

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：曾令昱

学 号：20213633

指导教师：李健宏

2024 年 5 月 21 日

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。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：分析 D：设计 I：实现 T：测试），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。

关键词：web 客户端技术；UI 的交互功能；全球互联网的部署

Abstract:In recent ten years, html5 as the core of the web standard software development technology with its cross-platform, open source advantages are widely used in various fields of application software development. Through the analysis of the completion task, the project chooses html5 web client technology as the technical route, and carries out the research and practice of program design and software development. Through extensive access to technical books, developer forums, and literature, a personalized user interface (UI) application was designed and developed. In the development, html language was comprehensively applied to conduct content modeling, css language to develop UI appearance design, and javascript language programming to realize UI interaction functions. In addition to directly using the API at the lowest level of the web client, each code of the project was written manually one by one, without importing any code (framework and library) of others. This project also adopts responsive design and programming, which can intelligently adapt to the diversified needs of users' screens in the era of mobile Internet; In addition, a large number of object-oriented programming ideas are used, such as using code to build a general pointer model, which only uses a set of code to achieve the control of the mouse and touch screen, to achieve high quality code, which is also the highlight of this project. From the perspective of engineering management, the incremental development mode adopted in this project has carried out six incremental code reconfiguration (A: analysis D: design I: implementation T: testing) in a way of gradual refinement, thus achieving the design, development and testing of the project more happily. From the perspective of open source and sharing of code, this project adopts git tool for version management, rebuilds the code six times in the long development process and formally submits the code, in addition to modifying and submitting the code twice in the test. Finally, gitbash tool is used to upload the code warehouse of this project to the famous github. Using the http server provided by github, this project realizes the deployment of UI application on the global Internet, and we can conveniently and efficiently access this program across platforms through the address and two-dimensional code.

Key Words:web client technology; Interactive function of UI; Deployment of the global Internet

1. 前言	1
1.1. 毕业设计和毕业论文分析	1
1.2. 研究的计划和研究方法	1
1.3. 研学计划	2
2. 技术总结和文献综述	1
2.1. Web platform and Web Programming	1
2.2. web 平台和客户端的过去和现在	1
2.3. 增量式开发模式	2
3. 内容设计概要	3
3.1. 分析和设计	3
3.2. 项目的实现和编程	4
3.3. 项目的运行和测试	4
3.4. 项目的代码提交和版本管理	6
4. 移动互联时代的 UI 开发初步——窄屏终端的响应式设计	7
4.1. 分析和设计	7
4.2. 项目的实现和编程	8
4.3. 项目的运行和测试	11
4.4. 项目的代码提交	11
5. 宽屏和窄屏通用的响应式设计和代码实现	12
5.1. 分析和设计	12
5.2. 项目的实现和编程	13
5.3. 项目的运行和测试	14
5.4. 项目的代码提交	14
6. 个性化 UI 设计中鼠标模型	15
6.1. 分析和设计	15
6.2. 项目的实现和编程	16
6.3. 项目的运行和测试	17
6.4. 项目的代码提交	17
7. 对触屏和鼠标的通用交互操作的设计开发	18
7.1. 分析和设计	18
7.2. 项目的实现和编程	19
7.3. 项目的运行和测试	22
7.4. 项目的代码提交	22
8. UI 的个性化键盘交互控制的设计开发	23
8.1. 分析和设计	23
8.2. 项目的实现和编程	24
8.3. 项目的运行和测试	25
8.4. 项目的代码提交	26
9. 用 git 工具展开代码的版本管理和发布	26
9.1. 经典 Bash 工具介绍	26
9.2. 通过 gitHub 平台实现本项目的全球域名	27
9.3. 创建一个空的远程代码仓库	27
9.4. 设置本地仓库和远程代码仓库的链接跨世纪的经典 Bash 工具	27
10. 谈谈本项目中的高质量代码	29
参考文献	31

1. 前言

1.1.毕业设计和毕业论文分析

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。

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（computing system）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.研究的计划和研究方法

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^[1]。

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，这个实现路径学生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. 研学计划

学习计划	学习时期	结果
第一次开发：设计内容的概要	2024/3/1——2024/3/15	开发成功
第二次开发：进行窄屏终端的设计	2024/3/16——2024/4/1	开发成功
第三次开发：宽、窄屏通用 UI 设计	2024/4/2——2024/4/15	开发成功
第四次开发：个性化鼠标模型设计	2024/4/16——2024/5/1	开发成功
第五次开发：触屏和鼠标通用交互设计	2024/5/2——2024/5/15	开发成功
第六次开发：个性化键盘交互 UI 设计	2024/5/16——2024/6/1	开发成功
论文编写	2024/6/2——2024/6/15	完成编写

2.技术总结和文献综述

2.1.Web platform and Web Programming

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，这也是我的毕设项目应用的技术路线。

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。这里的“内容”是指文本、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]。

HTML, CSS, JavaScript 也是体现人类社会化大生产的分工的智慧，MVC 设计模式，Model: 用 HTML 标记语言建模，View: 单独用 CSS 语言来实现外观，Controller: 用 JavaScript 结合前二者，实现功能层面和微观代码的所有控制。

2.2.web 平台和客户端的过去和现在

过去，Web 平台和客户端在功能和性能上存在明显的差异。Web 平台通常运行在浏览器上，受限于网络速度和浏览器性能，因此往往比客户端应用程序更慢和不稳定。而客户端应用程序则可以更高效地利用本地资源，提供更快的速度和

更好的用户体验。

然而，随着互联网技术的发展和浏览器的不断升级，Web 平台在功能和性能上已经取得了长足的进步。现在的 Web 应用程序可以实现与客户端应用程序相媲美的用户体验，包括复杂的交互和动画效果。此外，Web 平台还具有更大的灵活性和跨平台性，可以在不同设备和操作系统上运行，而无需定制不同版本的应用程序。

