

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）：WebclientbasedcustomizedUIDesignandProgramming

院（系）：元宇宙产业学院

专业：计算机科学与技术

学生姓名：唐恒彬

学号：20213582

指导教师：李建宏

2024 年 06 月 19 日

目录

基于 Web 客户端技术的个性化 UI 设计和实现	1
1 引言	1
1.1 项目概述	1
1.2 研学计划	1
1.3 研学方法	2
2.Web 平台和客户端技术概述	3
2.1 网络平台和网络编程	3
3 项目的增量式迭代开发模式	3
3.1 瀑布模型	4
3.2 软件开发的过程管理——增量式开发模式	4
4 内容设计概要	5
4.1 本项目的总体大纲介绍	5
4.2 本项目功能的具体实现	6
4.3 关于本项目的总结	8
5 移动互联时代的响应式设计和窄屏代码实现	9
5.1 分析与设计	9
5.2 项目的实现和编程	9
5.3 项目的运行和测试	11
5.4 项目的代码提交和版本管理	12
6 宽屏和窄屏通用的响应式设计和代码实现	13
6.1 分析与设计	13
6.2 项目的实现和编程	14
6.3 项目的运行和测试	17
6.4 项目的代码提交和版本管理	18
7 个性化 UI 设计中鼠标模型	18
7.1 分析与设计	18
7.2 项目的实现和编程	21
7.3 项目的运行和测试	24

7.4 项目的代码提交和版本管理	25
8 通用的 UI 设计，用一套代码同时为触屏和鼠标建模	26
8.1 分析与设计	26
8.2 项目的实现和编程	30
8.3 项目的运行和测试	34
8.4 项目的代码提交和版本管理	35
9. UI 的个性化键盘控制——应用 keydown 和 keyup 键盘底层事件	36
9.1 分析与设计	36
9.2 项目的实现和编程	37
9.3 项目的运行和测试	43
9.4 项目的代码提交和版本管理	44
10.谈谈本项目中的高质量代码	44
11.用 git 工具展开代码的版本管理和发布	47
11.1 经典 Bash 工具介绍	47
11.2 通过 gitHub 平台实现本项目的全球域名	48
11.3 创建一个空的远程代码仓库	48
11.4 设置本地仓库和远程代码仓库的链接	49
全文完成，谢谢！	51
参考文献	51

基于 Web 客户端技术的个性化 UI 设计和实现

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，它表现为三种形式，分别是超文本（hypertext）、超媒体（hypermedia）和超文本传输协议（http），本项目以 web 客户端技术为研究学习对象，探究了 HTML 内容建模、CSS 样式设计和 JavaScript 功能编程的基本技术技巧。实现了一个个性化的设计界面（UI： user interface），该用户界面可以适用于 PC 端和移动设备。以 DOM 技术和事件驱动模式的程序为支撑做到了最佳适配用户屏幕，即可以随着用户的界面变化而调整。实现了对鼠标、触屏、键盘的底层事件的支持和流畅的响应。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，用工程思想管理好项目，本人使用了软件工程的增量式增强的开发模式，一共做了 6 次 ADI（A:分析 D:设计 I:实现 T:测试）开发，逐步实现了整个 UI 应用的编写。为了与互联网上的同行共享代码，共同工作，本项目还使用了 git 工具进行代码和开发过程日志记录，项目一共做了 6 次提交代码的操作，详细记录和展示了开发思路。最后通过 git bash 把项目上传到 git hub 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。

关键字：Web；HTML；CSS，JavaScript；git；增量开发模式

1 引言

1.1 项目概述

本设计是结合本学科的核心课程知识，实现了一个个性化的基于 web 客户端技术的响应式用户界面(UI)，该 UI 能够较好的适配各种屏幕。在功能上，通过 DOM 技术和事件监听实现了对鼠标、触屏和键盘的底层事件的支持并实现了流畅地响应。基于面向对象的思想，为鼠标和触屏设计了对象模型，并通过代码实现了对这类指向性设备的模拟。

本论文是针对该项目做的介绍，包括运用该项目的技术，该项目的需求分析，该项目的迭代与更新。完成对该项目建立模型、软件设计、系统实施、测试调试一系列的软件开发流程的介绍后，对建立模型的方法对触屏，鼠标和键盘事件进行探讨，分析其区别、联系与实现方法。

1.2 研学计划

表 1 研学计划表

时间	操作
第一周	用三段论的方式完成了页面的设计与搭建
第二周	给页面添加了图片和导航按钮，完善页面
第三周	在页面的主区域中设计了鼠标移动事件，为 PC 端添加了键盘响应区，能够根据键盘按键作出响应，为后续的键盘事件做好铺垫。
第四周	继续探讨鼠标事件的应用，在页面的主区域添加了鼠标拖动事件。
第五周	在页面主区域为移动端添加了触屏拖动事件，为移动端用户提供方便。并在键盘响应区添加了键盘的监听事件，监听键盘的按下和抬起。实现了用同一套代码为触屏和鼠标事件建模
第六周	优化了键盘的监听事件，阻止用户按下键盘的功能键，优化了显示效果。

1.3 研学方法

为了确保论文的严谨性和科学性，本论文采用了 4 种研究方法。首先采用了文献综述法，通过查询课外电子书籍等相关资料文献，了解 HTML、CSS 和 JavaScript 最新标准。这一过程不仅为我们提供了坚实的理论基础，还帮助本文明确了研究方向和技术创新点。其次，采用增量开发策略将整体开发工作细分为多个小周期或迭代。每个迭代中，本研究专注于设计、实现，这种方法允许我们快速验证想法，及时调整方向，并确保每个功能的稳定性和可靠性，从而有效提升了开发效率和项目管理的灵活性。之后采用 Git 作为版本控制系统，本研究对每一次代码提交都进行了详细的记录和管理。这不仅便于团队成员间的协作，还使得代码的历史变更可追溯，错误回滚迅速，保证了代码库的整洁和项目的可持续发展。通过 GitHub 平台，我们实现了代码的版本控制、分支管理及合并请求，促进了代码审查和知识共享，减少了后期维护成本，确保了软件项目的健壮性和可维护性。

2.Web 平台和客户端技术概述

2.1 网络平台和网络编程

Web 之父 TimBernersLee 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]。

Web 平台和客户端技术是构建现代互联网应用不可或缺的组成部分，它们共同支撑了丰富的用户体验和功能实现。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。

Web 是一个文档的集合，被称为网页，它们由世界各地的计算机用户（在大部分时间内）共享。不同类型的网页可以做不同的事情，但至少，它们都能在电脑屏幕上显示内容。我们所说的“内容”是指文本、图片和用户输入机制，如文本框和按钮^[1]。

web 编程：Web 编程是一个很大的领域，通过不同的工具实现不同类型的 Web 编程。所有的工具都使用核心语言 HTML 来工作，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，所有的深入。这三种技术被认为是客户端网络编程的支柱。使用客户端 web 编程，所有网页计算都在终端用户的计算机（客户端计算机）上执行。

3 项目的增量式迭代开发模式

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)。所以本研究讨论最常见的两种：瀑布模型和增量模型。

3.1 瀑布模型

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。

例如，整个项目的分析阶段应在其设计阶段开始之前完成。整个设计阶段应在开始实施阶段之前完成。

瀑布模型既有优点也有缺点。其中一个优点是，每个阶段都在下一个阶段开始之前完成。例如，在设计阶段工作的小组确切地知道该做什么，因为他们拥有分析阶段的完整结果。测试阶段可以测试整个系统，因为正在开发的整个系统已经准备好了。然而，瀑布模型的一个缺点是难以定位问题：如果在部分过程中存在问题，则必须检查整个过程的。

3.2 软件开发的过程管理——增量式开发模式

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，如下图 2-1 所示

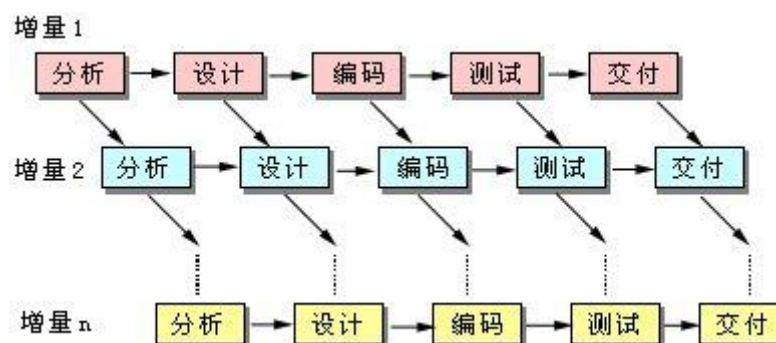


图 2-1

增量开发模式是软件开发的主流方法之一，它将一个软件开发项目分解为众多小的、可管理的部分，称为“增量”。这些增量代表了软件的不同部分或功能，

可以独立开发、测试和部署。其开发的主要特征和步骤如下：

项目规划：在项目开始时，将整体需求分析为多个模块或功能集。根据优先级和业务价值，为这些模块分配顺序。迭代开发：每个迭代专注于开发一个或多个增量。迭代周期通常较短，可能从几周到几个月不等，取决于增量的复杂性和规模。在每个迭代结束时，一个完整的工作软件版本被创建，即使它可能只包含部分功能。模块化开发：开发团队专注于一个或几个增量，允许团队成员并行工作，各自负责不同增量的开发。集成：随着每个增量的完成，它们被集成到现有的软件系统中。测试与反馈：完成的增量经过单元测试、集成测试和系统测试。用户或利益相关者可以对每个增量提供反馈，关系到后续增量的开发。风险管理：增量式开发降低了整体风险，因为如果一个增量出现问题，只会影响部分功能，而不是整个系统。更早地获得反馈，可以更快地识别和解决潜在问题。灵活适应变化：需求变化或优先级调整可以更容易地在下一个增量中处理，而不必重新设计整个系统。资源优化：开发资源可以根据增量的重要性和紧迫性进行分配，优先处理关键功能。

