目录

基于 Web 客户端技术的个性化 UI 的设计和编程

科师大元宇宙产业学院 2021 级 XXX

摘要：Web 平台以其跨操作系统平台的优势成为了广泛流行的软件开发技术，本项目以 Web 客户端技术为研究学习对象，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。实现了一个个性化的用户界面, 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。为了处理好设计和开发的关系，用工程思想管理项目，本次设计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。

关键词：web 平台；软件开发；web 客户端技术

1. 前言

我们作为计算机科学技术专业的本科生，在即将完成学业之际，的确很必要设计开发一个本专业的作品，来回顾总结本学科专业学习的知识系统，梳理课程体系最核心的东西，体现我们的真实能力。

1.1 毕设任务分析

在我的毕业设计中，涉及的有关核心课程的理论包括：面向对象的程序设计语言、数据结构和算法、操作系统、软件工程等。以前这些核心课程供理论指导感觉非常抽象，加之基本上以理论知识为主，因此学完后我们感觉一直有所缺憾，本人与导师沟通后也一致认为，若能在实践层面应用这些核心课程的关键知识，则必然会在理解和技术二个维度提升自己的专业性。

因此，我认为毕业设计的内涵就是大学理论的学习在实践层面做一次综合演练和总结，期间也需要要配合学习当前最新的一些流行技术，在以形成自己对计算机软硬件体系的系统而专业的理解后，再总结撰写毕业论文，这既是毕业论文的内涵。深刻理解计算机系统对我们专业开发者而言，是非常重要的，这也是我们即将成为建设国家现代化的工程师不同于与其他专业的人的特. 色，从其他专业眼中看来，我们是计算机专业的，我们对计算机系统的理解一定不是浮于表面的，而是尽量要更加接近计算机的本质，对任何技术的理解则是能接近技术的底层和基本原理。

1.2 研学计划

我的毕设分为二个阶段完成，首先选择一条自己感兴趣的技术实践路线，把核心的技术加以整合学习，以导师的案例项目为参考，主要是理解好各个技术之间的关系，在项目中的作用和分工，更重要的是在项目实施中提升自己的写高质量的代码能力。

当仿造导师的案例的技术基本实现后，则可以视为实践和理论基本打通，此时就可进入第二个阶段，开始真正做自己的毕设软件。

第二阶段一般按软件工程的标准来规范开发：1、结合自己的问题做出定义和分析；2、设计一套合适的技术解决方案；3、按解决方案设计流程和编写相关代码，实现技术部署；4、调试代码、测试软件、性能调优。其中第 3、4 步可以发现前面步骤的问题，因此可能会在第 2，3，4 步多次循环，发现和解决第 2 步的设计失误或第 3 步的代码错误。当然大部分工作是用在第 3 步的构建代码体系和落实软件架构的具体实施和细节。

本科毕设与个人开发者类似，项目的设计和具体实现都没有经验丰富的团队，很多时候为了提高效率，方案设计的细化优化和写代码具体部署二个步骤其实是交替进行的。前者是工程师落实微观和细节层面，而后者则是设计师的工作，确保宏观层面的设计不偏离需求。

在开发期间可以产生大量开发文档，对这些文档做一个总结，再结合本专业的理论就可形成自己的论文，这个实现路径可以用实践来驱动对理论理解，进而加深本科期间学习的理论的真实体会。从实践升华到理论，再用理论实现最佳实践。

1.3 研究方法

对于写代码的本科生，必须擅长使用的一种研究方法就是“模型研究法”。这个研究方法非常具体，比如承载我们 web 应用的台式机、笔记本、手机，平板，传递在线信息要用到的互联网、服务器，沟通硬件和我们的代码之间的操作系统、浏览器、代码编辑器、编译器，这些软硬件对象，对我们而言，都值得从写代码的角度去研究。我们笼统地称它们为对象，这些对象最终会在我们大脑中被理解为抽象的模型，我们再通过分析把这些模型序化、数据化，最后写出代码来。这种行为本质上就是先在思维上“建模”，再用 OOP 语言表出来。

在 OOP 分析和开发过程中，我在毕设中试图解决的问题，也被定义成为了各级各模型。模型研究只是更为抽象，与具体的计算机语言无关，在毕设中我也尝试使用国际标准 UML 语言来建立抽象模型。我感觉采用 UML 模型研究法和面向对象的程序设计方法的目标是一致的，只是在不同层面分析表述问题而已。因此采用模型研究法，我的毕设用例设计采用了类似的

UML 对问题建模，景观 UML 比较抽象，设计准确有一定难度，而使用 OOP 程序对画好的模型开展程序设计则更为具体直观，通过熟悉的 OOP 语言和代码运行环境运行和调试模型，我们甚至可以倒推出模型设计的问题和缺陷。因此，可能直接写代码建立模型研究模型，代码跑通后，再利用 UML 语言绘制模型，作为代码的文档资料则更合理。