客户端应用程序在过去是主流，但现在随着移动互联网的普及和 Web 技术的进步，Web 平台已经成为各种应用程序的首选平台之一。许多公司和开发者选择开发基于 Web 的应用程序，以便跨平台运行并更快地推出新的功能和更新。

总的来说，Web 平台和客户端应用程序在现在已经趋向一致，两者在功能和性能上的差距已经逐渐缩小。随着技术的不断进步和发展，预计未来 Web 平台将继续发展，成为更多应用程序开发的主流平台之一。

2.3.增量式开发模式

软件生命周期中的开发过程包括四个阶段：分析、设计、实现和测试。开发过程有几个模型。两个最常见的模型：瀑布模型和增量模型。

瀑布模型由于无法灵活应对变更需求、无法快速适应用户反馈、风险管理不足、可能导致沟通不畅等问题，不适合我们当前设计的开发，因此此文不加以讨论。

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。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^[4]。图 1 显示了增量模型的概念。

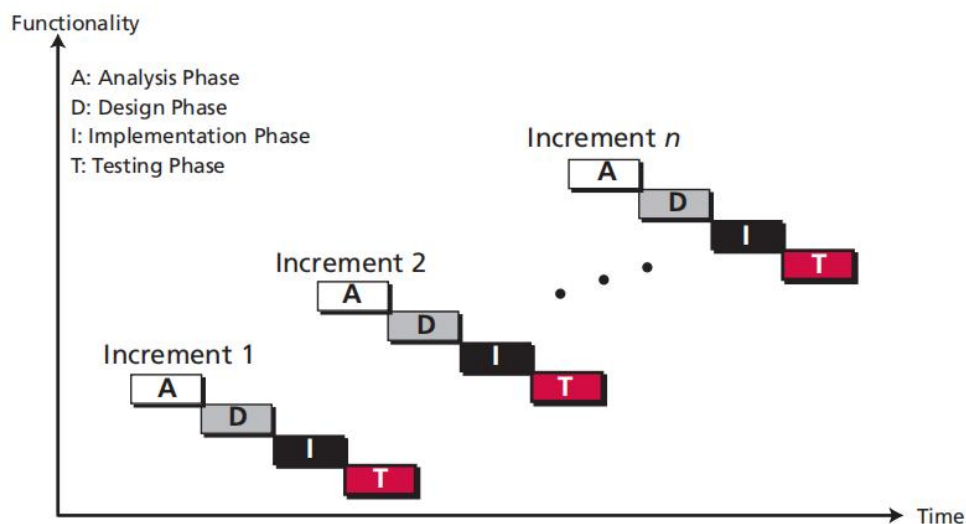


图 2 - 1 增量开发模型

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，显示一些用户可能关心的细节变化。如图 3-1 用例图所示用来：

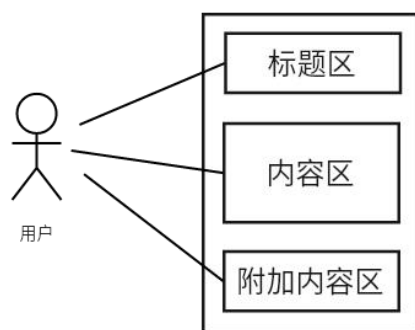


图 3-1 用例图

DOM 树如图 3-2:

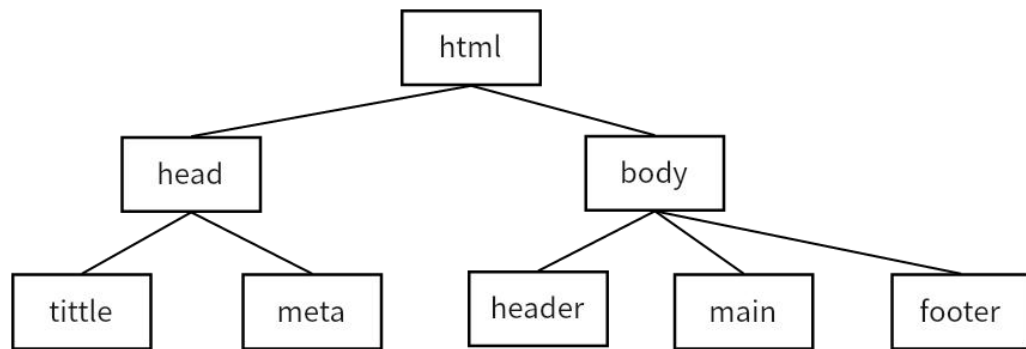


图 3-2 DOM 树图

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 曾令昱 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

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 edge 浏览器打开项目的结果，如下图 3-3 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 3-4 的二维码，运行测试本项目的第一次开发的阶段性效果



