维护和扩展。

以下是如何使用一套代码逻辑同时为触屏和鼠标建立对象模型的方法，利用

CSS 语言和 JavaScript 语言完成设计，代码块如图 7-1 所示：

```
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' ;
   }

});
```

JavaScript 块码 7-1

7.2 项目的运行和测试

本文通过 PC 平台，用 chrome 浏览器打开本阶段上传至 http 服务器的代码，网址（https://20216342.github.io/1.5.html），效果图如图 7-2 所示。还可以通过手机平台，用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址（https://20216342.github.io/1.5.html），效果图如图 7-3 所示。表明目前项目运行正常，达到了项目初步的目标。移动端可以使用图 7-4 的二维码，查看代码运行效果。

代码的运行效果图：

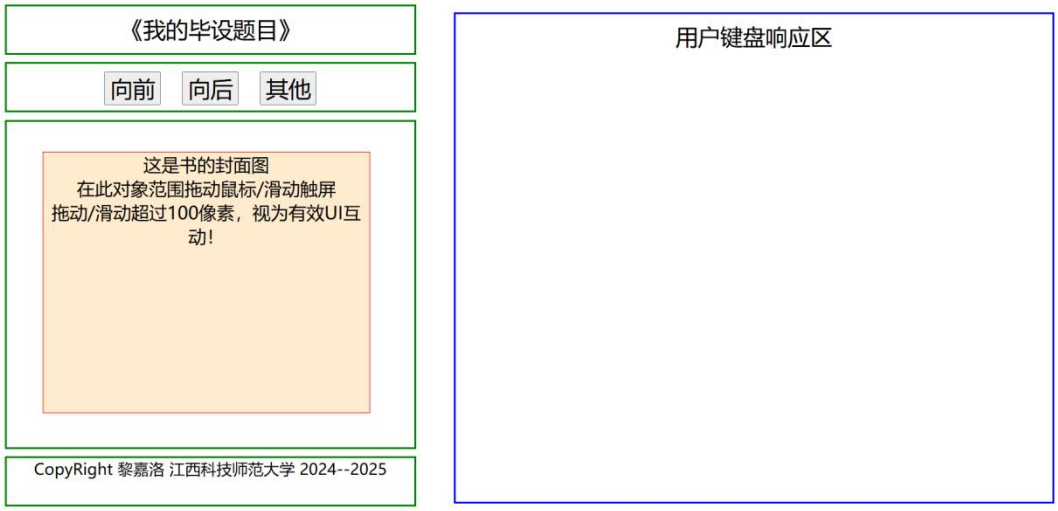


图 7-2 PC 端运行效果图

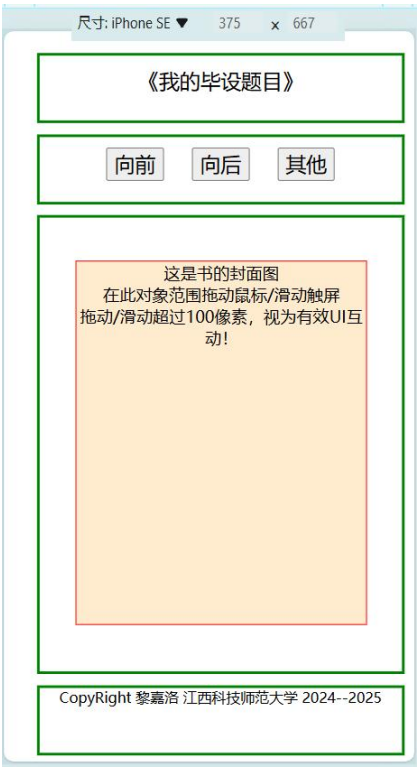


图 7-3 手机端运行效果图

图 7-4 移动端二维码入口

7.3 代码提交

本阶段完成的代码存放在 4index.html 和 4myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 7-5 所示：

```
$ git add 4index.html 4myCSS.css

L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   4index.html
        new file:   4myCSS.css
```