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。如下图 2-1 所示：

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 2-2 所示：

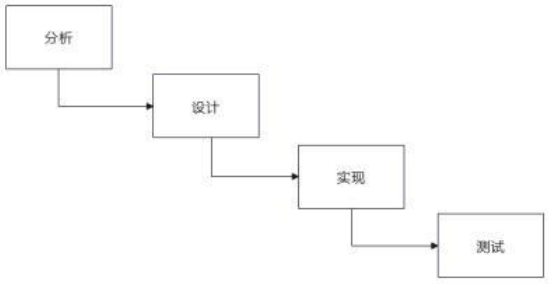


图 2-1 瀑布模型图

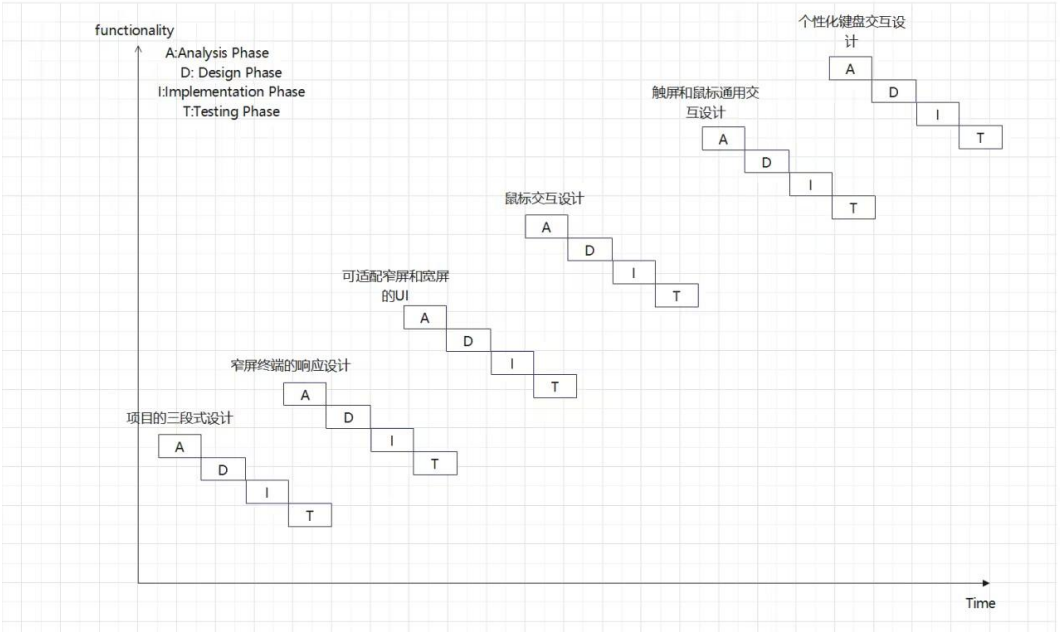


图 2-1 迭代图

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 3-1 用例图所示：

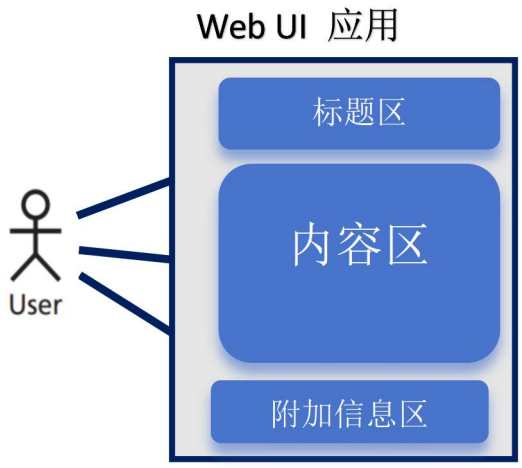


图 3-1 用例图

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 3-2 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 3-3 的二维码，运行测试本项目的第一次开发的阶段性效果。

《 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

图 3-2 PC 端运行效果图

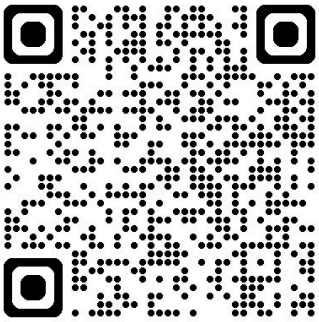


图 3-3 移动端二维码

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 wu
$ cd wu
$ git init
$ git config user.name 吴政霖
$ git config user.email 2771824488@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：

```
$ 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：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git log
commit 08692d665bc3a2ed06499bda618396167be4c665 (HEAD -> master)
Author: 吴政霖 <2771824488@qq.com>
Date:   Wed Jun 19 12:49:35 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

4.1 分析和设计

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。如图 4-1 用例图所示：

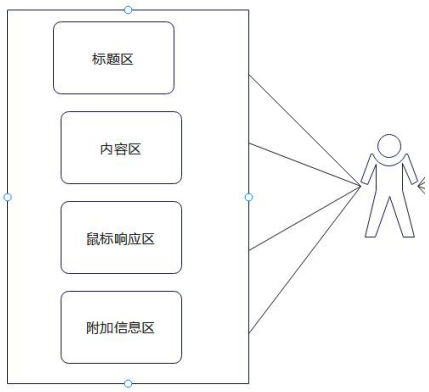


图 4-1 用例图

4.2 项目的实现和编程

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 4-2 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 4-2

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 4-3 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 4-3

4.3 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 4-4 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 4-5 的二维码，运行测试本项目的第一次开发的阶段性效果。

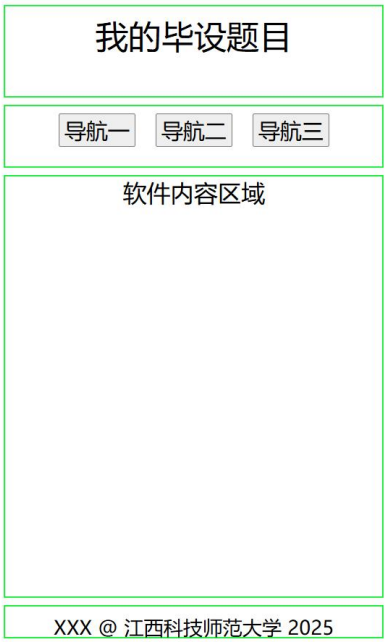


图 4-4 PC 端运行效果图

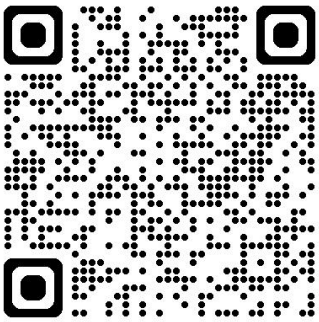


图 4-5 移动端二维码

4.4 项目的代码提交和版本管理

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd wu
$ git init
```