图 3 - 3 第一次开发 pc 平台 UI

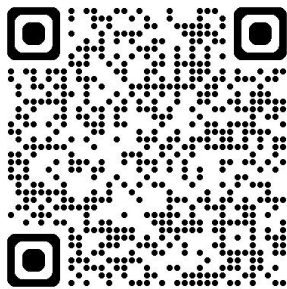


图 3-4 第一版本

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 zly  
$ git config user.email 3405788706@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：

```
ZLY@LAPTOP-PET3M3LF MINGW64 /d/abc/exp (master)
$ git log
commit f9846be3e4bffbcea7517b1b541050278a725ae2 (HEAD -> master, origin/master)
Author: zly <3405788706@qq.com>
Date: Tue Jun 4 10:20:42 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

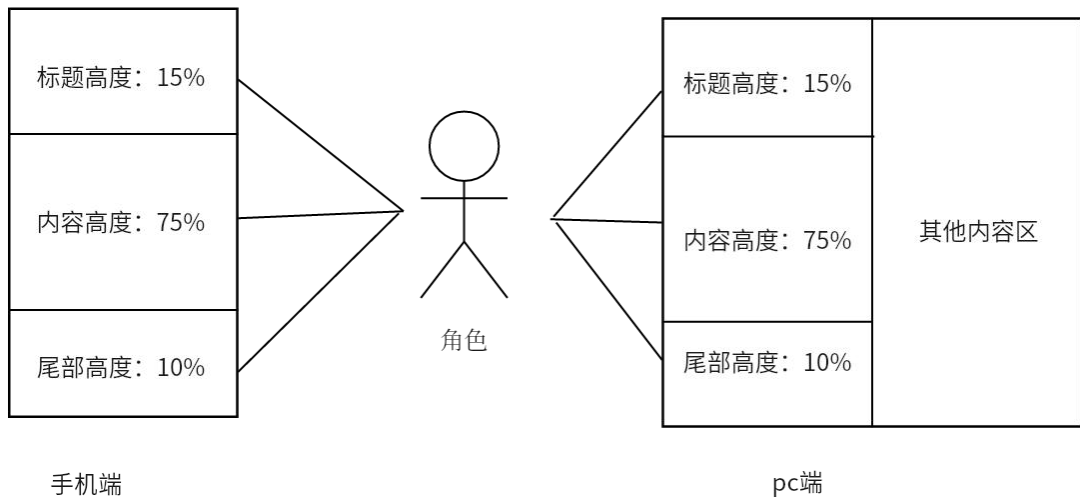
4.1.分析和设计

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^[5]。

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^[5]。

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。如图下 4-1 所示窄屏设计用例图：



4-1 窄屏设计用例图

DOM 树 如图 4-2:

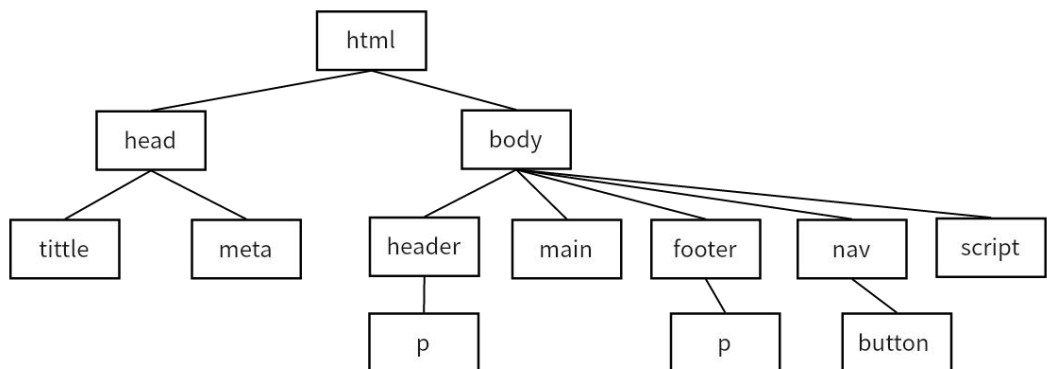


图 4-2 DOM 树图

4.2.项目的实现和编程

main 选择器表示选择网页中的 main 元素, border: 2px solid blue;表示给 main 元素添加了一个蓝色的 2 像素实线边框, height: 70%;表示 main 元素高度占页面高度的 70%, font-size: 1.2em;表示 main 元素的文字大小为 1.2 倍。

nav 选择器表示选择网页中的 nav 元素, border: 2px solid blue;表示给 nav 元素添加了一个蓝色的 2 像素实线边框, height: 10%;表示 nav 元素高度占

页面高度的 10%。

nav button 选择器表示选择 nav 元素内的按钮元素, font-size: 1.1em; 表示 nav 元素内按钮的文字大小为 1.1 倍。

footer 选择器表示选择网页中的 footer 元素, border: 2px solid blue; 表示给 footer 元素添加了一个蓝色的 2 像素实线边框, height: 5%; 表示 footer 元素高度占页面高度的 5%。

给页面中的不同部分 (header、main、nav、footer) 的样式, 包括边框、高度和文字大小等属性, 以实现页面布局 and 美化效果。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 4-3 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 4-3

UI 对象被定义来存储页面的宽度和高度信息。UI.appWidth 设置为窗口宽度大于 600 像素时为 600 像素，否则为窗口宽度。UI.appHeight 设置为窗口高度。

常量 LETTERS 被定义为 22。baseFont 被计算为 UI.appWidth 除以 LETTERS 的结果，用以确定页面字体大小的基准值。

document.body.style.fontSize 被设置为 baseFont + "px"，这样整个页面的字体大小都将被设置为基准值 baseFont。

document.body.style.width 被设置为 UI.appWidth - 2*baseFont + "px"，这会将页面的宽度设置为屏幕宽度减去两倍基准字体大小，从而实现全屏效果。

document.body.style.height 被设置为 UI.appHeight - 4*baseFont + "px"，这会将页面的高度设置为屏幕高度减去四倍基准字体大小，这也有助于实现全屏效果。

通过这些设置，页面的布局和字体大小会根据屏幕尺寸进行动态调整，从而实现响应式设计，确保网页在不同设备上的显示效果良好：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 4-4 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>
```


4.3.项目的运行和测试

本文此处给出了 PC 端用 edge 浏览器打开项目的结果，如下图 4-3 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 4-4 的二维码，运行测试本项目的第二次开发的阶段性效果



