深 圳 大 学 实 验 报 告

课程名称：web 开发和人机交互导论

实验项目名称：JavaScript 基础应用

学院：计算机与软件学院

专业：计算机科学与技术

指导教师：李俊杰

报告人：曹秦鲁 学号：2023150203 班级：国际班

实验时间：2024 年 11 月 30 日

实验报告提交时间：

教务处制

实验目的与要求：

1. 熟悉 JavaScript 的基础语法

2. 实现 JavaScript 语言的简单应用，并利用 JavaScript 实现简单的网页效果。

说明：本次实验代码没有单独创建 css 文件和 js 文件，因为项目容量小，没有必要单独创建文件，放在一起更方便。

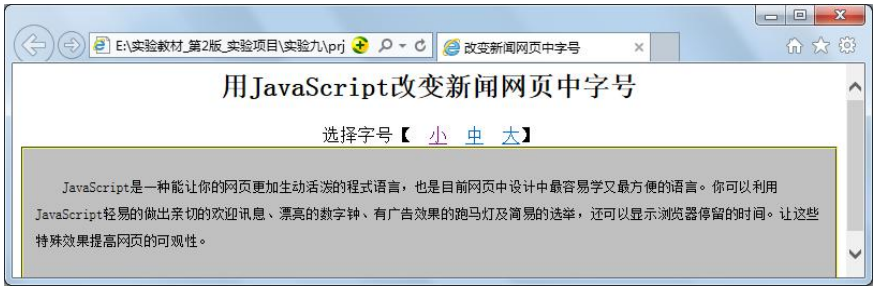
实验内容及过程：

1. JavaScript 文本样式控制器

1. 项目概述

本项目是一个基于 JavaScript 的文本样式控制器，允许用户动态调整网页文本的各种样式属性。项目采用纯前端技术栈，实现了一个简洁、易用的文本样式控制面板。

目标效果：



我的实现

2. 功能特点

2.1 基础文本控制
- 字号调整：支持小、中、大三种字号切换
- 字体切换：支持宋体、楷体、黑体三种字体 创新
- 字体粗细：可在正常和加粗之间切换 创新
- 文本对齐：支持左对齐、居中、右对齐 创新

2.2 特效功能
- 主题切换：支持明暗两种主题模式 创新
- 关键词高亮：自动识别并高亮显示重要关键词 创新
- 自动滚动：实现文本自动平滑滚动效果 创新
- 字数统计：实时显示文本字数 创新

3. 技术栈
- HTML5：页面结构和语义化标签
- CSS3：
  - Flexbox 布局
  - CSS 变量（自定义属性）
  - 过渡动画
  - 媒体查询
- JavaScript：
  - DOM 操作
  - 事件处理
  - 正则表达式

4. 实现思路

4.1 布局设计
- 采用 Flexbox 布局实现控制面板的按钮对齐
- 使用 CSS 变量实现主题切换

4.2 交互实现

```
// 字号调整实现
function changeFontSize(size) {
    const content = document.getElementById('newsContent');
    switch(size) {
        case 'small': content.style.fontSize = '14px'; break;
        case 'medium': content.style.fontSize = '16px'; break;
        case 'large': content.style.fontSize = '20px'; break;
    }
}
```

1. 函数接收一个 size 参数，表示要设置的字号大小
2. 使用 getElementById 获取内容元素
3. 通过 switch 语句根据不同的 size 值设置不同的字号
4. 直接操作 DOM 元素的 style 属性修改 fontSize
5. 使用像素(px)作为单位确保跨浏览器兼容性

```
// 样式切换实现
function changeStyle(property, value) {
    const content = document.getElementById('newsContent');
    content.style[property] = value;
}
```

1. 函数采用两个参数：property（CSS 属性名）和 value（属性值）
2. 使用中括号语法动态设置 style 属性
3. 这种设计使得函数具有通用性，可以修改任何 CSS 属性
4. 支持字体、对齐方式等多种样式的切换

```
// 主题切换实现
function toggleTheme() {
    document.body.classList.toggle('dark-theme');
}
```