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 项目第二版：移动互联时代的 UI 开发初步——窄屏终端的响应式设计
```

成功提交代码后，gitbash 的反馈如下所示：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git commit -m 项目第二版:移动互联时代的UI开发初步--窄屏终端的响应式设计
[master (root-commit) 3167d45] 项目第二版:移动互联时代的UI开发初步--窄屏终端的响
应式设计
 2 files changed, 0 insertions(+), 0 deletions(-)
```

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：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git log
commit 3167d45eaa9fc3d94f1adcef6b065b0bcf1c68ff (HEAD -> master)
Author: 吴政霖 <2771824488@qq.com>
Date:   Wed Jun 19 12:33:31 2024 +0800

    项目第二版:移动互联时代的UI开发初步--窄屏终端的响应式设计
```

5. 应用响应式设计技术开发可适配窄屏和宽屏的 UI

5.1 分析和设计

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。如图 5-1 用例图所示：

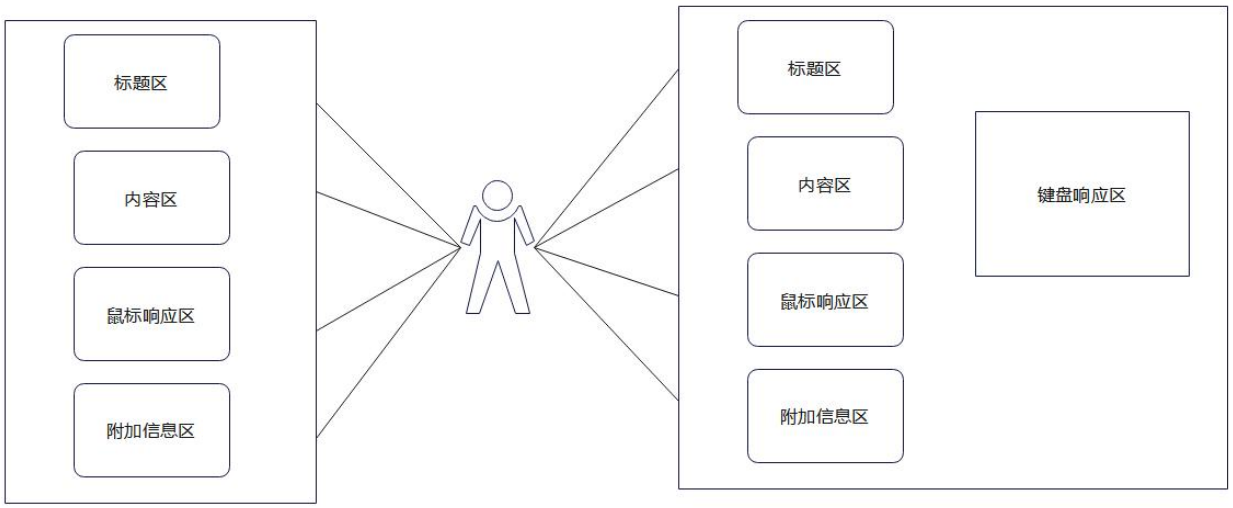


图 5-1 用例图

5.2 项目的实现和编程

适用移动互联时代阐述移动互联时代的用户终端的多样性，使用 css 语言和 JavaScript 语言实现响应式设计，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 - 4*baseFont + "px";
```

代码块 5-2

5.3 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 5-3 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 5-4 的二维码，运行测试本项目的第一次开发的阶段性效果。

图 5-3 PC 端运行效果图

图 5-4 移动端二维码

5.4 项目的代码提交和版本管理

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd wu
$ git init
```

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 项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的 UI
```

成功提交代码后，gitbash 的反馈如下所示：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git commit -m 项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的UI
[master 5dc7a77] 项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的UI
 2 files changed, 0 insertions(+), 0 deletions(-)
```

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：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git log
commit 5dc7a778d57e512288464ff1c20306486155f6aa (HEAD -> master)
Author: 吴政霖 <2771824488@qq.com>
Date:   Wed Jun 19 12:53:34 2024 +0800

    项目第三版：应用响应式设计技术开发可适配窄屏和宽屏的UI
```