图 4 - 5 第二次开发 窄屏设计实现 UI

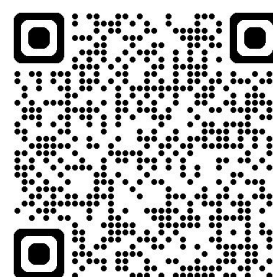


图 4 - 6 第 2 版本二维码

4.4 项目的代码提交

```
$ git add 1.2.html  
$ git commit -m 项目第二版：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：

```
$ git commit -m "项目第二版：窄屏终端的响应式设计"
[main 2a6e9ae] 项目第二版：窄屏终端的响应式设计
1 file changed, 78 insertions(+)
create mode 100644 exp/1.2.html
```

gitbash 反馈代码的仓库日志如下所示：

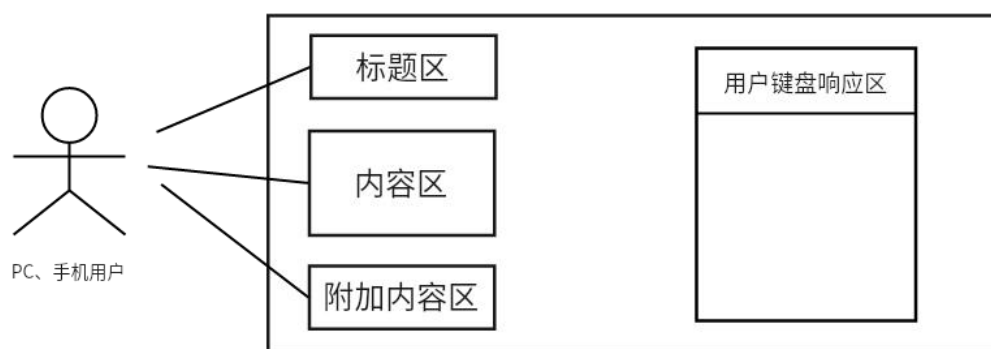
```
$ git log
commit 2a6e9aef43787f0e17513364ca72c59753768bc0 (HEAD -> main)
Author: zly <3405788706@qq.com>
Date:   Wed Jun 5 10:57:02 2024 +0800
```

项目第二版：窄屏终端的响应式设计

5.宽屏和窄屏通用的响应式设计和代码实现

5.1.分析和设计

根据屏幕大小自动调整页面的字体大小和布局，以实现响应式设计的效果。通过设定基准字体大小和页面宽高，以及根据浏览器窗口尺寸调整布局，确保页面在不同设备上都能有良好的显示效果。如图下 5-1 所示宽屏和窄屏通用的响应式设计用例图：



5-1 窄屏设计用例图

DOM 树 如图 5-2:

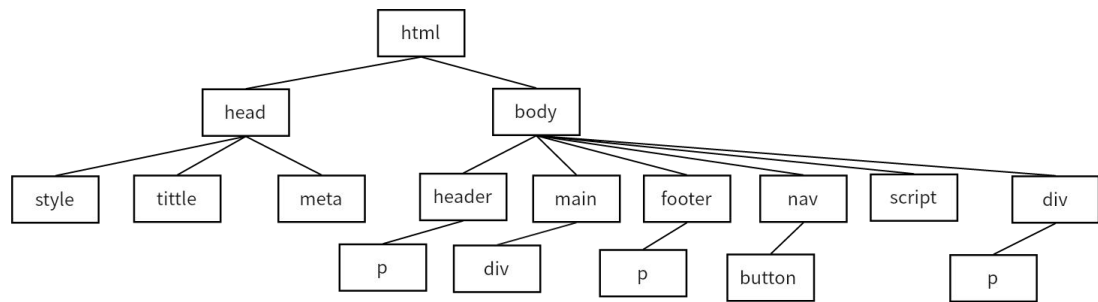


图 5-2 DOM 树图

5.2.项目的实现和编程

这一步中我们设置了一些 UI 相关的变量,计算出合适的字体大小使页面自适应不同设备宽度。通过设置适应设备宽度的字体大小、元素宽度和高度,实现了简单的响应式设计效果当窗口宽度小于 900px 时,隐藏了 `<div id="aid">` 并且给 `<div id="bookface">`。如代码块 5-1 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';
```

代码块 5-3

5.3.项目的运行和测试

本文此处给出了 PC 端用 edge 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三次开发的阶段性效果



图 5 - 4 第三次开发 pc 平台 UI



图 5 - 5 第三次开发 手机平台 UI

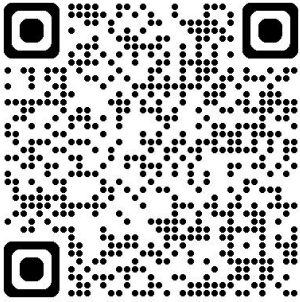


图 5 - 6 第 3 版二维码

5.4.项目的代码提交

```
$ git add 1.3.html
$ git commit -m 项目第三版: 宽屏和窄屏通用的响应式设计和代码实现
```