1. 使用 classList.toggle 方法切换类名
2. 当 dark-theme 类存在时移除它，不存在时添加它
3. CSS 中通过 dark-theme 类定义了不同的配色方案
4. 使用 CSS 变量实现主题切换，提高代码复用性
5. 整个过程不需要使用 JavaScript 直接操作样式

4.3 特效实现

```
// 关键词高亮
function highlightKeywords() {
    const content = document.getElementById('newsContent');
    const keywords = ['JavaScript', '网页', '程式语言'];
    let text = content.innerHTML;

    keywords.forEach(keyword => {
        const regex = new RegExp(keyword, 'g');
        text = text.replace(regex, `<span class="highlight">${keyword}</span>`);
    });

    content.innerHTML = text;
}
```

1. 关键词数组定义：
   - 预定义了需要高亮显示的关键词数组
   - 可以根据需要轻松添加或修改关键词

2. 正则表达式使用：
   - 使用 RegExp 创建动态的正则表达式
   - 'g'标志表示全局匹配，会替换所有匹配项
   - 支持中英文关键词的匹配

```
更新字数统计
计算文本内容的字数并显示
```

```
function updateWordCount() {
    const content = document.getElementById('newsContent');
    const wordCount = content.textContent.trim().length;
    document.getElementById('wordCount').textContent = `字数统计：${wordCount}个字`;
}
```

5. 创新点

5.1 界面设计
- 采用现代化的 UI 设计
- 控制面板布局合理，操作直观
- 使用 CSS 变量实现主题切换，提高代码复用性

5.2 功能创新
- 实时字数统计
- 智能关键词识别和高亮
- 平滑的自动滚动效果
- 深色模式支持

5.3 技术创新
- 使用 CSS 变量实现主题切换，避免大量样式重复
- 模块化的 JavaScript 函数设计，便于维护和扩展
- 响应式设计适配不同设备

页面效果展示：

关键字高亮

不同主题切换：黑暗主题

项目 2　计算任意区间内连续自然数的累加和

1. 项目概述

本项目是一个基于 Web 技术的连续自然数累加和计算器，能够计算任意区间内连续自然数的累加和，并提供可视化的计算过程展示。项目采用纯前端技术栈，实现了一个交互友好的计算工具。

2. 功能特点

2.1 基础功能

- 输入验证：确保输入为有效的正整数
- 区间计算：自动处理起始数大于终止数的情况
- 结果显示：清晰展示计算结果
- 一键清空：快速重置所有输入和显示

2.2 增强功能　　以下都是要求之外的创新点

- 实时输入验证：即时检查输入的有效性
- 区间可视化：动态显示数字范围
- 计算过程展示：分步骤显示计算公式和过程
- 动画效果：提供流畅的视觉反馈

3. 技术栈

- HTML5：页面结构和语义化标签
- CSS3：
  - Flexbox 布局
  - 过渡动画
  - 变量定义
  - 伪类选择器
- JavaScript：
  - DOM 操作
  - 事件处理
  - 数学计算
  - 动画控制

4. 实现思路详解

4.1 输入验证实现

```javascript
function isValidPositiveInteger(value) {
    // 检查是否为空
    if (!value || !value.trim()) {
        return false;
    }
    // 检查是否为正整数
```

```
    const num = parseInt(value);
    return !isNaN(num) && num > 0 && num === parseFloat(value) && /^\d+$/.test(value);
}

function calculate() {
    const startInput = document.getElementById('startNum');
    const endInput = document.getElementById('endNum');
    const startNum = parseInt(startInput.value);
    const endNum = parseInt(endInput.value);

    // 验证输入
    if (!isValidPositiveInteger(startInput.value)) {
        alert('请输入有效的起始数！');
        startInput.focus();
        return;
    }
    if (!isValidPositiveInteger(endInput.value)) {
        alert('请输入有效的终止数！');
        endInput.focus();
        return;
    }

    // 检查起始数是否大于终止数
    if (startNum > endNum) {
        alert('起始数不能大于终止数，请重新输入！');
        startInput.focus();
        return;
    }

    // 继续计算...
}
```