图 7-5 本次代码提交的文件列表

命令行如下：

```
git add 4index.html  4myCSS.css
git commit -m UI 的个性化键盘交互控制的设计开发
```

本次代码提交成功，结果如图 7-6 所示：

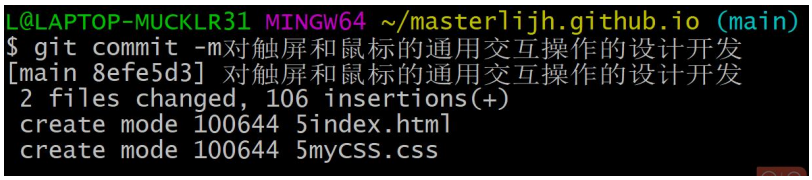
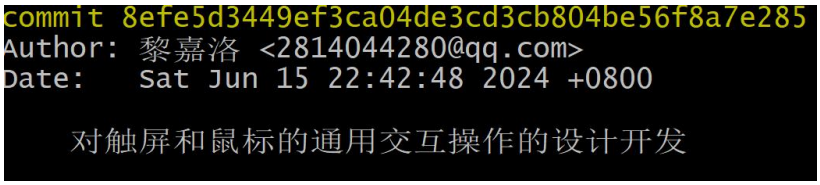


图 7-6 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：



8. UI 的个性化键盘交互控制的设计开发

8.1 设计和开发 UI 个性化键盘

在设计和开发 UI 的个性化键盘交互控制时，可以利用 keydown 和 keyup 事件来捕获和处理键盘交互。这两个事件是键盘交互的底层事件，它们分别在用户按下和释放键盘按键时触发。通过精细地控制这些事件，可以为未来的 UI 提供强大且灵活的键盘功能。分为 keydown 和 keyup 事件的基础，keydown 事件：当用户按下键盘上的任意键时触发，无论该键是否产生字符值。它通常用于处理按键的按下动作，如游戏中的连续移动。keyup 事件：当用户释放键盘上的按键时触发。它通常用于处理那些需要在按键释放时执行的动作，如输入验证或命令执行。为了捕获这些事件，可以在 DOM 元素上添加事件监听器。通常，我们会在文档级别或特定的输入元素上添加这些监听器。在某些情况下，你可能需要阻止键盘事件的默认行为。例如，你不希望按箭头键时滚动页面，可以在事件处理函数中调用在设计键盘交互时，需要考虑到不同浏览器和操作系统的兼容性。虽然 keydown 和 keyup 事件在现代浏览器中得到了很好的支持，但在旧版本浏览器中可能会有不同的行为。因此，进行全面的跨浏览器测试是必要的。在设计键盘交互时，还需要考虑到无障碍性（Accessibility）。确保所有的功能和操作都可以通过键盘访问，这对于使用键盘导航的用户（如视力受损的用户）来说是非常重要的。通过探索和利用 keydown 和 keyup 事件的底层能力，可以为 UI 的键盘交互提供强大的基础，实现更加丰富和个性化的用户体验。在设计时，应该充分