成功提交代码后，gitbash 的反馈如下所示：

```
$ git commit -m "宽屏和窄屏通用的响应式设计和代码实现"
[main 6ef728f] 宽屏和窄屏通用的响应式设计和代码实现
1 file changed, 160 insertions(+)
create mode 100644 exp/1.3.html
```

gitbash 反馈代码的仓库日志如下所示：

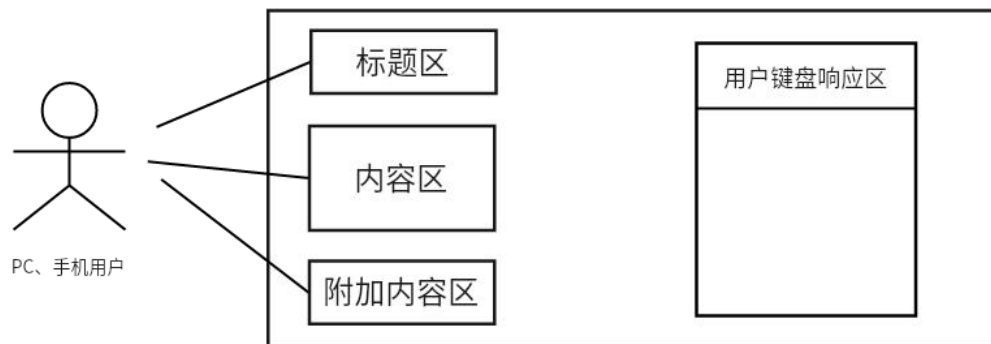
```
$ git log
commit 6ef728fe87d8f13fed1e68278d6252d814e15f87 (HEAD -> main)
Author: zly <3405788706@qq.com>
Date: Wed Jun 5 11:36:14 2024 +0800
```

宽屏和窄屏通用的响应式设计和代码实现

6. 个性化 UI 设计中鼠标模型

6.1. 分析和设计

实现基本的鼠标交互控制，当用户按下鼠标拖动时，可以在页面上实时显示鼠标的移动距离，并根据移动距离判断是否为有效拖动。这段代码相比前几段代码更专注于鼠标交互的设计与控制，通过监听相应事件实现了拖动效果。如图下 6-1 所示宽屏和窄屏通用的响应式设计用例图：



6-1 窄屏设计用例图

DOM 树 如图 6-2:

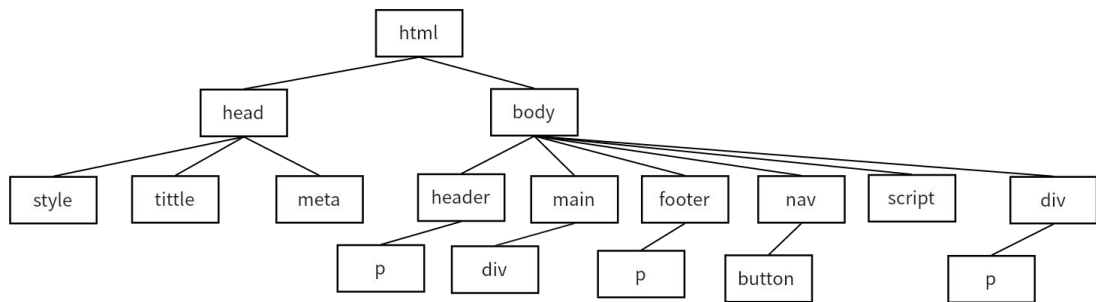


图 6-2 DOM 树图

6.2.项目的实现和编程

定义了一个 `mouse` 对象，包括 `isDown`（鼠标是否按下）、`x`（鼠标点击位置的 `x` 坐标）、`y`（鼠标点击位置的 `y` 坐标）和 `deltaX`（鼠标移动的水平距离）属性；监听了鼠标的 `mousedown`（鼠标按下）、`mouseup`（鼠标松开）、`mouseout`（鼠标移出区域）和 `mousemove`（鼠标移动）事件。

在 `mousedown` 事件中，记录了鼠标点击时的坐标，并更新了界面显示；在 `mouseup` 和 `mouseout` 事件中，判断鼠标移动的距离是否超过 100 像素，如果是则认为是有效拖动，否则认为是无效拖动，并更新界面显示；在 `mousemove` 事件中，如果鼠标处于按下状态，计算鼠标水平移动的距离并更新界面显示。如代码块 6-3 所示：

```

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;
});

```

代码块 6-3

6.3.项目的运行和测试

本文此处给出了 PC 端用 edge 浏览器打开项目的结果，如下图 6-2 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四次开发的阶段性效果



图 6 - 4 第四次开发 pc 平台 UI

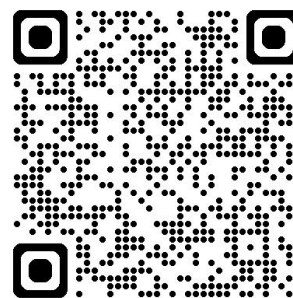


图 6 - 5 第 4 版二维码

6.4.项目的代码提交

```

$ git add 1.4.html
$ git commit -m 项目第四版：个性化 UI 设计中鼠标模型

```

成功提交代码后，gitbash 的反馈如下所示：


```
$ git commit -m "项目第四版：个性化UI设计中鼠标模型"
[main 59c2a7c] 项目第四版：个性化UI设计中鼠标模型
1 file changed, 191 insertions(+)
create mode 100644 exp/1.4.html
```

gitbash 反馈代码的仓库日志如下所示：

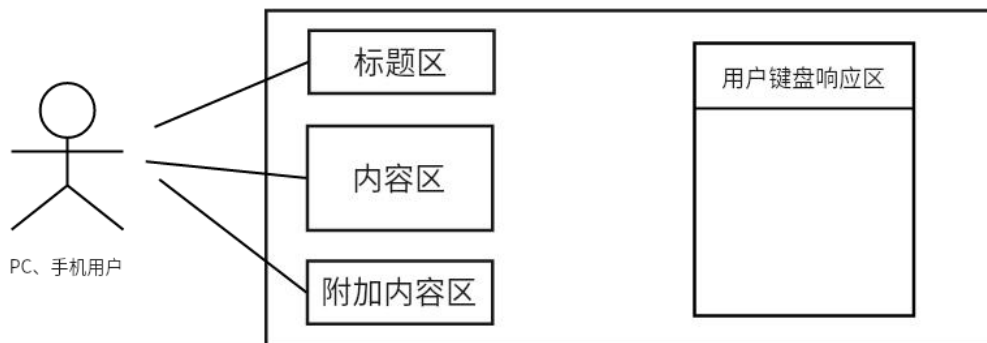
```
$ git log
commit 59c2a7c8dd7b9636e8cc16bebedc1c2841fa6f54 (HEAD -> main)
Author: zly <3405788706@qq.com>
Date: Wed Jun 5 12:25:39 2024 +0800
```