在第二个版本中，添加了更多的细节，而一些没有完成，系统再次测试。如果有问题，开发人员就知道问题在于新功能。在现有的系统正常工作之前，它们不会添加更多的功能。

4 内容设计概要

4.1 本项目的总体大纲介绍

本项目主要运用 CSS，HTML，和 JavaScript 技术，建立一个可供窄屏和宽屏用户共同使用的网页。该网页对宽屏用户实现了对鼠标的移动进行响应，同时也实现了对键盘的敲击进行响应，可输出键盘对应的键。本项目意在理解 HTML 底层的元素之间的关系以及各元素对应事件的使用。具体情况如图 1，图 1 所示：

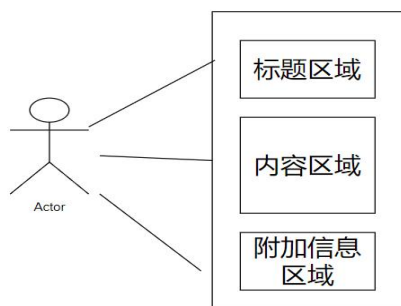


图 1

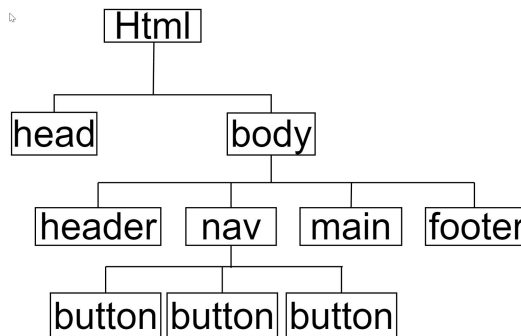


图 2

4.2 本项目功能的具体实现

(1) 本项目的第一次设计概要：1.用 THTML 语言实现了内容模型的建立，把应用分别分成了 header, main nav 和 footer 三个部分。2.用 CSS 语言实现了项目的大致 UI 外观。3.初步完成了软件的架构设计，lesson 文件夹和 myface 文件夹分别放项目的图片和资源。第一次设计的网页实际为一个导航栏，以后对于该项目的每次迭代都可以根据该导航栏进行查看。其效果图如图 3- 1 所示：



图 3- 1 导航栏效果图

(2) 本项目的第二次设计概要：本次代码提交初步完成了响应式设计。代码实施分以下几步：1.为软件 4 个区域分配了高度的比率，通过把设备的高度全部分配给了 body 对象，结合之前的高度分配，实现各区域的响应式设计。使得可以对用户的窄屏和宽屏作出响应2.为应用计算了基础字体的大小，并利用 body 的遗传机制，结合 CSS 的字体的相对控制，实现了字体的响应式设计。其效果图如图 3- 2，图 3- 3 所示：

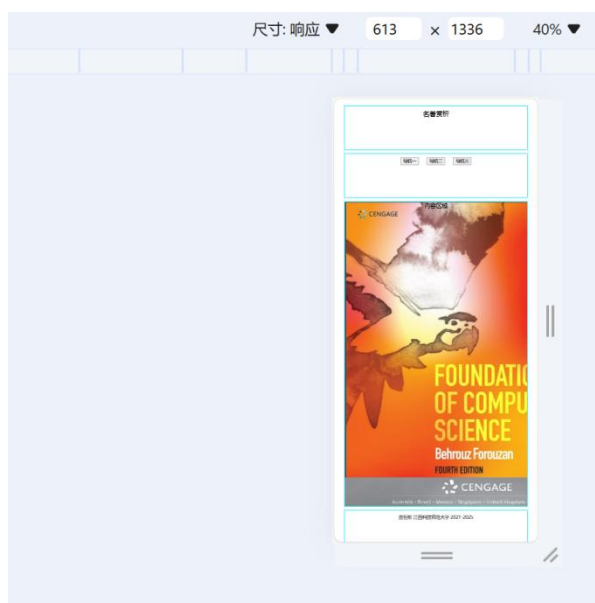


图 3-2 用户窄屏效果图



图 3-3 用户宽屏效果图

(3) 本项目的第三次设计概要: 本次代码提交完成了鼠标对 UI 的初步控制, 包括提交两个文件 1.3.html、1.4.html。其中 1.3.html, 对屏幕超过 1000 像素的用户屏幕增加了一个用于键盘反馈的 UI 界面, 还对主区域书的封面元素设置了鼠标按下、鼠标移动和鼠标抬起做了事件响应。在 1.4.html 内增加了一个鼠标模型 Mouse 用来模拟鼠标的操作数据, 并根据鼠标模型和用户操作设计了 UI 反馈。另外增加了底层函数\$用于快速抓取页面元素对象, 在代码中已经反复使用\$函数。其效果图如图 3-4, 图 3-5 所示:

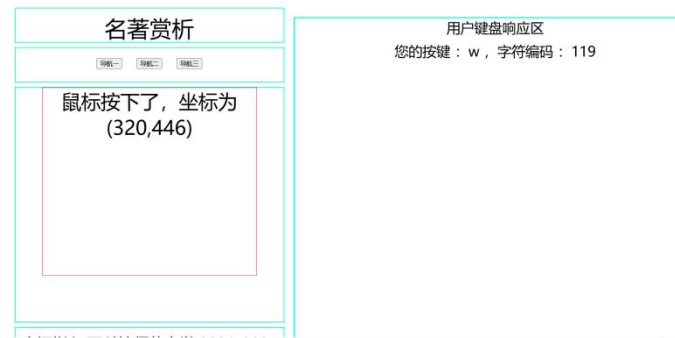


图 3-4 1.3.THML 效果图



图 3-5 1.4.HTML 效果图

(4) 本项目的第四次设计概要：通过对页面各个部分所占百分比的方式完成响应式设计（宽屏和窄屏），对于 UI 设计之鼠标模型的控制以及键盘响应，本项目采用事件监听的方式，监听到需要对鼠标或键盘作出响应的操作之后，底层函数\$快速抓取页面元素对象，对该页面对象作出处理。

(5) 本项目的第五次设计概要：本次迭代位鼠标模型和键盘模型设计了通用的 UI，用一套代码同时为触屏和鼠标建模。减少了代码的冗杂性，使代码的复用性更高，同时便于后期代码的维护。

4.3 关于本项目的总结

经过本阶段的学习，本人对于 CSS 和 JavaScript 的基本代码有一定的了解，同时经过对鼠标事件、触屏事件以及键盘事件的研究对响应式设计有了一定的理解。

响应式设计的目标是消除“移动版”和“桌面版”网站之间的差异，提供单一的 URL 和内容源，简化网站管理和维护，同时也为用户提供一致的浏览体验，无论们使用的是桌面电脑、平板还是智能手机。

5 移动互联时代的响应式设计和窄屏代码实现

5.1 分析与设计

首先，创建了一个名为 UI 的对象，并设置了两个属性：UI.appWidth：存储了浏览器窗口的内部宽度，即 window.innerWidth，UI.appHeight：存储了浏览器窗口的内部高度，即 window.innerHeight。

计算基础字体大小 baseFont：将 appWidth 除以 20，得到的结果作为字体大小。这样可以根据窗口宽度动态调整文本的相对大小，实现响应式设计。

设置 body 元素的字体大小：将 baseFont 转换为带有单位的字符串 ('px')，并将其赋值给 document.body.style.fontSize。这样，body 及其所有子元素的字体大小都会根据窗口宽度自动调整。

设置 body 元素的高度：将 UI.appHeight 减去 25（可能是为了排除某些边距或标题栏的高度），然后加上 'px' 单位，赋值给 document.body.style.height。这使得 body 元素的高度与浏览器窗口高度相同，实现全屏效果。