考虑用户的实际需求，创造出既直观又高效的键盘交互方式。

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)
});
```

码块 8-1

8.2 项目的运行和测试

本文通过 PC 平台，用 chrome 浏览器打开本阶段上传至 http 服务器的代码，网址（https://20216342.github.io/1.6.html），效果图如图 8-2 所示。还可以通过手机平台，用 safari 浏览器打开本阶段上传至 http 服务器的代码。网址（https://20216342.github.io/1.6.html），效果图如图 8-3 所示。表明目前项目运行正常，达到了项目初步的目标。移动端可以使用图 8-4 的二维码，查看代码运行效果。

代码的运行效果图：

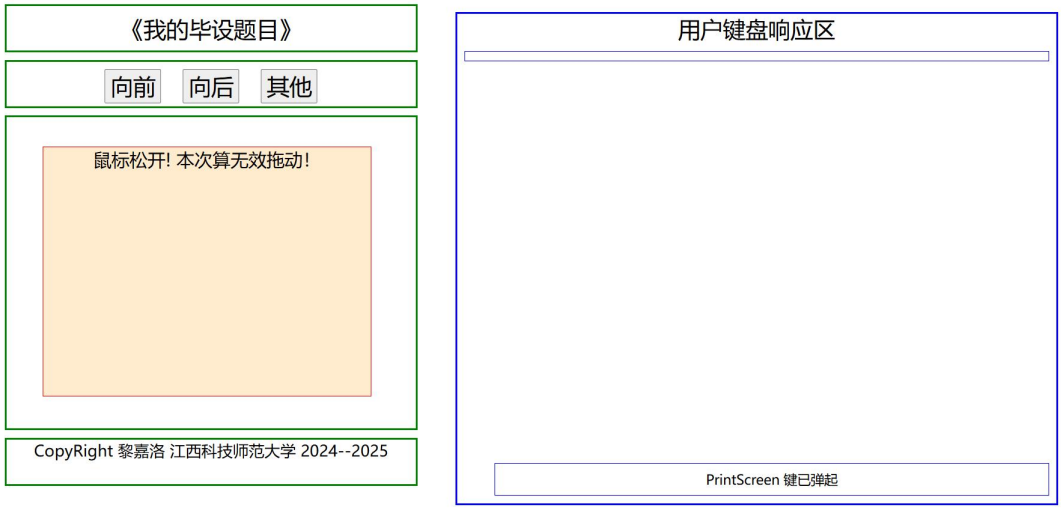


图 8-2 PC 端运行效果图

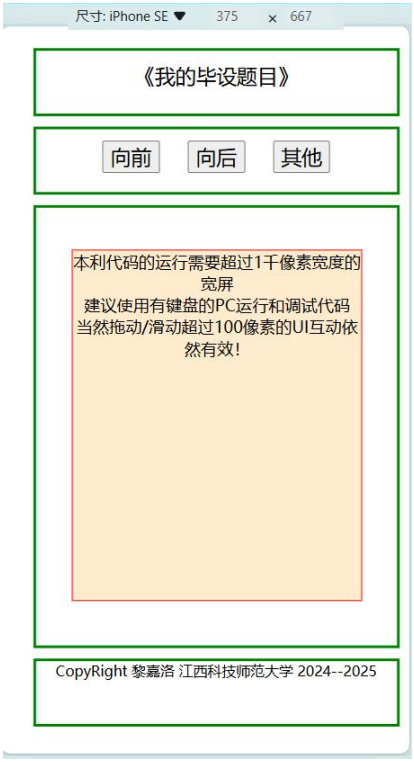


图 8-3 手机端效果运行图

图 8-4 移动端二维码入口

8.3 代码提交

本阶段完成的代码存放在 5index.html 和 5myCSS.css 两个文件中，通过 gitbash 工具作正式的代码提交。结果如图 8-5 所示：

命令行如下：

```
git add 5index.html  5myCSS.css
git commit -m 对触屏和鼠标的通用交互操作的设计开发
```

```
L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main
$ git add 5index.html 5mycss.css