6. 个性化 UI 设计中对鼠标交互的设计开发

6.1 分析和设计

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。如图 6-1 用例图所示：

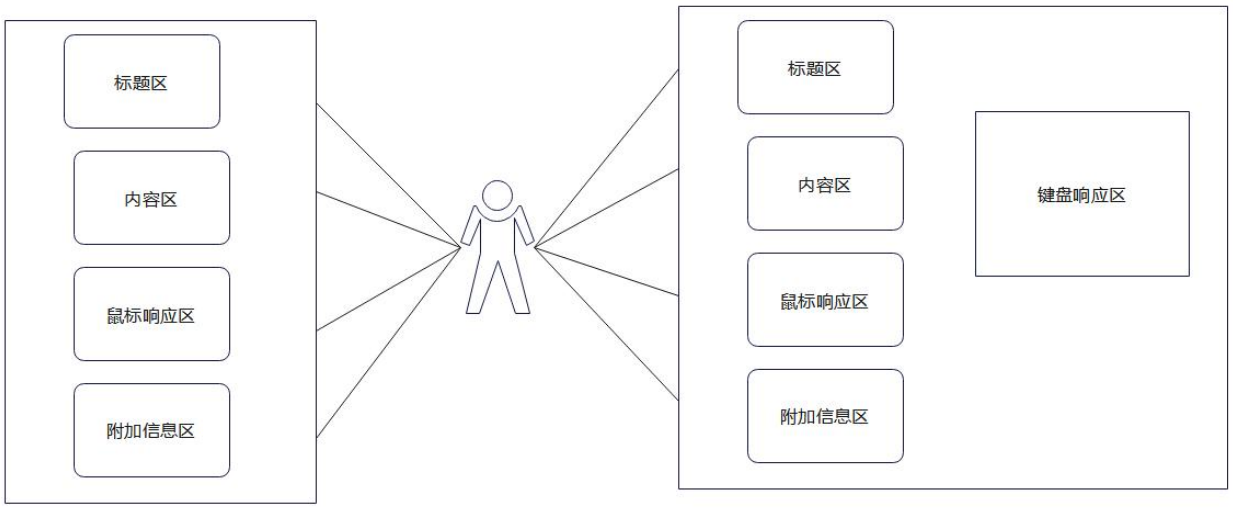


图 6-1 用例图

6.2 项目的实现和编程

用一套代码逻辑为鼠标建立对象模型，尝试对鼠标设计 UI 控制，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 $
```

代码块 6-2

6.3 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 6-3 的二维码，运行测试本项目的第一次开发的阶段性效果。

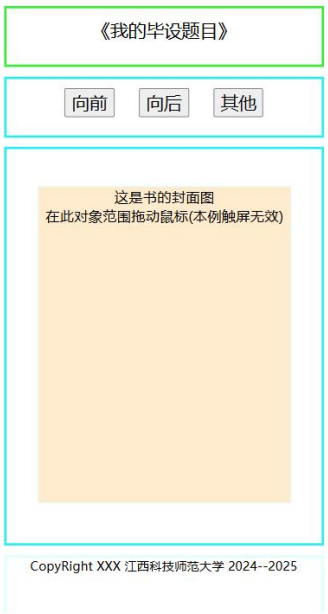


图 6-3 PC 端运行效果图

图 6-4 移动端二维码

6.4 项目的代码提交和版本管理

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd wu
$ git init
```

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 项目第四版：个性化 UI 设计中对鼠标交互的设计开发
```

成功提交代码后，gitbash 的反馈如下所示：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git commit -m 项目第四版：个性化UI设计中对鼠标交互的设计开发
[master a1b7628] 项目第四版：个性化UI设计中对鼠标交互的设计开发
 2 files changed, 0 insertions(+), 0 deletions(-)
```

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：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git log
commit a1b76280cf605f9e3bcc1352b08a43d5f2170169 (HEAD -> master)
Author: 吴政霖 <2771824488@qq.com>
Date:   Wed Jun 19 13:04:57 2024 +0800

    项目第四版：个性化UI设计中对鼠标交互的设计开发
```

7. 对触屏和鼠标的通用交互操作的设计开发

7.1 分析和设计

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。如图 7-1 用例图所示：

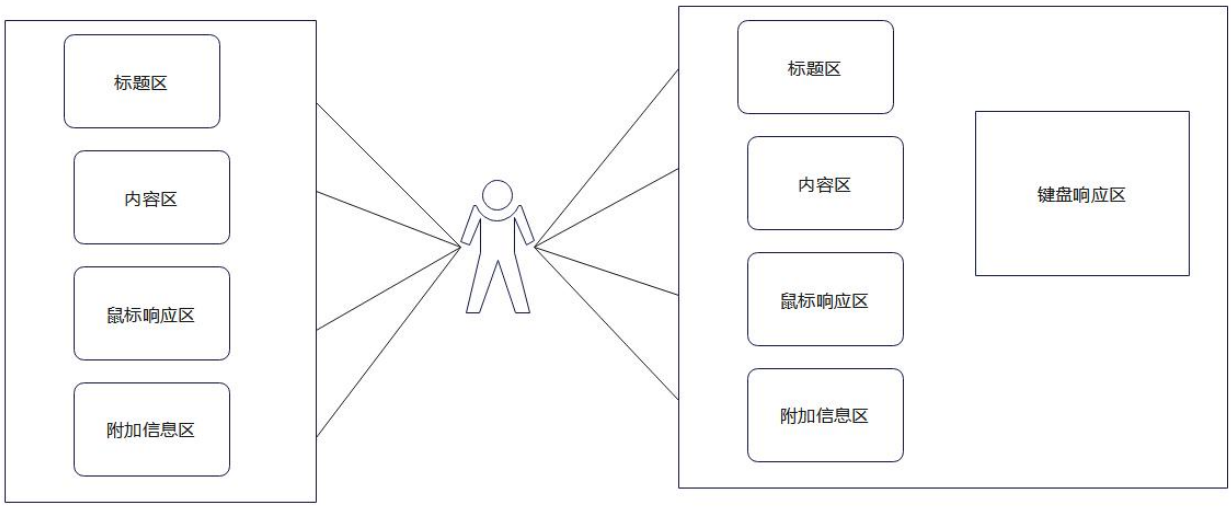


图 7-1 用例图

7.2 项目的实现和编程

阐述用一套代码逻辑同时为触屏和鼠标建立对象模型，如代码块 7-2 所示：

```
var UI = {};
  if(window.innerWidth>600){
    UI.appWidth=600;
      }else{
    UI.appWidth = window.innerWidth;
  }

  UI.appHeight = window.innerHeight;

  let baseFont = UI.appWidth /20;
  //通过改变 body 对象的字体大小，这个属性可以影响其后代
```

```
  document.body.style.fontSize = baseFont +"px";
  //通过把 body 的高度设置为设备屏幕的高度，从而实现纵向全屏
  //通过 CSS 对子对象百分比（纵向）的配合，从而达到我们响应式设计的目标
  document.body.style.width = UI.appWidth - 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(ev.touches){
     $("bookface").textContent= "触屏事件结束!";
     if(Math.abs(Pointer.deltaX) > 100){
         $("bookface").textContent += "，这是有效触屏滑动! "  ;
     }else{
      $("bookface").textContent += " 本次算无效触屏滑动! "  ;
    $("bookface").style.left = '7%' ;
   }
   }else{
```