通过以上步骤，页面布局和样式会根据窗口尺寸的变化进行自适应调整，适合不同设备和屏幕尺寸。

用例图以及 Dom 树如图 4-1 图 4-1

图 4-2 所示：

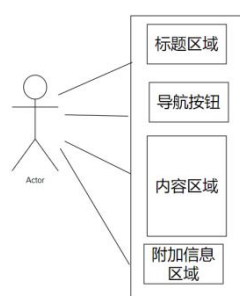


图 4-1

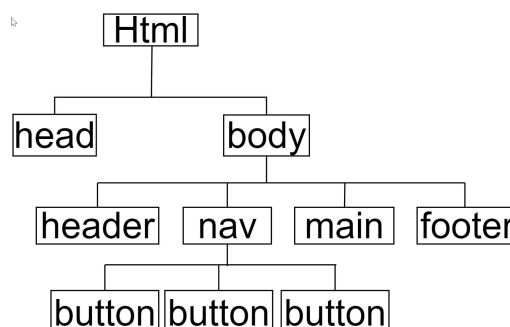


图 4-2

5.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
```

```

<p>
名著赏析
</p>
</header>
<nav>
<button>导航一</button>
<button>导航二</button>
<button>导航三</button>
</nav>
<main id="main">
内容区域
</main>
<footer>
<p id="statusInfo">
唐恒彬 江西科技师范大学 2021-2025
</p>
</footer>

```

二、CSS 代码编写如下：

```

<style>
*{
    margin: 10px;
    text-align: center;
}
header{
    border: 2px solid rgb(0, 234, 255);
    height: 10%;
    /* 相对字体大小 */
    font-size: 1.6em;
}
nav{
    border: 2px solid rgb(0, 234, 255);
    height: 10%;
    font-size: 1.2em;
}
#main{
    border: 2px solid rgb(0, 234, 255);
    height: 70%;
    font-size: 1.4em;
    background-image: url(../lesson/CS.jpg);
    background-size: cover;
}
footer{
    border: 2px solid rgb(0, 234, 255);

```

```

        height: 10%;
        font-size: 1.1em;
    }
</style>

```

三、JavaScript 代码编写如下：

```

var UI = {};
UI.appWidth = window.innerWidth;
UI.appHeight = window.innerHeight;
let baseFont = UI.appWidth / 50;
//因为我们改变了 body 对象的字体大小，这个属性会遗传给 body 的子子孙孙，
//形成自己的响应式设计
document.body.style.fontSize = baseFont + 'px';
//通过把 window 的高度设置为 body（屏幕）的高度，实现全屏。
//通过 CSS 代码对 body 子对象的百分比分配，从而实现响应式设计的目标
document.body.style.height = UI.appHeight - 25 + 'px';

```

5.3 项目的运行和测试

效果图和二维码如图 4-3，图 4-4 下所示：



图 4-3



图 4-4

5.4 项目的代码提交和版本管理

编写好这部分代码且测试成功后，执行下面命令提交代码：

```
$ git add . #添加所有修改过的文件
```

```
$ git commit -m 本项目的第一次设计概要，完成了以下事情：1 用 HTML 语言实现了内容模型的建立，分别把应用分成了 header、main 和 footer 三个部分。2 用 css 语言实现了项目大致的 UI 外观。3 初步完成了软件的架构设计，index.html 文件是主程序，mycss.css 样式文件是整个软件的外观，myJs.js 文件用于整个软件的工程设计。lesson 文件夹和 myface 文件夹分别放项目的各种图片与资源。# 添加提交的信息
```

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git add 1.2.html

Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git commit -m'首先，创建了一个名为UI的对象，并设置了两个属性：UI.appwidth：存储了浏览器窗口的内部宽度，即window.innerWidth，UI.appHeight：存储了浏览器窗口的内部高度，即window.innerHeight。计算基础字体大小baseFont：将appwidth除以20，得到的结果作为字体大小。这样可以根据窗口宽度动态调整文本的相对大小，实现响应式设计。设置body元素的字体大小：将baseFont转换为带有单位的字符串（'px'），并将其赋值给document.body.style.fontSize。这样，body及其所有子元素的字体大小都会根据窗口宽度自动调整。设置body元素的高度：将UI.appHeight减去25（可能是为了排除某些边距或标题栏的高度），然后加上px单位，赋值给document.body.style.height。这使得body元素的高度与浏览器窗口高度相同，实现全屏效果。'
[master 4d20b1c] 首先，创建了一个名为UI的对象，并设置了两个属性：UI.appwidth：存储了浏览器窗口的内部宽度，即window.innerWidth，UI.appHeight：存储了浏览器窗口的内部高度，即window.innerHeight。计算基础字体大小baseFont：将appwidth除以20，得到的结果作为字体大小。这样可以根据窗口宽度动态调整文本的相对大小，实现响应式设计。设置body元素的字体大小：将baseFont转换为带有单位的字符串（px），并将其赋值给document.body.style.fontSize。这样，body及其所有子元素的字体大小都会根据窗口宽度自动调整。设置body元素的高度：将UI.appHeight减去25（可能是为了排除某些边距或标题栏的高度），然后加上px单位，赋值给document.body.style.height。这使得body元素的高度与浏览器窗口高度相同，实现全屏效果。
1 file changed, 1 insertion(+), 1 deletion(-)
```

项目代码仓库自此也保存了历史记录，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：


```

Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git log
commit 4d20b1cf20739d1b89443e2ba77ac7d047bb26d4 (HEAD -> master)
Author: thb <2120794050@qq.com>
Date: Tue Jun 18 14:04:16 2024 +0800

```

首先，创建了一个名为UI的对象，并设置了两个属性：UI.appwidth：存储了浏览器窗口的内部宽度，即window.innerWidth，UI.appHeight：存储了浏览器窗口的内部高度，即window.innerHeight。计算基础字体大小baseFont：将appwidth除以20，得到的结果作为字体大小。这样可以根据窗口宽度动态调整文本的相对大小，实现响应式设计。设置body元素的字体大小：将baseFont转换为带有单位的字符串（px），并将其赋值给document.body.style.fontSize。这样，body及其所有子元素的字体大小都会根据窗口宽度自动调整。设置body元素的高度：将UI.appHeight减去25（可能是为了排除某些边距或标题栏的高度），然后加上px单位，赋值给document.body.style.height。这使得body元素的高度与浏览器窗口高度相同，实现全屏效果。

6 宽屏和窄屏通用的响应式设计和代码实现

6.1 分析与设计

首先设置 UI.appWidth 为当前窗口宽度，但不超过 600px，确保最小宽度为 600px。UI.appHeight 设置为窗口的总高度。然后计算基础字体大小为 UI.appWidth 除以 20，并应用到 body 元素上，使文本大小随窗口宽度适配。再设置 body 元素的宽度和高度，高度减去 25px（可能为顶部或底部的留白），实现接近全屏的效果。如果窗口宽度小于 1000px，隐藏 ID 为"aid"的元素。特定元素响应式布局：对于 ID 为"aid"的元素，当窗口宽度大于 1000px 时，其宽度设置为窗口宽度减去 UI.appWidth 和 25px，高度同样减去 25px，以适应剩余空间。

这段代码通过条件判断、动态计算尺寸和直接操作 DOM，实现了页面元素的响应式调整，特别是在不同窗口大小下的布局和字体适应性。

用例图以及 Dom 树如下图 4-5 所示：

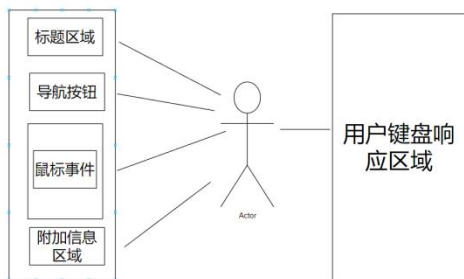


图 4-5

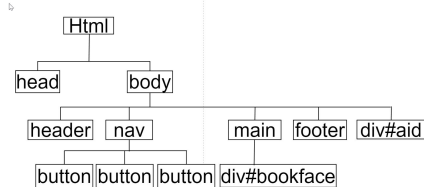


图 4-6

6.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
  <p>
    名著赏析
  </p>
</header>
<nav>
  <button>导航一</button>
  <button>导航二</button>
  <button>导航三</button>
</nav>
<main id="main">
  <div id="bookface">
    书的封面
  </div>

</main>
<footer>
  <p id="statusInfo">
    唐恒彬 江西科技师范大学 2021-2025
  </p>
</footer>
<div id="aid">
  用户键盘响应区
  <p id="keyboard"></p>
</div>
```

二、CSS 代码编写如下

```
*{
  margin: 10px;
  text-align: center;
}
header{
  border: 2px solid rgb(0, 255, 225);
  height: 10%;
  /* 相对字体大小 */
  font-size: 1.6em;
}
nav{
  border: 2px solid rgb(0, 255, 225);
```

```

        height: 10%;
        font-size: 1.2em;
    }
    #main{
        border: 2px solid rgb(0, 255, 225);
        height: 70%;
        font-size: 1.4em;
    }
    footer{
        border: 2px solid rgb(0, 255, 225);
        height: 10%;
        font-size: 1.1em;
    }
    body{
        position: relative;
    }
    #aid{
        position: absolute;
        border: 3px solid rgb(0, 255, 225);
        top: 10px;
        left: 600px;
    }
    #bookface{
        height: 80%;
        width: 80%;
        border: 1px solid red;
        margin: auto;
    }

```

三、JavaScript 代码编写如下