L@LAPTOP-MUCKLR31 MINGW64 ~/masterlijh.github.io (main
$ git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   5index.html
        new file:   5myCSS.css
```

图 8-5 本次代码提交的文件列表

本次代码提交成功，结果如图 8-6 所示：

图 8-6 本次代码提交反馈图

gitbash 反馈代码的仓库日志如下所示：

9. 谈谈本项目中的高质量代码

这是一本关于计算机教学的书。今天，电脑就像螺丝刀一样常见，但它们要复杂得多，让它们做你想让它们做的事情并不总是那么容易。如果你给你的电脑的任务是一个常见的，很容易理解的任务，比如显示你的电子邮件或像计算器一样工作，你可以打开适当的应用程序并开始工作。但对于独特的或开放式的任务，可能没有应用程序。

这就是编程可能发挥作用的地方。编程是构建程序的行为——一组告诉计算机该做什么的精确指令。因为计算机是愚蠢的、迂腐的野兽，编程从根本上来说是乏味和令人沮丧的。幸运的是，如果你能克服这个事实，甚至享受用愚蠢的机器可以处理的方式思考的严谨性，编程是有回报的。它可以让你在几秒钟内完成手工永远做不完的事情。它是一种让你的计算机工具做它以前不能做的事情的方法。它还提供了一个很好的抽象思维练习。[6]

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。
（围绕着抽象定义函数、代码块、模型设计以及降低全局变量的使用来写）

10. 用 gitBash 工具管理项目的代码仓库和 http 服务器