```
    $("bookface").textContent= "鼠标松开!";
    if(Math.abs(Pointer.deltaX) > 100){
        $("bookface").textContent += "，这是有效拖动！"  ;
    }else{
     $("bookface").textContent += " 本次算无效拖动！"  ;
   $("bookface").style.left = '7%' ;
    }
  }
 };
let handleMoving = function(ev){
   ev.preventDefault();
  if (ev.touches){
    if (Pointer.isDown){
    console.log("Touch is moving");
    Pointer.deltaX = parseInt( ev.touches[0].pageX - Pointer.x );
    $("bookface").textContent= "正在滑动触屏，滑动距离：" + Pointer.deltaX +"px 。";
    $('bookface').style.left =  Pointer.deltaX + 'px' ;
    }
  }else{
    if (Pointer.isDown){
    console.log("Pointer isDown and moving");
    Pointer.deltaX = parseInt( ev.pageX - Pointer.x );
    $("bookface").textContent= "正在拖动鼠标，距离：" + Pointer.deltaX +"px 。";
    $('bookface').style.left =  Pointer.deltaX + 'px' ;
    }
  }
 };

 $("bookface").addEventListener("mousedown",handleBegin );
 $("bookface").addEventListener("touchstart",handleBegin );
 $("bookface").addEventListener("mouseup", handleEnd );
 $("bookface").addEventListener("touchend",handleEnd );
 $("bookface").addEventListener("mouseout", handleEnd );
 $("bookface").addEventListener("mousemove", handleMoving);
 $("bookface").addEventListener("touchmove", handleMoving);
 $("body").addEventListener("keypress", function(ev){
   $("aid").textContent += ev.key ;
 });
} //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 $
```

代码块 7-2

7.3 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 7-3 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 7-4 的二维码，运行测试本项目的第一次开发的阶段性效果。

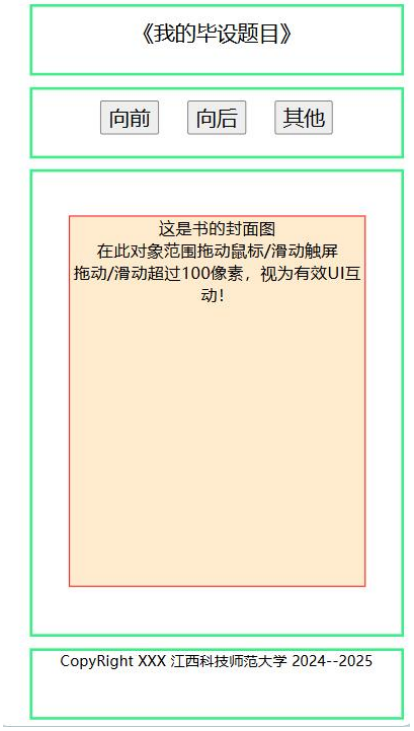


图 7-3 PC 端运行效果图

图 7-4 移动端二维码

7.4 项目的代码提交和版本管理

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd wu
$ git init
```

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 项目第六版：对触屏和鼠标的通用交互操作的设计开发
```

成功提交代码后，gitbash 的反馈如下所示：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git commit -m 项目第六版：对触屏和鼠标的通用交互操作的设计开发
[master 0b6ea15] 项目第六版：对触屏和鼠标的通用交互操作的设计开发
 2 files changed, 0 insertions(+), 0 deletions(-)
```

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：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git log
commit 0b6ea150b4511c346c9684d31b406c818a262ee1 (HEAD -> master)
Author: 吴政霖 <2771824488@qq.com>
Date:   Wed Jun 19 13:22:13 2024 +0800

    项目第六版：对触屏和鼠标的通用交互操作的设计开发
```

8. UI 的个性化键盘交互控制的设计开发

8.1 分析和设计

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。如图 8-1 用例图所示：

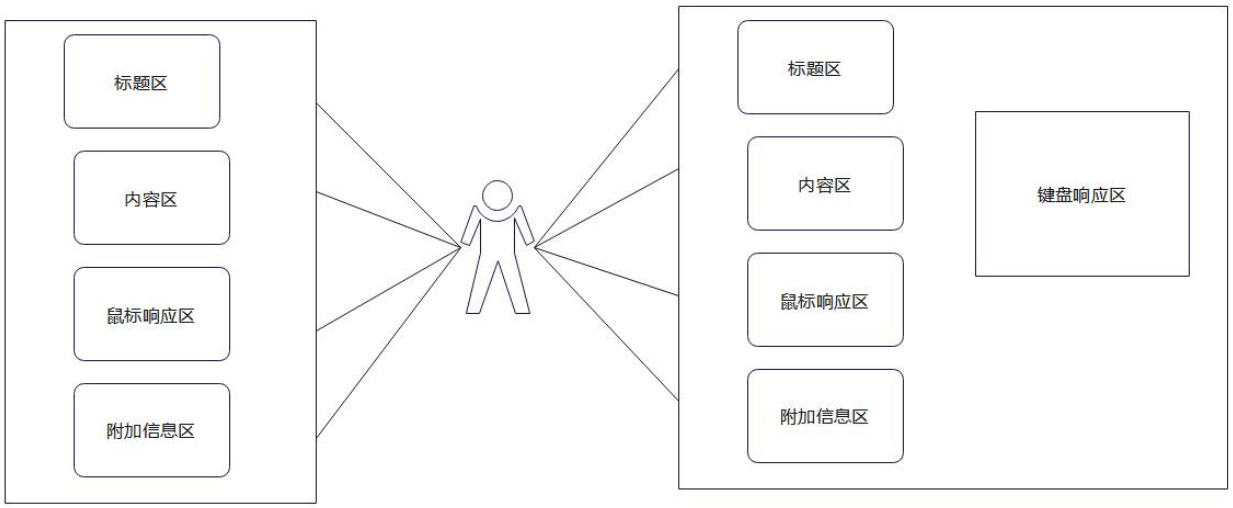


图 8-1 用例图

8.2 项目的实现和编程

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。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。如代码块 8-2 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)
 });
```