```

var UI = {};
UI.appWidth = window.innerWidth > 600 ? 600 : window.innerWidth;
// UI.appWidth = window.innerWidth;
UI.appHeight = window.innerHeight;
let baseFont = UI.appWidth / 20;
//因为我们改变了 body 对象的字体大小，这个属性会遗传给 body 的子子孙孙，
形成自己的响应式设
document.body.style.fontSize = baseFont + 'px';
//通过把 window 的高度设置为 body（屏幕）的高度，实现全屏。
//通过 CSS 代码对 body 子对象的百分比分配，从而实现响应式设计的目标
document.body.style.width = UI.appWidth + 'px';
document.body.style.height = UI.appHeight - 25 + 'px';
if(window.innerWidth < 1000){
    $("aid").style.display = 'none';
}
$("aid").style.width = window.innerWidth - UI.appWidth - 25 + 'px';

```

```

$("aid").style.height = UI.appHeight - 25 + '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("鼠标移动了，坐标为" + "(" + x + "," + y + ")");
    $("bookface").textContent = "鼠标移动了，坐标为" + "(" + x + "," + y + ")";
});
/对键盘做出响应控制
$("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;
});
unction $(ele){
    if (typeof ele !== 'string'){
        throw("自定义的$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 $

```

6.3 项目的运行和测试

主要效果图以及代码如下图所示：

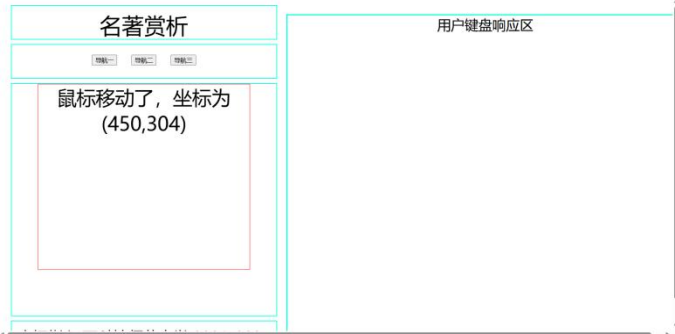


图 4- 7



图 4- 8



图 4- 9

6.4 项目的代码提交和版本管理

编写好 1.3.html 的代码且测试成功后，提交代码：

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git add 1.3.html

Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git commit -m'首先设置UI.appwidth为当前窗口宽度，但不超过600px，确保最小宽度为600px。UI.appHeight设置为窗口的总高度。然后计算基础字体大小为UI.appwidth除以20，并应用到body元素上，使文本大小随窗口宽度适配。再设置body元素的宽度和高度，高度减去25px（可能为顶部或底部的留白），实现接近全屏的效果。如果窗口宽度小于1000px，隐藏ID为"aid"的元素。特定元素响应式布局：对于ID为"aid"的元素，当窗口宽度大于1000px时，其宽度设置为窗口宽度减去UI.appwidth和25px，高度同样减去25px，以适应剩余空间。'
[master 0cd2f35] 1 file changed, 1 insertion(+), 1 deletion(-)
```

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：

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git log
commit 0cd2f35e55a7444464dc7999332996e502e19901 (HEAD -> master)
Author: thb <2120794050@qq.com>
Date: Tue Jun 18 14:23:25 2024 +0800

    首先设置UI.appwidth为当前窗口宽度，但不超过600px，确保最小宽度为600px。UI.appHeight设置为窗口的总高度。然后计算基础字体大小为UI.appwidth除以20，并应用到body元素上，使文本大小随窗口宽度适配。再设置body元素的宽度和高度，高度减去25px（可能为顶部或底部的留白），实现接近全屏的效果。如果窗口宽度小于1000px，隐藏ID为"aid"的元素。特定元素响应式布局：对于ID为"aid"的元素，当窗口宽度大于1000px时，其宽度设置为窗口宽度减去UI.appwidth和25px，高度同样减去25px，以适应剩余空间。
```

7 个性化 UI 设计中鼠标模型

7.1 分析与设计

使用了简化的\$函数来选取元素。它为 ID 为"bookface"的元素添加了一个鼠标按下(mousedown)事件监听器。当用户在该元素上按下鼠标按钮时，执行以下操作：

通过调用 `ev.preventDefault()`阻止了事件的默认行为，比如链接的跳转或文字的选择等。将 `mouse.isDown` 标记设为 `true`，表示鼠标已被按下。这里假设 `mouse`

是一个全局对象，用于存储与鼠标相关的状态信息。然后获取并记录鼠标点击时相对于页面的水平坐标 `ev.pageX` 和垂直坐标 `ev.pageY` 到 `mouse.x` 和 `mouse.y`。之后在控制台打印一条消息，显示鼠标按下时的坐标位置。最后改变 ID 为 "bookface" 的元素的文本内容，显示鼠标按下时的坐标信息。

综上，此代码段实现了一个基本的交互功能，当用户在名为 "bookface" 的元素上按下鼠标时，会显示并记录鼠标的点击位置，并在该元素内实时反馈这一坐标信息，同时阻止元素的任何默认鼠标按下行为

用例图如下图所示：

```
// 鼠标按下
$("#bookface").addEventListener("mousedown", function(ev){
    // 阻止所有默认事件发生
    ev.preventDefault();

    mouse.isDown = true;
    mouse.x = ev.pageX;
    mouse.y = ev.pageY;

    console.log("鼠标按下了，坐标为" + "(" + mouse.x + "," + mouse.y + ")");
    $("#bookface").textContent = "鼠标按下了，坐标为" + "(" + mouse.x + "," + mouse.y + ")";
});
```

图 4-10

在此使用了简化的 `$` 函数来选取元素，它为 ID 为 "bookface" 的元素添加了一个鼠标移动 (`mousemove`) 事件监听器。当用户在按下鼠标后移动鼠标时，执行以下操作：

通过调用 `ev.preventDefault()` 阻止了事件的默认行为，如链接的激活等。检查 `mouse.isDown` 是否为 `true`，表示鼠标已被按下。如果按下，执行后续操作。获取当前鼠标相对于页面的水平坐标 `x` 和垂直坐标 `y`。计算鼠标移动的距离 `deltaX`，即当前鼠标位置 `x` 与上一次鼠标按下位置 `mouse.x` 之间的差值。在控制台打印两条消息，显示鼠标拖动的距离和当前坐标。然后改变 ID 为 "bookface" 的元素的文本内容，显示鼠标拖动时的坐标信息。最后根据鼠标移动的距离，平移元素的位置。元素的左边缘相对于其原始位置偏移 `deltaX/2` 像素。然后重置 `mouse.deltaX` 为 0，以便下一次计算。

这段代码实现了一个简单的拖动效果，当用户按下鼠标并移动时，元素会跟随鼠标移动，同时提供实时的拖动信息反馈。

其具体代码如下图所示：

```

// 鼠标移动
$("#bookface").addEventListener("mousemove", function(ev){
    // 阻止所有默认事件发生
    ev.preventDefault();

    if(mouse.isDown){
        let x = ev.pageX;
        let y = ev.pageY;

        // 每次鼠标移动按下时的移动距离
        mouse.deltaX = x - mouse.x;
        console.log("鼠标正在拖动, 距离为" + mouse.deltaX);
        console.log("鼠标正在拖动, 坐标为" + "(" + x + "," + y + ")");
        $("#bookface").textContent = "鼠标正在拖动, 坐标为" + "(" + x + "," + y + ")";

        $("#bookface").style.left = $("#bookface").offsetLeft/2 + mouse.deltaX/2 + 'px';
        mouse.deltaX = 0;
    }
});

```

图 4- 11

同样使用了简化的\$函数来选取元素，为 ID 为"bookface"的元素添加了一个鼠标松开(mouseup)事件监听器。当用户在该元素上释放鼠标按钮时，执行以下操作：

通过 ev.preventDefault()防止元素的默认行为发生。将 mouse.isDown 设为 false，表示鼠标已松开；并将 mouse.x 与 mouse.y 重置为 0，为下一次按下做准备。在控制台打印消息“鼠标松开了”，并更新元素的文本内容为“鼠标松开了”。检查 mouse.deltaX 的绝对值是否大于 50 像素。如果是，则在元素的文本内容后追加“这是有效拖动”，表明用户完成了一次具有足够位移的有效拖动操作。

总结而言，此段代码处理了鼠标松开的动作，不仅更新了状态并通知用户鼠标已释放，还根据拖动距离判断拖动的有效性，并据此提供视觉反馈。

其主要代码如下：

```

// 鼠标松开
$("#bookface").addEventListener("mouseup", function(ev){
    // 阻止所有默认事件发生
    ev.preventDefault();

    mouse.isDown = false;
    mouse.x = 0;
    mouse.y = 0;
    console.log("鼠标松开了");
    $("#bookface").textContent = "鼠标松开了";