项目第四版：个性化UI设计中鼠标模型

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

实现了对鼠标、触摸和键盘事件的监听和处理，通过不同事件触发的操作更新页面显示，实现更丰富的用户交互体验；同时也实现对传入参数的判断和异常处理。如图下 7-1 所示对触屏和鼠标的通用交互操作的设计用例图：



7-1 窄屏设计用例图

DOM 树 如图 7-2:

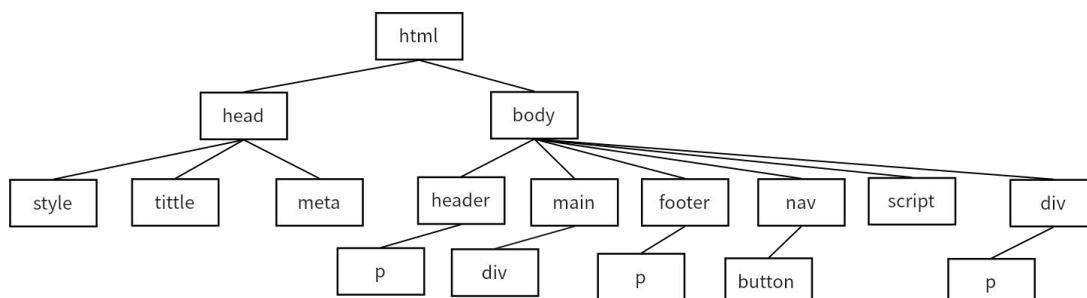


图 7-2 DOM 树图

7.2.项目的实现和编程

定义了一个名为 Pointer 的对象，包括 isDown（鼠标或触摸是否按下）、x（点击位置的 x 坐标）、deltaX（水平移动的距离）等属性，并定义了三个处理函数 handleBegin、handleEnd 和 handleMoving

handleBegin 函数用于处理鼠标或触摸事件的开始，根据事件类型判断是鼠标事件还是触摸事件，记录相应的坐标信息，并更新界面显示

handleEnd 函数用于处理鼠标或触摸事件的结束，根据事件类型进行不同的处理逻辑，判断移动距离是否超过 100 像素，并更新界面显示

handleMoving 函数用于处理鼠标或触摸事件的移动，在移动过程中计算移动距离并更新界面显示

添加了相关事件的监听器，分别监听 mousedown、touchstart、mouseup、touchend、mouseout、mousemove、touchmove 事件，并将事件处理交给相应的处理函数；监听了 body 元素的 keypress 事件，在用户输入字符时将字符添加到页面指定区域；实现了一个自定义的\$函数，用于根据传入的参数获取对应的元素节点，如果元素存在则返回该元素，否则抛出异常。如代码块 7-3 所示：

```
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 7-3

7.3.项目的运行和测试

本文此处给出了 PC 端用 edge 浏览器打开项目的结果，如下图 7-2 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 7-3 的二维码，运行测试本项目的第五次开发的阶段性效果



图 7 - 4 第五次开发 pc 平台 UI

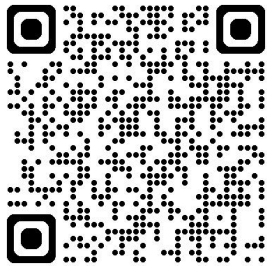


图 7 - 5 第 5 版二维码

7.4.项目的代码提交

```
$ git add 1.5.html
$ git commit -m 项目第五版: 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：

```
$ git commit -m "项目第五版: 对触屏和鼠标的通用交互操作的设计开发"
[main 9c79181] 项目第五版: 对触屏和鼠标的通用交互操作的设计开发
1 file changed, 216 insertions(+)
create mode 100644 exp/1.5.html
```

gitbash 反馈代码的仓库日志如下所示：

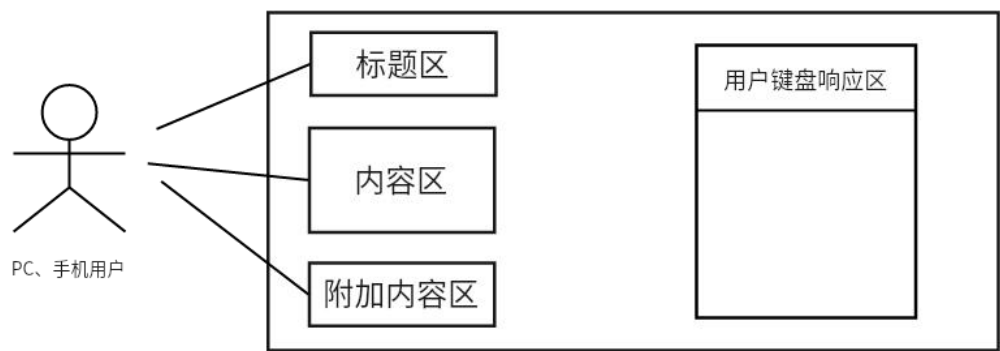
```
$ git log
commit 9c79181df1150e1190f0bce556a449981b5596d6 (HEAD -> main)
Author: zly <3405788706@qq.com>
Date: Wed Jun 5 12:37:08 2024 +0800
```