代码块 8-2

8.3 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 8-3 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 8-4 的二维码，运行测试本项目的第一次开发的阶段性效果。

图 8-3 PC 端运行效果图

图 8-4 移动端二维码

8.4 项目的代码提交和版本管理

进入 gitBash 命令行后，按次序输入以下命令：

```
$ cd wu
$ git init
```

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 项目第五版：UI 的个性化键盘交互控制的设计开发
```

成功提交代码后，gitbash 的反馈如下所示：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git commit -m 项目第五版：UI的个性化键盘交互控制的设计开发
[master da77bf6] 项目第五版：UI的个性化键盘交互控制的设计开发
 2 files changed, 0 insertions(+), 0 deletions(-)
```

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：

```
86132@DESKTOP-9TELICF MINGW64 /wu (master)
$ git log
commit da77bf6db7943c9937b109f9bc24041bab96b700 (HEAD -> master)
Author: 吴政霖 <2771824488@qq.com>
Date:   Wed Jun 19 13:09:01 2024 +0800

    项目第五版：UI的个性化键盘交互控制的设计开发
```

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。（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）

```
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+")";
}
   };
```

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

Git Bash 是一个命令行界面工具，它提供了类似于 Linux 终端的环境，可以在 Windows 系统中执行 Git 命令。您可以从 Git 官方网站下载 Git Bash，并按照安装向导进行安装。安装完成后，您可以在 Git Bash 中使用 Git 命令来管理和操作您的代码库。

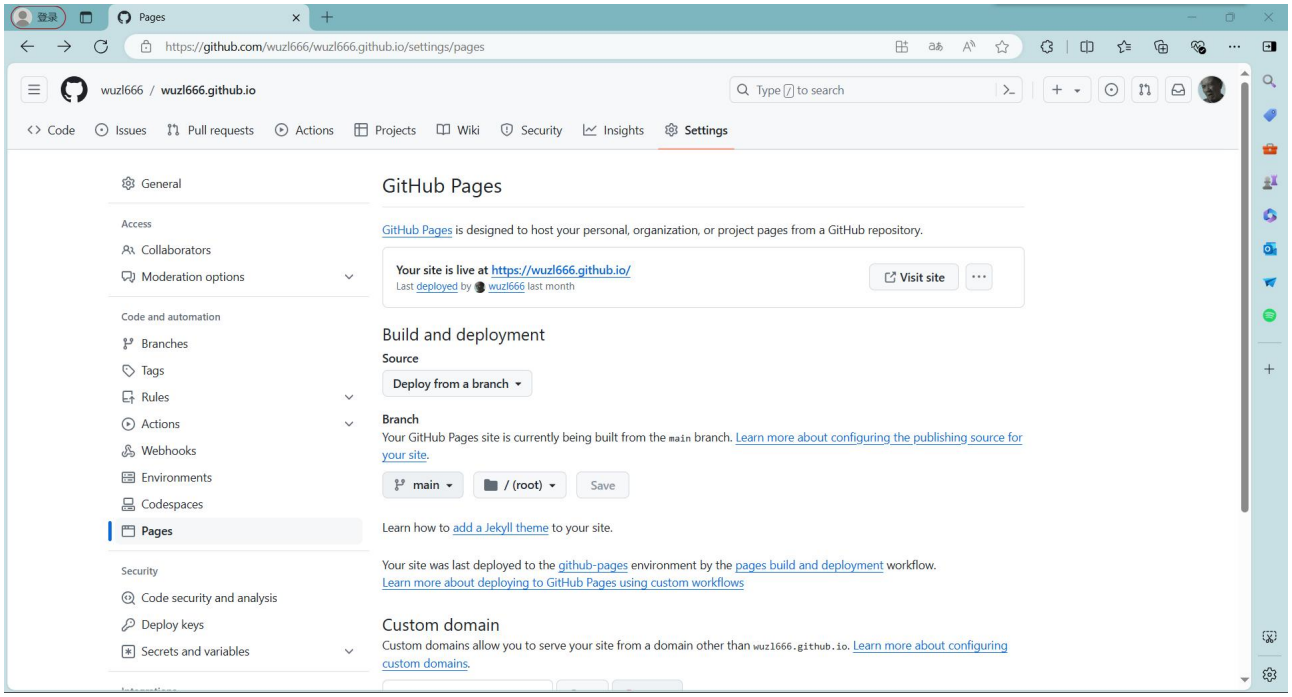
Git Bash 是 git(版本管理器)中提供的一个命令行工具，外 观类似于 Windows 系统内置的 cmd 命令行工具。可以将 Git Bash 看作是一个终端模拟器，它提供了类似于 Linux 和 Unix 系统下 Bash Shell 环境的功能。通过 Git Bash，用户可以在 Windows 系统中运行基于 Bash 的命令行，使用一些常见的 Linux 命令以及 Git 命令。

Git Bash 内置了一些常用的 Linux 命令，使得在 Windows 系统上执行 Shell 脚本或进行常规的命令行操作更加方便。除此之外，Git Bash 还提供了 Git 命令的支持，使得用户可以直接在 Windows 系统中使用 Git 功能，如创建版本库、提交变更、查看日志等。