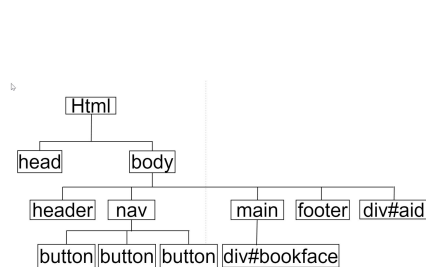
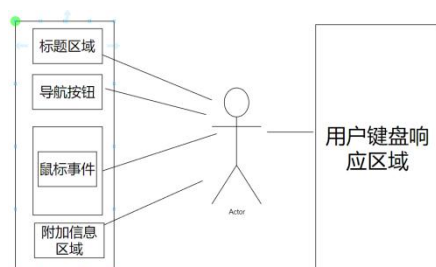
    if(Math.abs(mouse.deltaX) > 50){
        $("#bookface").textContent += "这是有效拖动";
    }
});

```

图 4- 12

用例图以及 Dom 树如下图 4- 13 图 4- 13

图 4- 14 所示：



7.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
  <p>
    名著赏析
  </p>
</header>
<nav>
  <button>导航一</button>
  <button>导航二</button>
  <button>导航三</button>
</nav>
<main id="main">
  <div id="bookface">
    书的封面
  </div>

  </main>
<footer>
  <p id="statusInfo">
    唐恒彬 江西科技师范大学 2021-2025
  </p>
</footer>
<div id="aid">
  用户键盘响应区
  <p id="keyboard"></p>
</div>
```

二、CSS 代码编写如下：

```
*{
  margin: 10px;
  text-align: center;
}
header{
  border: 2px solid rgb(255, 119, 0);
  height: 10%;
  /* 相对字体大小 */
  font-size: 1.6em;
}
```



```

nav{
    border: 2px solid rgb(255, 119, 0);
    height: 10%;
    font-size: 1.2em;
#main{
    border: 2px solid rgb(255, 119, 0);
    height: 70%;
    font-size: 1.4em;
    position: relative;
}
footer{
    border: 2px solid rgb(255, 119, 0);
    height: 10%;
    font-size: 1.1em;
}
body{
    position: relative;
}
#aid{
    position: absolute;
    border: 3px solid rgb(255, 119, 0);
    top: 10px;
    left: 600px;
}
#bookface{
    position: absolute;
    height: 80%;
    width: 80%;
    border: 1px solid red;
    background-color: rgb(0, 229, 255);
    left: 7%;
}

```

三、JavaScript 代码编写如下：

```

var UI = {};
UI.appWidth = (window.innerWidth > 600 ? 600 : window.innerWidth);
UI.appHeight = window.innerHeight;
let baseFont = UI.appWidth / 20;
//因为我们改变了 body 对象的字体大小，这个属性会遗传给 body 的子子孙孙，
形成自己的响应式
document.body.style.fontSize = baseFont + 'px';
//通过把 window 的高度设置为 body（屏幕）的高度，实现全屏。
//通过 CSS 代码对 body 子对象的百分比分配，从而实现响应式设计的目标
document.body.style.width = UI.appWidth + 'px';

```

```

document.body.style.height = UI.appHeight - 25 + 'px';
if(window.innerWidth < 1000){
    $("aid").style.display = 'none';
}
$("aid").style.width = window.innerWidth - UI.appWidth - 25 + 'px';
$("aid").style.height = UI.appHeight - 25 + 'px';
//尝试对鼠标设计 UI 控制
var mouse = {};
mouse.isDown = false;
mouse.x = 0;
mouse.y = 0;
mouse.deltaX = 0;
//鼠标按下
$("bookface").addEventListener("mousedown", function(ev){
    //阻止所有默认事件发生
    ev.preventDefault();
    mouse.isDown = true;
    mouse.x = ev.pageX;
    mouse.y = ev.pageY;
    console.log("鼠标按下了，坐标为" + "(" + mouse.x + "," + mouse.y + ")");
    $("bookface").textContent = "鼠标按下了，坐标为" + "(" + mouse.x + "," +
mouse.y +
    });
//鼠标移动
$("bookface").addEventListener("mousemove", function(ev){
    //阻止所有默认事件发生
    ev.preventDefault();

    if(mouse.isDown){
        let x = ev.pageX;
        let y = ev.pageY;
        //每次鼠标移动按下时的移动距离
        mouse.deltaX = x - mouse.x;
        console.log("鼠标正在拖动，距离为" + mouse.deltaX);
        console.log("鼠标正在拖动，坐标为" + "(" + x + "," + y + ")");
        $("bookface").textContent = "鼠标正在拖动，坐标为" + "(" + x + "," + y + ")";
        $("bookface").style.left = $("bookface").offsetLeft/2 + mouse.deltaX/2 + 'px';
        mouse.deltaX = 0;
    }
});
//鼠标松开
$("bookface").addEventListener("mouseup", function(ev){
    //阻止所有默认事件发生

```

```

    ev.preventDefault();
    mouse.isDown = false;
    mouse.x = 0;
    mouse.y = 0;
    console.log("鼠标松开了");
    $("#bookface").textContent = "鼠标松开了";
    if(Math.abs(mouse.deltaX) > 50){
        $("#bookface").textContent += "这是有效拖动";
    }
});
/对键盘做出响应控制
$("#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 $

```

7.3 项目的运行和测试

鼠标模型实现效果以及二维码如下：

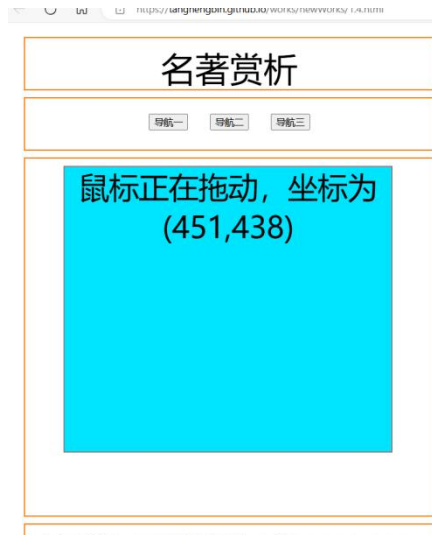


图 4-15



图 4-16

7.4 项目的代码提交和版本管理

编写好 1.4.html 的代码且测试成功后，提交代码：

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git add 1.4.html

Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git commit -m'此代码段实现了一个基本的交互功能，当用户在名为"bookface"的元素上
按下鼠标时，会显示并记录鼠标的点击位置，并在该元素内实时反馈这一坐标信息，同时阻
止元素的任何默认鼠标按下行为。同时实现了一个简单的拖动效果，当用户按下鼠标并移动
时，元素会跟随鼠标移动，同时提供实时的拖动信息反馈。并且处理了鼠标松开的动作，不
仅更新了状态并通知用户鼠标已释放，还根据拖动距离判断拖动的有效性，并据此提供视觉
反馈。'
[master 7e4db8b] 此代码段实现了一个基本的交互功能，当用户在名为"bookface"的元素
上按下鼠标时，会显示并记录鼠标的点击位置，并在该元素内实时反馈这一坐标信息，同时
阻止元素的任何默认鼠标按下行为。同时实现了一个简单的拖动效果，当用户按下鼠标并移
动时，元素会跟随鼠标移动，同时提供实时的拖动信息反馈。并且处理了鼠标松开的动作，
不仅更新了状态并通知用户鼠标已释放，还根据拖动距离判断拖动的有效性，并据此提供视
觉反馈。
1 file changed, 1 insertion(+), 1 deletion(-)
```

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：

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git log
commit 7e4db8b04423607fc9136c34d3e296858016bb73 (HEAD -> master)
Author: thb <2120794050@qq.com>
Date: Tue Jun 18 14:52:58 2024 +0800
```

此代码段实现了一个基本的交互功能，当用户在名为“bookface”的元素上按下鼠标时，会显示并记录鼠标的点击位置，并在该元素内实时反馈这一坐标信息，同时阻止元素的任何默认鼠标按下行为。同时实现了一个简单的拖动效果，当用户按下鼠标并移动时，元素会跟随鼠标移动，同时提供实时的拖动信息反馈。并且处理了鼠标松开的动作，不仅更新了状态并通知用户鼠标已释放，还根据拖动距离判断拖动的有效性，并据此提供视觉反馈。

8 通用的 UI 设计，用一套代码同时为触屏和鼠标建模

8.1 分析与设计

下面这段代码定义了一个名为 `Pointer` 的全局对象，用于追踪和管理鼠标或触摸事件的状态。`Pointer` 对象有以下属性和功能：`isDown`：一个布尔值，表示是否有指针（鼠标或触摸）按下。`x`：指针最后一次的位置的 `x` 坐标。`deltaX`：用于存储指针移动时的 `x` 轴偏移量。

代码块定义了一个名为 `handleBegin` 的函数，用于处理鼠标或触摸的开始事件。这个函数根据事件类型（触摸或鼠标）执行不同的操作

（1）触摸事件处理：

当事件对象 `ev` 包含 `touches` 属性（表明是触摸事件）时，它读取第一个触摸点的 `pageX` 和 `pageY` 坐标，更新 `Pointer` 的状态，并在控制台和 `bookface` 元素上显示触摸开始的坐标信息。