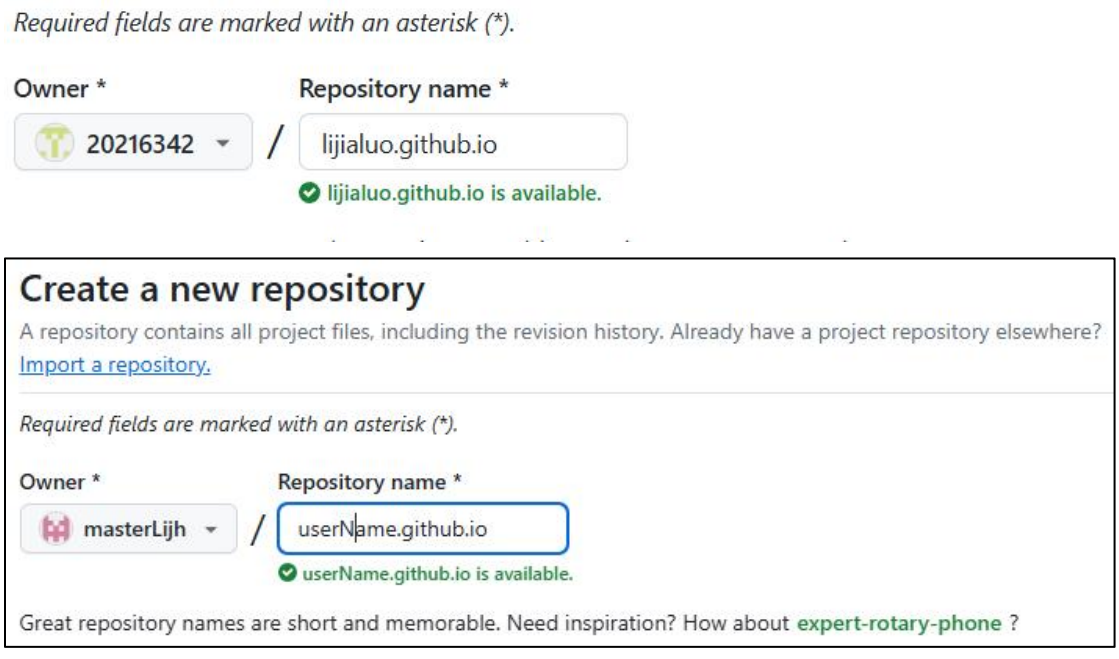
10.1 经典 Bash 工具介绍

当我们谈到命令行时，我们实际上指的是 shell。shell 是一个接受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 发行版都提供了一个来自 GNU 项目的 shell 程序，名为 bash。这个名字是 Bourne -again shell 的首字母缩略词，指的是 bash 是 sh 的增强替代品，sh 是 Steve Bourne 编写的原始 Unix shell 程序。[7]

像 Windows 一样，像 Linux 这样的类 unix 操作系统用所谓的分层目录结构来组织文件。这意味着它们被组织成树状的目录模式(在其他系统中有时称为文件夹)，其中可能包含文件和其他目录。文件系统中的第一个目录称为根目录。根目录包含文件和子目录，子目录包含更多的文件和子目录，以此类推。[7]

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



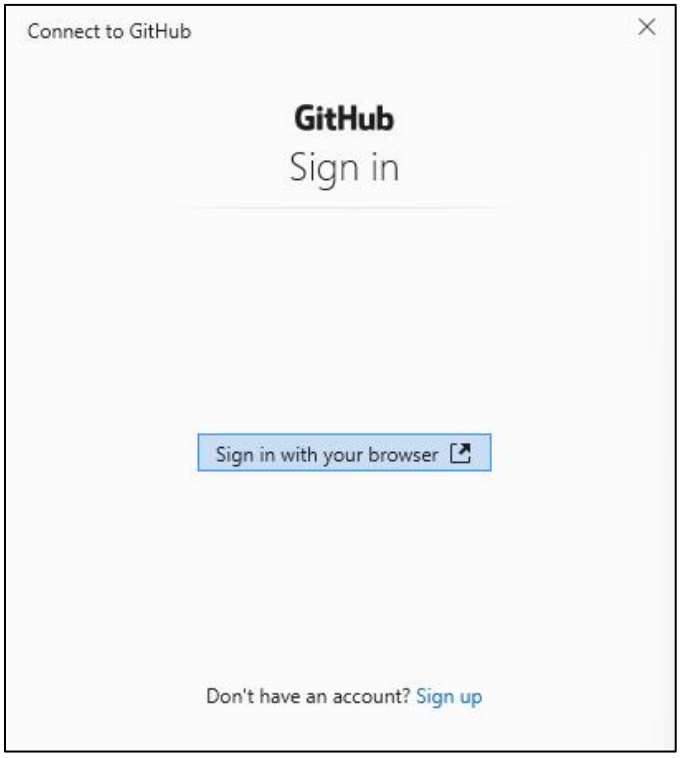
点击窗口右下角的绿色“Create repository”，则可创建一个空的远程代码仓库。

10.4 设置本地仓库和远程代码仓库的链接

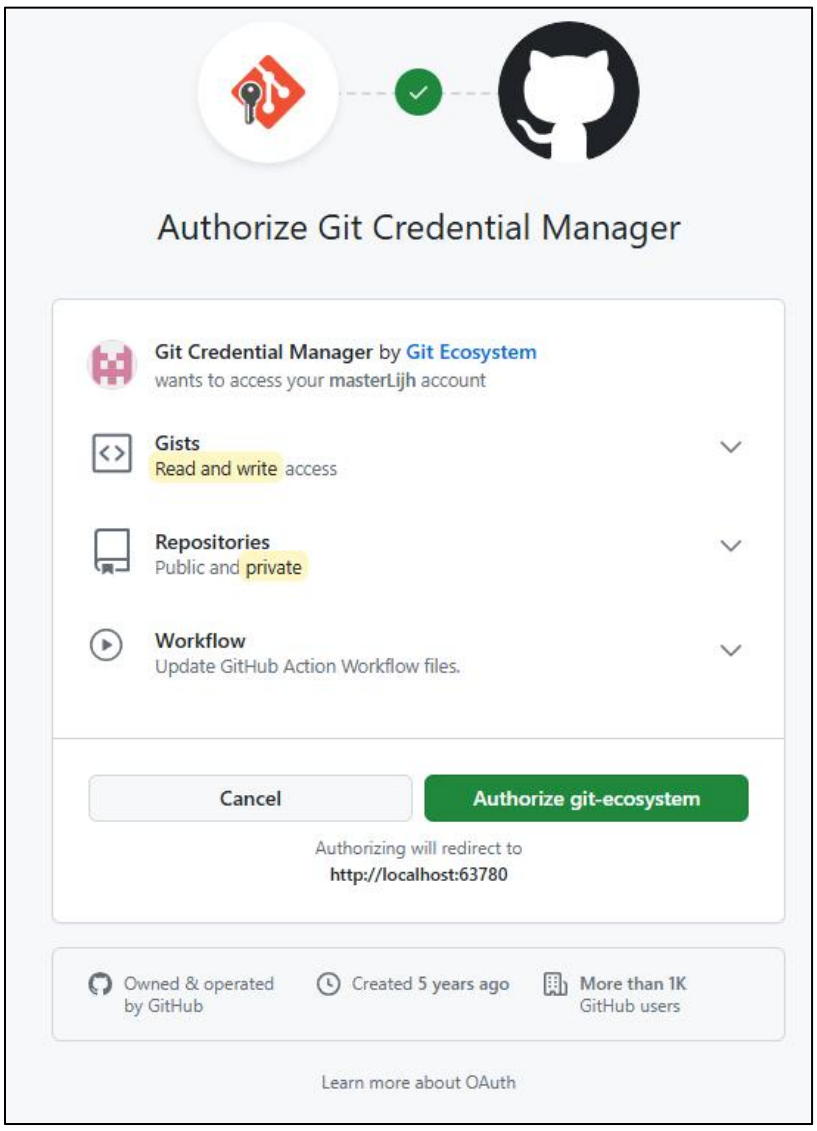
进入本地 webUI 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接