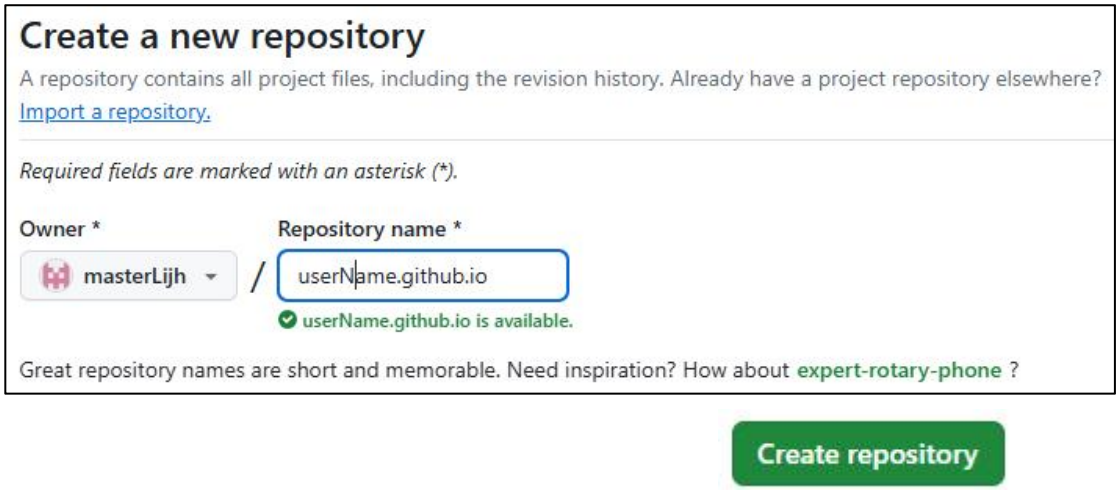
尽管 Git Bash 提供了类似于 Linux 和 Unix 系统的 Shell 环境和命令，但需要注意的是它并不完全等同于真正的 Linux 或 Unix 系统。有些特定的 Shell 命令和功能可能会因为操作系统差异而有所不同。

通过 Git Bash，Windows 用户可以访问常见的 Unix 命令，如 ls、cd、cat 以及一些高级命令，如 grep、awk 和 sed。此外，Git Bash 还提供了完全支持 Git 的环境，包括 git clone、git pull、git push 等命令，这使得 Windows 用户可以轻松使用 Git 进行版本控制，与其他开发人员协作。

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。

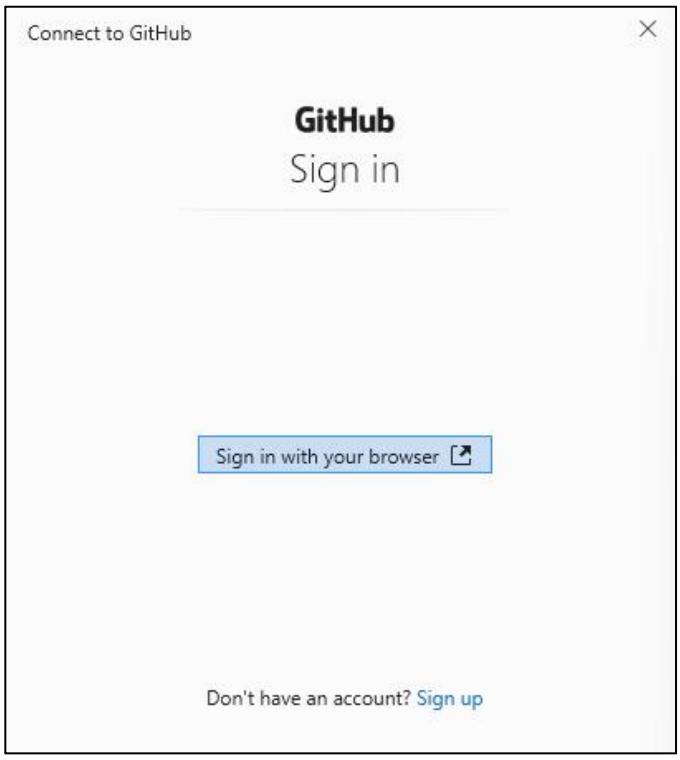
10.4 设置本地仓库和远程代码仓库的链接

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

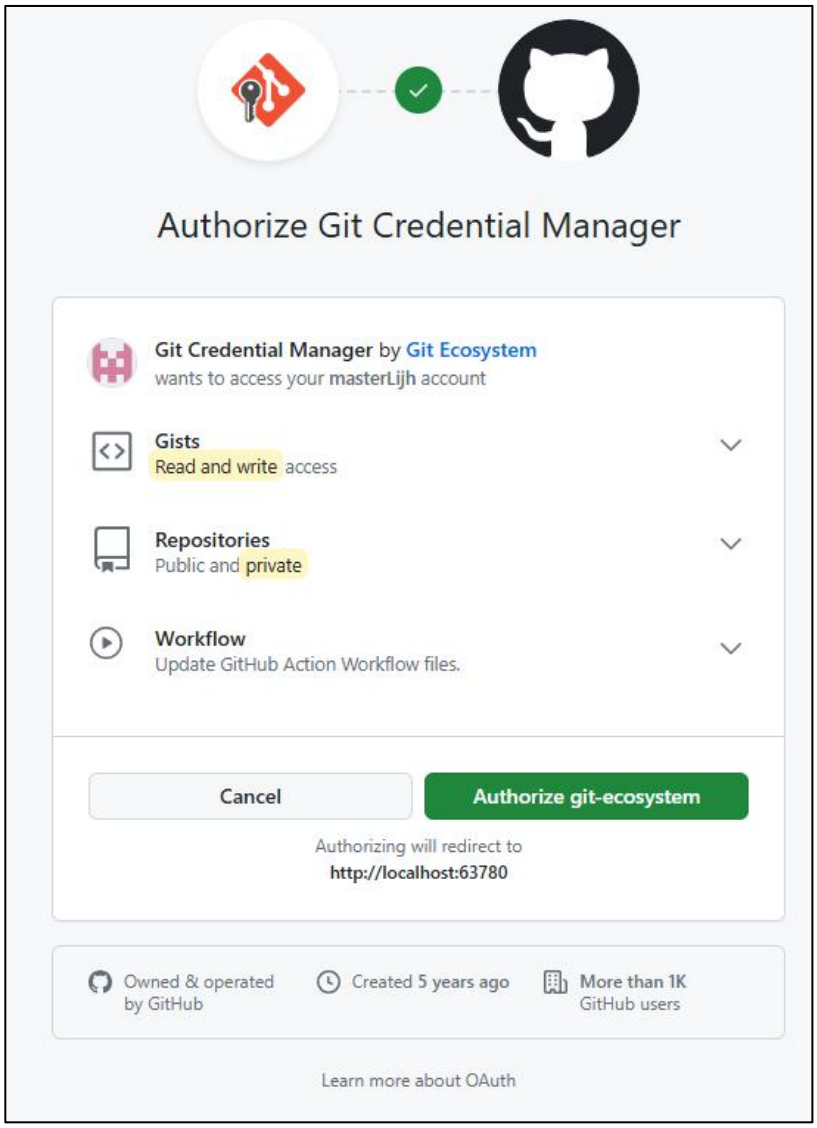
```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md
$ git init
```

```
$ 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/wuzl666/wuzl666.github.io.git
$ git push -u origin main
```