（2）鼠标事件处理：

对于非触摸事件（通常是鼠标事件），它读取事件的 `pageX` 和 `pageY` 坐标，同样更新 `Pointer` 的状态，并在控制台和页面上显示鼠标按下的坐标信息。

这个函数设计得可以兼容触摸屏设备和传统鼠标操作，确保在不同设备上都能正确响应指针的开始（按下或触摸）事件。

主要代码如下图所示：

```

//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+")";
    }
  };
}

```

图 4-17

下面这段代码定义了一个名为 `handleEnd` 的函数，用于处理指针（鼠标或触摸）的结束事件。这个函数根据事件类型（触摸或鼠标）执行不同的操作：

（1）通用操作:

将 `Pointer.isDown` 设为 `false`，表示指针已经释放。阻止事件的默认行为，例如链接的点击或表单的提交等。

（2）触摸事件处理:

如果事件对象 `ev` 包含 `touches` 属性（表示触摸事件结束），更新 `bookface` 元素的文本内容为“触屏事件结束”。检查 `Pointer.deltaX` 的绝对值，如果大于 100，说明进行了有效的触屏滑动，更新文本内容并保持当前位置。否则，视为无效滑动，将元素的 `left` 属性设置回'7%'，表示回到初始位置。

（3）鼠标事件处理:

对于非触摸事件（通常是鼠标事件），更新 `bookface` 元素的文本内容为“鼠标松开”。同样检查 `Pointer.deltaX` 的绝对值，若大于 100，表示进行了有效的拖动，更新文本内容。否则，视为无效拖动，将元素的 `left` 属性设置回'7%'。

这个函数确保在触摸屏和鼠标操作的结束时，能够正确地识别和处理有效或无效的滑动拖动，并提供相应的反馈。

主要代码如下图所示：

```

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%' ;
    }
  }
}
};

```

图 4-18

下面这段代码定义了一个名为 `handleMoving` 的函数，用于处理指针（鼠标或触摸）的移动事件。这个函数根据事件类型（触摸或鼠标）执行不同的操作：

（1）通用操作：

阻止事件的默认行为，例如文本选中等。

（2）触摸事件处理：

如果事件对象 `ev` 包含 `touches` 属性（表示触摸事件），并且 `Pointer.isDown` 为 `true`（表示触摸已开始），则：更新 `Pointer.deltaX` 为当前触摸点相对于初始位置的 `x` 轴偏移量。更新 `bookface` 元素的文本内容，显示当前的滑动距离。并根据 `Pointer.deltaX` 更新元素的 `left` 属性，实现元素的水平移动。

（3）鼠标事件处理：

对于非触摸事件（通常是鼠标事件），如果 `Pointer.isDown` 为 `true`，则更新 `Pointer.deltaX` 为当前鼠标位置相对于初始位置的 `x` 轴偏移量。更新 `bookface` 元素的文本内容，显示当前的拖动距离。并根据 `Pointer.deltaX` 更新元素的 `left` 属性，实现元素的水平移动。

这个函数确保在触摸屏或鼠标移动时，能够正确地追踪和处理滑动/拖动，更新元素的位置，并提供实时的反馈。

主要代码如下图所示：

```

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' ;
    }
  }
}
};

```


图 4-19

最后这段代码为 ID 为"bookface"的元素以及 body 元素添加了多个事件监听器，以实现不同类型的用户交互，事件监听器依次为：

(1) 鼠标按下(mousedown):

添加 handleBegin 函数作为事件处理程序，用于处理鼠标按下事件，开始追踪鼠标或触摸的位置。

(2) 触摸开始(touchstart):

添加 handleBegin 函数，处理触摸开始事件，与鼠标按下事件类似。

(3) 鼠标松开和触摸结束(mouseup 和 touchend):

添加 handleEnd 函数，处理鼠标或触摸结束事件，更新元素状态并判断是否进行了有效拖动或滑动。

(3) 鼠标离开(mouseout):

添加 handleEnd 函数，处理鼠标离开事件，模拟鼠标松开的效果，结束追踪。

(4) 鼠标移动和触摸移动(mousemove 和 touchmove):

添加 handleMoving 函数，处理鼠标或触摸的移动，更新元素位置并显示移动距离。

(5) 键盘按键(keypress):为 body 元素添加事件监听器，当用户按下键盘键时，会在 ID 为"aid"的元素中显示所按的键。

这些事件监听器组合起来，使得"bookface"元素支持鼠标和触摸操作，包括开始、结束、移动以及在元素内的有效滑动/拖动。同时，监听键盘按键事件，用于在指定元素中显示用户输入的字符。

主要代码如下图所示：

```
$("#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 ;
    | });
```

图 4-20

用例图以及 DOM 树如下图 4-21

图 4-22 所示：

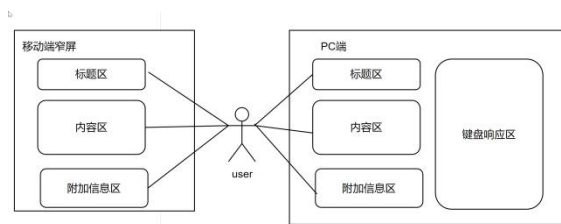


图 4- 21

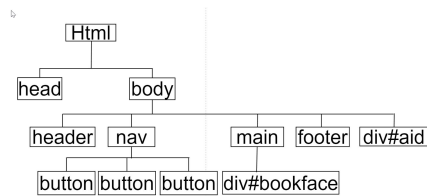


图 4- 22

8.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
  <p id="book">
    名著赏析
  </p>
</header>
<nav>
  <button>向前</button>
  <button>向后</button>
  <button>其他</button>
</nav>
<main id="main">
<div id="bookface">
  这是书的封面图<br>
  在此对象范围拖动鼠标(本例触屏无效)
</div>
</main>
<footer>
```

Copyright 唐恒彬 江西科技师范大学 2024--2025

```
</footer>
<div id="aid">
  <p>用户键盘响应区</p>
</div>
```

二、CSS 代码编写如下：

```
*{
  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% ;
    }

```

三、JavaScript 代码编写如下：

```

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";
f(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
ar 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 $

```

8.3 项目的运行和测试

其实现效果和二维码如下：



图 4-23

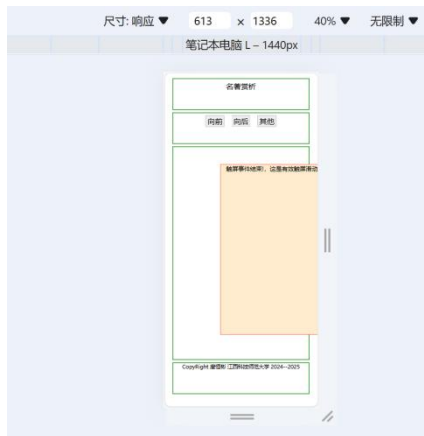


图 4-24



图 4-25

8.4 项目的代码提交和版本管理

编写好 1.5.html 的代码且测试运行成功后，执行下面命令提交代码：

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git add 1.5.html

Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git commit -m'首先代码定义了一个名为Pointer的全局对象，用于追踪和管理鼠标或触摸事件的状态。Pointer对象有以下属性和功能：isDown：一个布尔值，表示是否有指针（鼠标或触摸）按下。x：指针最后一次的位置的x坐标。deltaX：用于存储指针移动时的x轴偏移量。然后定义了一个名为handleBegin的函数，用于处理鼠标或触摸的开始事件。这个函数设计得可以兼容触摸屏设备和传统鼠标操作，确保在不同设备上都能正确响应指针的开始（按下或触摸）事件，一个名为handleEnd的函数，用于处理指针（鼠标或触摸）的结束事件。这个函数确保在触摸屏和鼠标操作的结束时，能够正确地识别和处理有效或无效的滑动拖动，并提供相应的反馈。同时定义了一个名为handleMoving的函数，这个函数确保在触摸屏或鼠标移动时，能够正确地追踪和处理滑动/拖动，更新元素的位置，并提供实时的反馈。最后这段代码为ID为"bookface"的元素以及body元素添加了多个事件监听器，以实现不同类型的用户交互。'

[master ef68c1a] 首先代码定义了一个名为Pointer的全局对象，用于追踪和管理鼠标或触摸事件的状态。Pointer对象有以下属性和功能：isDown：一个布尔值，表示是否有指针（鼠标或触摸）按下。x：指针最后一次的位置的x坐标。deltaX：用于存储指针移动时的x轴偏移量。然后定义了一个名为handleBegin的函数，用于处理鼠标或触摸的开始事件。这个函数设计得可以兼容触摸屏设备和传统鼠标操作，确保在不同设备上都能正确响应指针的开始（按下或触摸）事件，一个名为handleEnd的函数，用于处理指针（鼠标或触摸）的结束事件。这个函数确保在触摸屏和鼠标操作的结束时，能够正确地识别和处理有效或无效的滑动拖动，并提供相应的反馈。同时定义了一个名为handleMoving的函数，这个函数确保在触摸屏或鼠标移动时，能够正确地追踪和处理滑动/拖动，更新元素的位置，并提供实时的反
```