```
$ 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/20216342/lijialuo.github.io.git
$ git push -u origin main
```

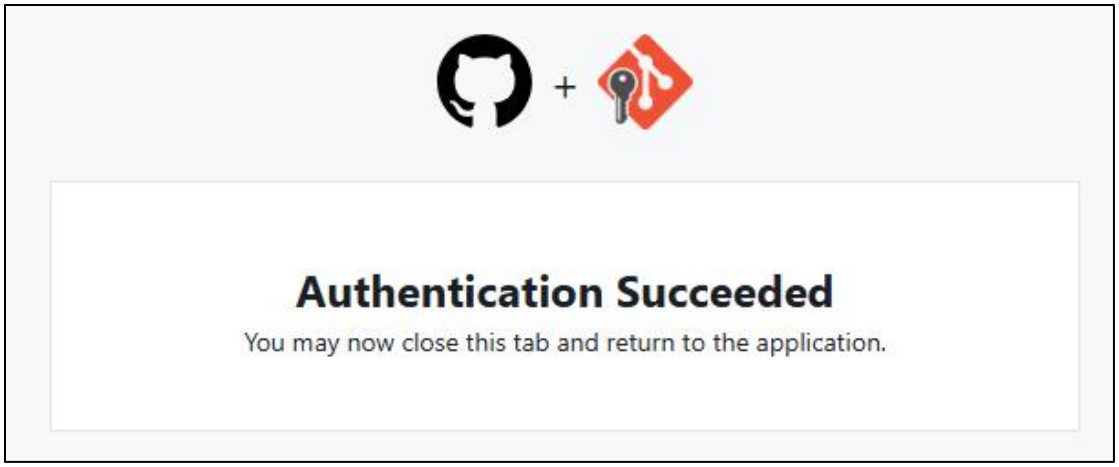
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



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：

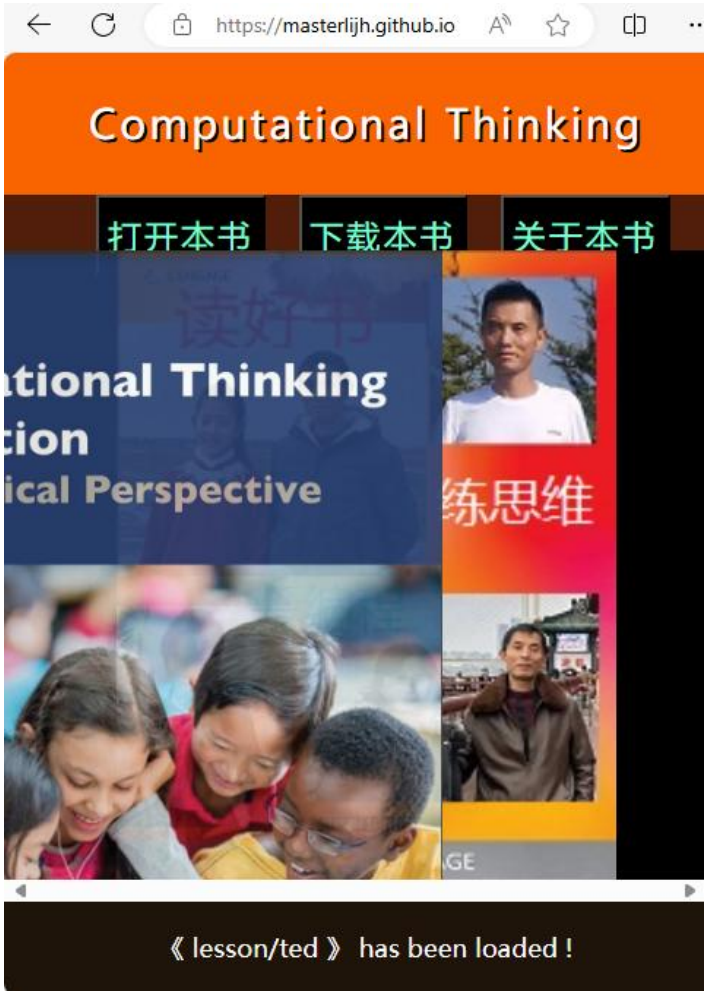


最后，GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。



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。