项目第五版: 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

利用 keydown 和 keyup 事件实现了按键状态和文本内容的同时输出，并通过 printLetter 函数对按键进行判断，进一步优化用户输入体验。可以在 printLetter 函数中继续优化处理连续空格和制表键的逻辑，以提升用户交互的效果。如图下 8-1 所示 UI 的个性化键盘交互控制的用例图：



8-1 窄屏设计用例图

DOM 树 如图 8-2:

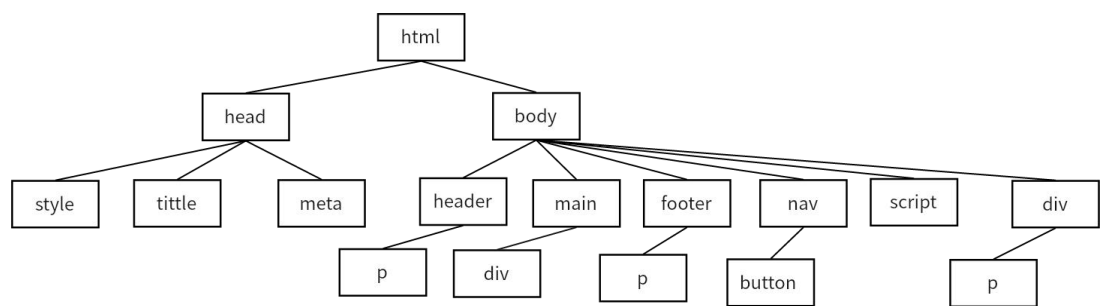


图 8-2 DOM 树图

8.2.项目的实现和编程

keydown 事件监听器函数中，通过事件对象 ev 获取按下的键值 k 和字符编码 c，并通过 DOM 操作将按键状态和字符编码信息显示在页面中。

keyup 事件监听器函数中，同样获取松开的键值 key 和字符编码 code，并将这些信息显示在页面中。此外，在松开按键时，根据 printLetter 函数的逻辑判断，如果按键符合字母、数字或特定符号，则将按键内容追加到页面中的 typeText 元素中。

printLetter 函数用于判断按键是否为字母、数字或特定符号，并根据条件返回 true 或 false。若按键为字母、数字或特定符号，则返回 true；否则返回 false。在判断过程中，特定符号存储在数组 puncs 中，用于区分符号和其他按键。

在处理连续空格和制表键 tab 等情况时，可以进一步优化 printLetter 函数的逻辑，特别处理这些情况，例如判断连续空格并只追加一个空格字符，或将制表键 tab 转换为相应的空格和缩进符号。如下图所示 8-3：

```
$("#body").addEventListener("keydown",function(ev){
    ev.preventDefault() ;
    let k = ev.key;
    let c = ev.keyCode;
    $("#keyboard").textContent = "您已按键 : " + k + " , "+ "字符编码 : " + c;
});
$("#body").addEventListener("keyup",function(ev){
    ev.preventDefault() ;
    let key = ev.key;
    let code = ev.keyCode;
    $("#keyboard").textContent = "松开按键 : " + key + " , "+ "字符编码 : " + code;
    if (printLetter(key)){
        $("#typeText").textContent += key ;
    }
    function printLetter(k){
        if (k.length > 1){ //学生必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 8-3

8.3.项目的运行和测试

本文此处给出了 PC 端用 edge 浏览器打开项目的结果，如下图 8-2 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 8-3 的二维码，运行测试本项目的第六次开发的阶段性效果



图 8-4 第六次 pc 平台 UI

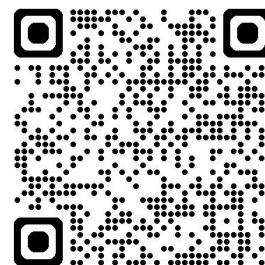


图 8-5 第六版二维码

8.4.项目的代码提交

```
$ git add 1.6.html
$ git commit -m 项目第六版: 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:

```
$ git commit -m "项目第六版: UI的个性化键盘交互控制的设计开发"
[main 2761b67] 项目第六版: UI的个性化键盘交互控制的设计开发
1 file changed, 266 insertions(+)
create mode 100644 exp/1.6.html
```

gitbash 反馈代码的仓库日志如下所示:

```
$ git log
commit 2761b675ec4d9246d76685c36b650a900954c365 (HEAD -> main)
Author: zly <3405788706@qq.com>
Date:   Wed Jun 5 12:41:12 2024 +0800
```