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：

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git log
commit ef68c1a8831c4ccd8ce37c3680e5579d4de0842b (HEAD -> master)
Author: thb <2120794050@qq.com>
Date: Tue Jun 18 15:26:35 2024 +0800
```

首先代码定义了一个名为Pointer的全局对象，用于追踪和管理鼠标或触摸事件的状态。Pointer对象有以下属性和功能：isDown：一个布尔值，表示是否有指针（鼠标或触摸）按下。x：指针最后一次的位置的x坐标。deltaX：用于存储指针移动时的x轴偏移量。然后定义了一个名为handleBegin的函数，用于处理鼠标或触摸的开始事件。这个函数设计得可以兼容触摸屏设备和传统鼠标操作，确保在不同设备上都能正确响应指针的开始（按下或触摸）事件，一个名为handleEnd的函数，用于处理指针（鼠标或触摸）的结束事件。这个函数确保在触摸屏和鼠标操作的结束时，能够正确地识别和处理有效或无效的滑动拖动，并提供相应的反馈。同时定义了一个名为handleMoving的函数，这个函数确保在触摸屏或鼠标移动时，能够正确地追踪和处理滑动/拖动，更新元素的位置，并提供实时的反馈。最后这段代码为ID为"bookface"的元素以及body元素添加了多个事件监听器，以实现不同类型的用户交互。

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

9.1 分析与设计

最后用底层函数\$快速抓取页面元素 body 对象。并用监听事件对键盘的敲击做出相应得响应。这段代码包含了两个部分，分别处理键盘的按下(keydown)和释放(keyup)事件，目标是监控用户的键盘输入并在页面上显示相关信息，同时根据输入的字符类型（字母、数字、特定标点符号）更新一个名为 typeText 的元素的内容。

Keydown 处理：

- 1.为 body 元素添加了一个 keydown 事件监听器。
- 2.使用 ev.preventDefault()防止按键的默认行为，如滚动页面。
- 3.通过 ev.key 和 ev.keyCode 获取按键的字符表示和键盘编码。
- 4.将按键信息更新到 ID 为 keyboard 的元素中。

Keyup 处理：

- 1.为 body 元素添加了一个 keyup 事件监听器。
- 2.同样使用 ev.preventDefault()。
- 3.通过 ev.key 和 ev.keyCode 获取按键的字符表示和键盘编码。

- 4.在释放按键时，更新 keyboard 元素的文本内容。
- 5.自定义函数 printLetter:
- 6.此函数用于判断输入的字符是否应该被打印到 typeText 元素中。
- 7.检查字符长度大于 1 的情况（通常不适用于键盘输入，可能是为了过滤某些特殊输入）。字母、数字、特定标点符号：允许字母、数字以及指定的标点符号通过，这些字符会被追加到 typeText 中。
- 8.通过数组 puncs 定义了一组允许的标点符号。
- 9.如果字符符合条件，则返回 true，否则 false。

用例图以及 Dom 树如图 4- 26 图 4- 27 所示：

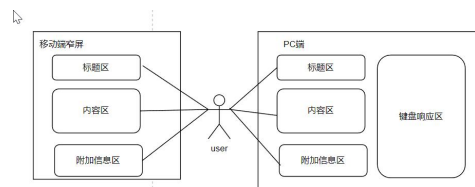


图 4- 26

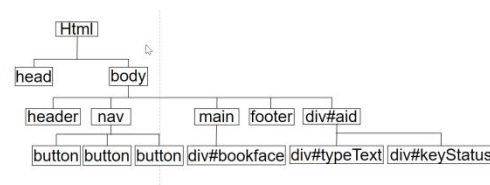


图 4- 27

9.2 项目的实现和编程

一、HTML 代码编写如下：

```
<header>
  <p id="book">
    名著赏析
  </p>
</header>
<nav>
  <button>向前</button>
  <button>向后</button>
  <button>其他</button>
</nav>
<main id="main">
<div id="bookface">
  这是书的封面图<br>
  在此对象范围拖动鼠标/滑动触屏<br>
  拖动/滑动超过 100 像素，视为有效 UI 互动！
</div>
</main>
<footer>
```


</footer>

<div id="aid">

用户键盘响应区

<p id="typeText"></p>

<hr>

<p id="keyboard"></p>

</div>

二、CSS 代码编写如下：

```
*{
    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 rgb(115, 0, 128);
    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% ;
}

```

三、JavaScript 代码编写如下：

```

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';

//尝试对鼠标设计 UI 控制
var Pointer = {};
Pointer.isDown= false;

```

```

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

```

```

        console.log("Touch is moving");
        Pointer.deltaX = parseInt( ev.touches[0].pageX - Pointer.x );
        $("#bookface").textContent= "正在滑动触屏，滑动距离： " + Pointer.deltaX
+"px 。 ";
        $('#bookface').style.left = Pointer.deltaX + 'px' ;
    }
} else {
    if (Pointer.isDown){
        console.log("Pointer isDown and moving");
        Pointer.deltaX = parseInt( ev.pageX - Pointer.x );
        $("#bookface").textContent= "正在拖动鼠标，距离： " + Pointer.deltaX +"px 。
";
        $('#bookface').style.left = Pointer.deltaX + 'px' ;
    }
}
};

```

```

$("#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);
/**** 添加"keydown"和"keyup"这 2 个键盘底层事件处理后，keypress 这个高
层键盘事件响应被系统忽略

```

```

$("#body").addEventListener("keypress", function(ev){
    $("#typeText").textContent += ev.key ;
});

```

```

$("#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 k = ev.key;
    let c = ev.keyCode;
    $("#keyboard").textContent = "松开按键 : " + k + " , "+" 字符编码 : " + c;
});

```

*****/

//提出问题： 研究利用"keydown"和"keyup"2 个底层事件，实现同时输出按键状

态和文本内容

```
$("#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)
});

} //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 $

```

9.3 项目的运行和测试

实现效果如下图所示：



图 4-28

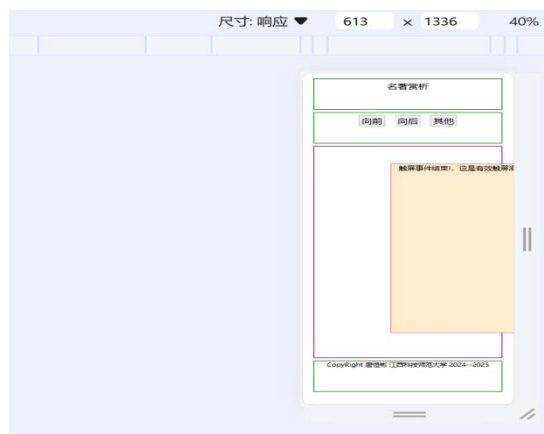


图 4-29



图 4-30

9.4 项目的代码提交和版本管理

编写好 1.6.html 的代码且测试运行成功后，执行下面命令提交代码：

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git add 1.6.html

Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git commit -m'最后用底层函数$快速抓取页面元素body对象。并用监听事件对键盘的敲
击做出相应得响应。这段代码包含了两个部分，分别处理键盘的按下(keydown)和释放(keyup)事件，目标是监控用户的键盘输入并在页面上显示相关信息，同时根据输入的字符类型（字母、数字、特定标点符号）更新一个名为typeText的元素的内容。'
[master 5f397aa] 最后用底层函数$快速抓取页面元素body对象。并用监听事件对键盘的敲
击做出相应得响应。这段代码包含了两个部分，分别处理键盘的按下(keydown)和释放(keyup)事件，目标是监控用户的键盘输入并在页面上显示相关信息，同时根据输入的字符类型（字母、数字、特定标点符号）更新一个名为typeText的元素的内容。
1 file changed, 1 insertion(+), 1 deletion(-)
```

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：

```
Lenovo@LAPTOP-M84ELIHO MINGW64 /works/THB_text/newWorks (master)
$ git log
commit 5f397aa5ca5f8685c5c06a93ef9b81669a6dd637 (HEAD -> master)
Author: thb <2120794050@qq.com>
Date: Tue Jun 18 16:36:16 2024 +0800

最后用底层函数$快速抓取页面元素body对象。并用监听事件对键盘的敲击做出相应得
的响应。这段代码包含了两个部分，分别处理键盘的按下(keydown)和释放(keyup)事件，目
标是监控用户的键盘输入并在页面上显示相关信息，同时根据输入的字符类型（字母、数字、
特定标点符号）更新一个名为typeText的元素的内容。
```

10.谈谈本项目中的高质量代码

本项目 URL 地址



图 5- 1

编写高质量代码是软件开发的核心目标之一，它直接关系到软件项目的可维护性、扩展性、性能及团队协作效率。高质量代码的编写，可以体现在具有清晰的命名约定、模块化和组件化、良好的注释和文档、遵循设计模式和原则等。