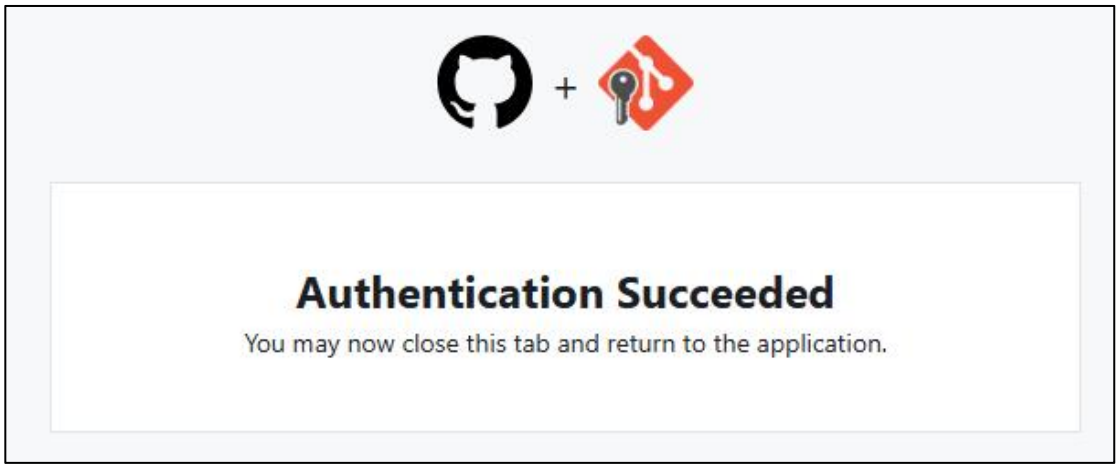
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



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：

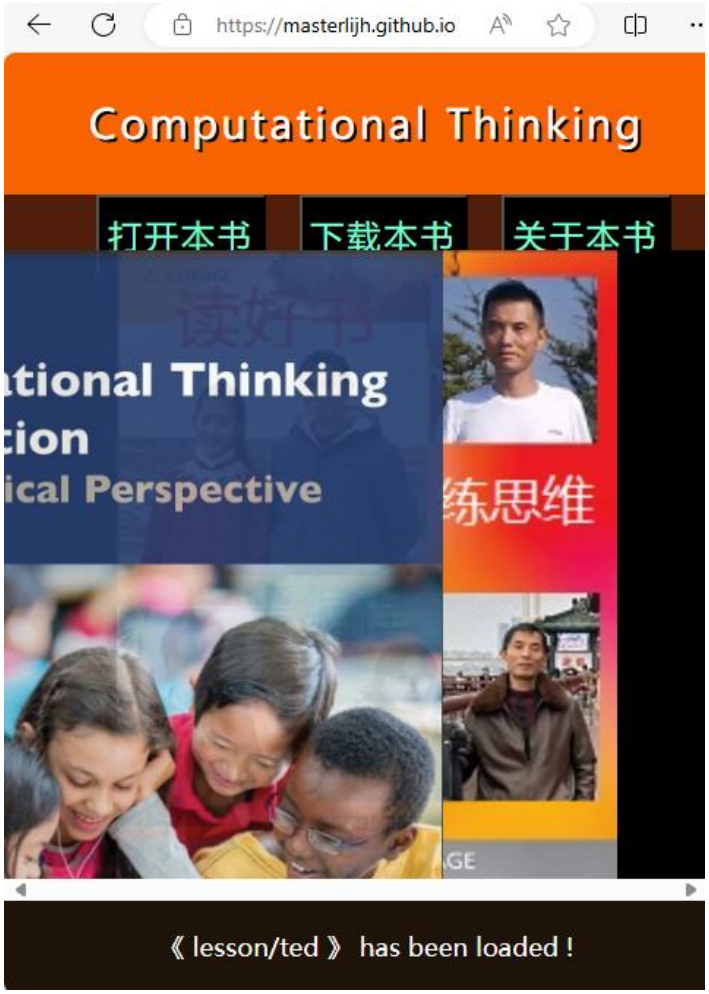


最后，GitHub 平台反馈：gitBash 和 gitHub 平台成功实现远程链接。



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