项目第六版: UI的个性化键盘交互控制的设计开发

9.用 git 工具展开代码的版本管理和发布

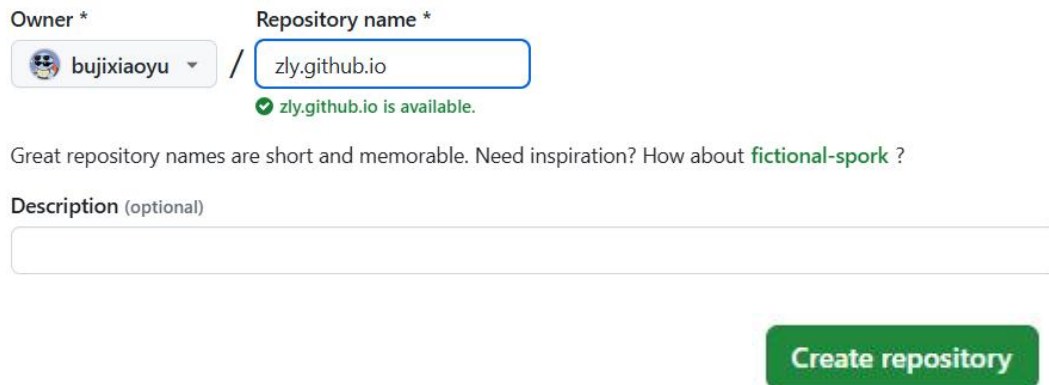
9.1.经典 Bash 工具介绍

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^[6]。

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^[6]。

9.2.通过 gitHub 平台实现本项目的全球域名

9.3.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。

9.4.设置本地仓库和远程代码仓库的链接跨世纪的经典 Bash 工具

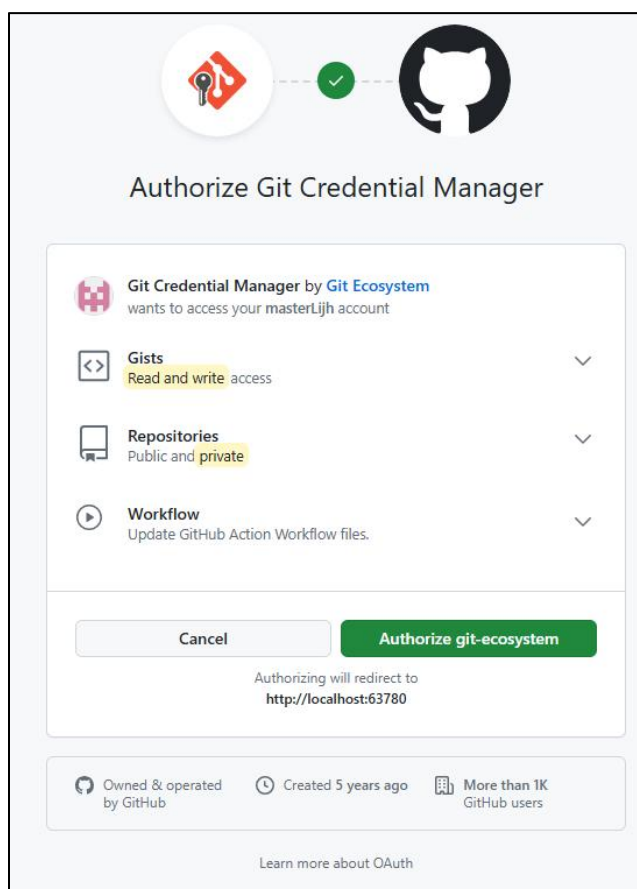
进入本地 webUI 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接

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

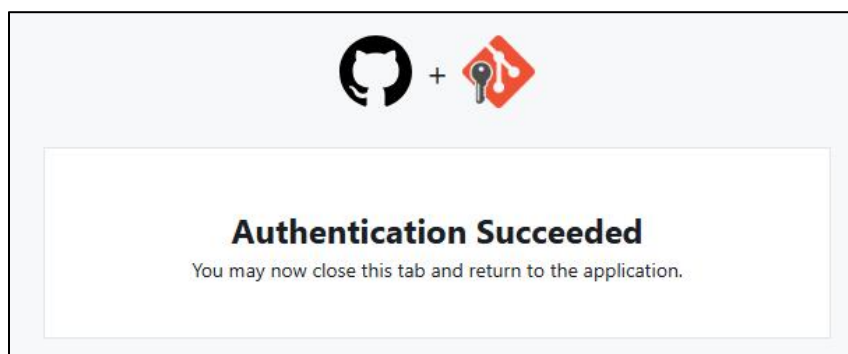
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



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：



10. 谈谈本项目中的高质量代码

在这个项目中有一些相对高质量的代码。如代码块 10-1:

```
function printLetter(k) {  
    if (k.length > 1) {  
        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;
    }
}

```

代码块 10-1

在这段代码中，我们首先定义了一个名为 `printLetter` 的函数，用于判断用户输入的按键是否是有效的字符或标点符号。这个函数接受一个字符作为参数，并返回一个布尔值，表示该字符是否应该被输出到页面上。

在函数中，我们首先检查参数 `k` 的长度是否大于 1。如果大于 1，说明用户输入的是一个组合键或特殊字符，不应该被输出，因此返回 `false`。

然后，我们定义了一个包含各种标点符号和空格的数组 `puncs`。接着使用条件语句判断参数 `k` 是否在字母、数字或标点符号数组 `puncs` 中。如果是字母、数字或特殊字符，则输出 "letters" 或 "puncs" 到控制台，并返回 `true`，表示该字符应该被输出到页面上。

最后，如果参数 `k` 不是字母、数字或特殊字符，则返回 `false`，表示该字符不应该被输出。

值得注意的是，函数中还提出了一个更高阶的问题，即如何处理连续的空格和制表键 `tab`。这可能需要进一步的逻辑处理，例如在函数中添加额外的条件判断来处理连续的空格或 `tab` 键的情况。

参考文献

- [1] Behrouz Forouzan. Foundations of Computer Science[M](4th Edition). Cengage Learning EMEA,2018: 274--275
- [2] W3C. W3C's history. W3C Community. [EB/OL]. <https://www.w3.org/about/>. 2023.12.20
- [3] Douglas E. Comer. The Internet Book [M] (Fifth Edition). CRC Press Taylor & Francis Group, 2019: 217-218
- [4] John Dean,PhD. Web programming with HTML5,CSS,and JavaScript[M]. Jones & Bartlett Learning,LLC. 2019: 2
- [5] John Dean,PhD. Web programming with HTML5,CSS,and JavaScript[M]. Jones & Bartlett Learning,LLC. 2019: xi
- [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
- [8] Martina Seidl, Marion Scholz, et al. UML @ Classroom An Introduction to Object-Oriented Modeling [M]. Springer International Publishing Switzerland, 2015
- [9] Matti Tedre, Peter J.Denning. Computational Thinking, A Professional and Historical Perspective. Computational Thinking in Education A Pedagogical Perspective[C]. Routledge Taylor & Francis Group, 2022:1-17