例如：

(1) `let baseFont = UI.appWidth / 50;`

//因为我们改变了 `body` 对象的字体大小，这个属性会遗传给 `body` 的子子孙孙，形成自己的响应式设计

`document.body.style.fontSize = baseFont + 'px';`

//通过把 `window` 的高度设置为 `body`（屏幕）的高度，实现全屏。

//通过 CSS 代码对 `body` 子对象的百分比分配，从而实现响应式设计的目标。

`document.body.style.height = UI.appHeight - 25 + 'px';`

利用 `body` 的遗传机制，对 CSS 字体实行相对控制，实现了对字体的响应式设计。

(2) //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

```

本代码实现了对鼠标事件和触屏事件的设计，一次性对两种事件进行分析，减少代码量的同时也增加了代码的可读性和可扩展性。以代码块的形式，使得项目逻辑更清晰，易于理解，其模块化的特性可以体现其复用性。

11.用 git 工具展开代码的版本管理和发布

11.1 经典 Bash 工具介绍

Bash（Bourne-Again SHell）是一个广泛使用的 Unix shell，它提供了许多内置命令和功能，使得用户可以通过命令行接口与操作系统交互。

在当今现代软件开发的时代，良好的代码版本管理是保持一个项目整洁和团队协作的关键。Git 是一个开源的分布式版本控制系统，是目前世界上最先进、最流行的版本控制系统，可以快速高效地处理从很小到非常大的项目版本管理。其特点为：项目越大越复杂，协同开发者越多，越能体现出 Git 的高性能和高可用性^[10]。Git 可以高效的对代码进行版本控制和发布管理，Git 的分布式的特点意味着每个开发者在本地计算机上都可以拥有一个完整的项目副本，可以对这个副本进行独立的开发和测试，不需要依赖中央服务器。这样的方式允许开发者可以自由地创建分支，实验新的功能或修复 bug，然后通过合并请求将更改合并返回主分支。这种灵活性允许团队成员并行工作，提高了开发效率，减少了等待时间。以我的理解，其对代码的控制主要分为以下几个部分：

A.安装 Git:首先确保你的计算机上已安装 Git。可以从 Git 官方网站下载安装。

创建本地仓库:在项目根目录打开终端或命令提示符。运行 `git init` 命令来初始化一个新的 Git 仓库。

B.日常开发流程

添加文件:使用 `git add<file>`将文件添加到暂存区,或者使用 `git add`.添加所有修改过的文件。提交更改:使用 `git commit -m "Commit message"`提交暂存区的更改到本地仓库。

C.远程仓库操作

添加远程仓库:如果是第一次推送,需要添加远程仓库地址:`git remote add origin <repository_url>`,开发者可以将本地仓库与 GitHub 远程仓库关联。使用 `git push` 和 `git pull` 进行代码的同步,实现团队间的共享和协作。推送代码:推送主分支到远程仓库:`git push -u origin main` (或 `master`, 取决于你的默认分支名称)。

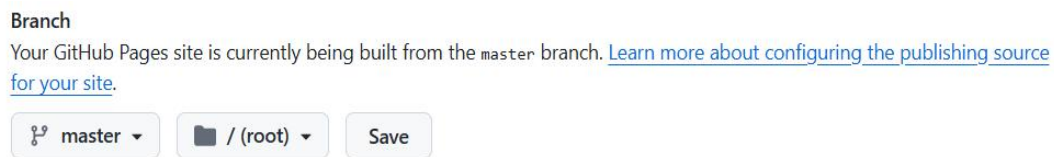
D.版本回退与历史查看

查看提交历史:使用 `git log` 查看提交历史, `git log --oneline` 查看简略历史。

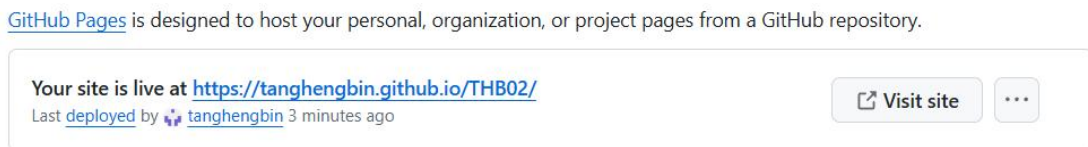
版本回退:回退到指定提交:`git reset --hard <commit-hash>`, 其中<commit-hash>是从 `git log` 获取的提交 ID。

11.2 通过 gitHub 平台实现本项目的全球域名

代码仓库建立成功后,如图:



选择分支后保存,即可得到全球域名,如下图所示:




11.3 创建一个空的远程代码仓库

建立新代码仓库下图所示:

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 *	Repository name *
 tanghengbin ▾	/ THB02
	✓ THB02 is available.

Great repository names are short and memorable. Need inspiration? How about [literate-octo-fiesta](#) ?

Description (optional)

输入合适的地址后点击创建即可。

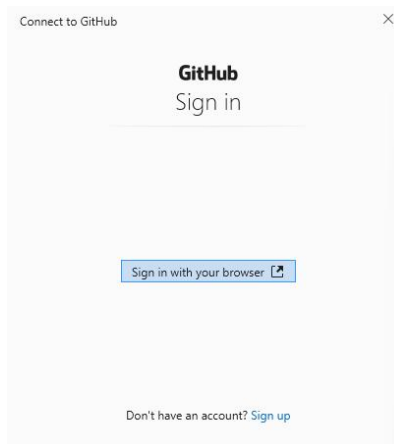
Create repository

11.4 设置本地仓库和远程代码仓库的链接

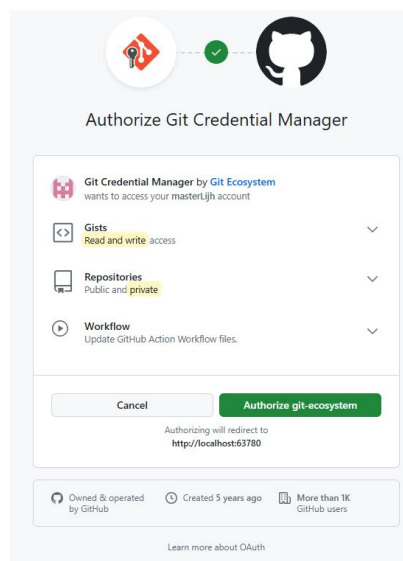
进入本地 webUI 项目的文件夹后，通过下面的命令把本地代码仓库与远程建立密钥链接。

```
$ echo "WebUI 应用的远程 http 服务器设置" >> README.md #添加了一个文件  
# 名称叫做 README，内容为 WebUI 应用的远程 http 服务器设置的 markdown 文件  
$ git init #初始化仓库  
$ git add README.md # 将文件添加进仓库的暂存区中  
$ git commit -m "这是我第一次把代码仓库上传至 gitHub 平台"  
$ git branch -M main # 创建分支 main  
$ git remote add origin https://github.com/tanghengbin/THB02.git  
#添加远程仓库  
$ git push -u origin main #将本地仓库中的文件提交到远程仓库中
```

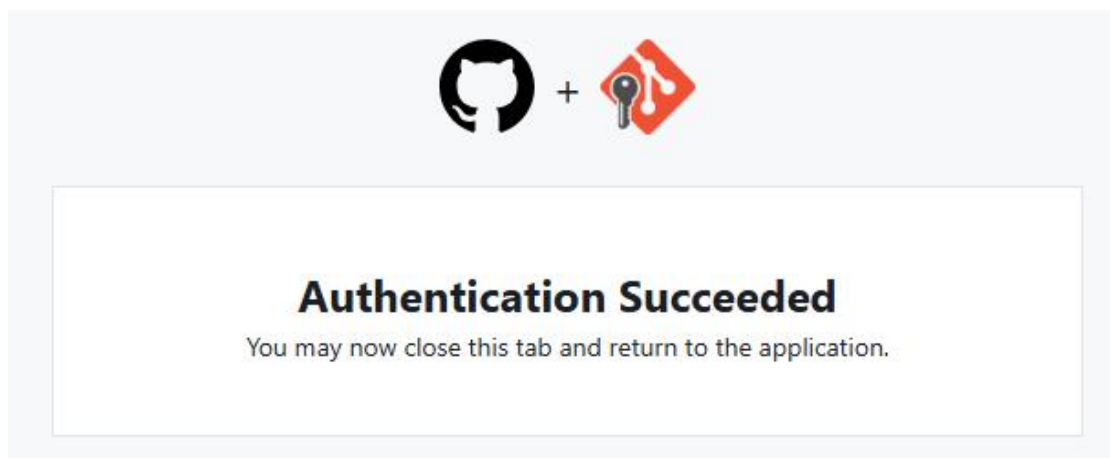
本项目使用 window 平台，gitbash 通过默认浏览器实现密钥生成和记录，第一次链接会要求开发者授权，如下图所示：



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
- [9]仇礼钦,王鑫,盛飞龙,等.基于 Git 的软件项目管理配置方法及应用实践[J].机电工程技术,2023,52(05):223-227.