实现说明：

1. 输入格式验证：
   - 检查输入是否为空
   - 验证是否为正整数
   - 使用正则表达式确保格式正确
   - 防止小数和非数字输入

2. 数值范围验证：
   - 确保起始数不大于终止数
   - 提供清晰的错误提示
   - 自动聚焦到需要修改的输入框

3. 错误处理机制：
   - 分步验证，逐项检查
   - 即时反馈错误信息
   - 中断无效的计算过程

4. 用户体验优化：
   - 精确的错误提示信息
   - 自动焦点管理
   - 保留原有输入值

输入错误会弹出提示框：

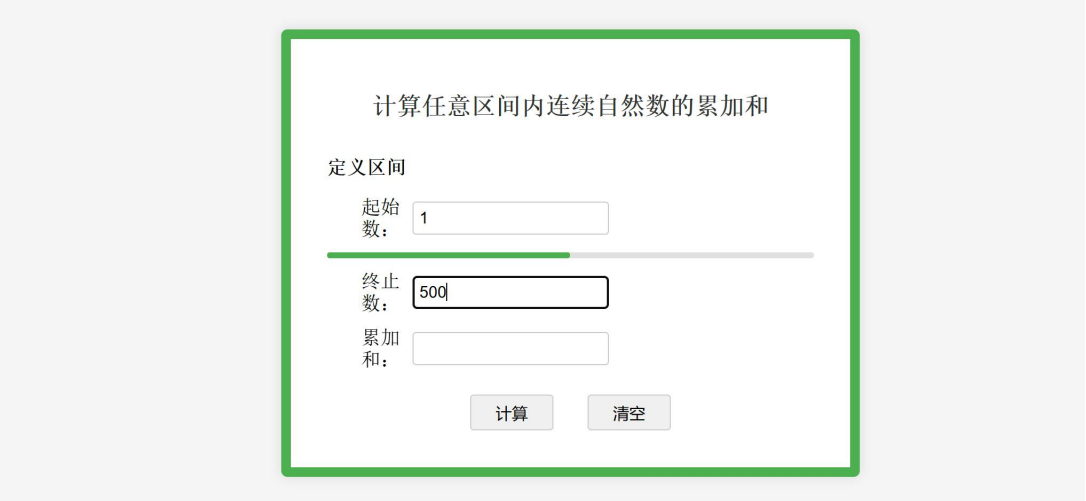
4.2 区间可视化实现

```javascript
function updateRangeIndicator() {
    const start = parseInt(document.getElementById('startNum').value) || 0;
    const end = parseInt(document.getElementById('endNum').value) || 0;

    if (isValidPositiveInteger(start.toString()) &&
isValidPositiveInteger(end.toString())) {
        const progress = document.querySelector('.range-progress');
        const width = ((end - start) / Math.max(end, 1000)) * 100;
        progress.style.width = Math.max(0, Math.min(width, 100)) + '%';
    }
}
```

进度条（范围指示器）的作用是直观地显示用户输入的两个数字之间的区间大小。

绿色的进度条就是范围指示器



实现说明：

1. 动态计算区间范围
2. 使用相对比例计算进度条宽度
3. 确保进度条不超出范围
4. 添加平滑过渡动画

4.3 累加和计算实现

```javascript
function sum(n1, n2) {
    // 确保n1小于等于n2
    if (n1 > n2) {
        [n1, n2] = [n2, n1];
    }
    // 使用等差数列求和公式：(首项 + 末项) × 项数 ÷ 2
    return (n1 + n2) * (n2 - n1 + 1) / 2;
}

function calculate() {
    const startInput = document.getElementById('startNum');
    const endInput = document.getElementById('endNum');
    const startNum = startInput.value;
    const endNum = endInput.value;

    // 验证输入
    if (!isValidPositiveInteger(startNum)) {
        alert('请输入有效的起始数！');
        startInput.focus();
        return;
    }
    if (!isValidPositiveInteger(endNum)) {
        alert('请输入有效的终止数！');
```

```
        endInput.focus();
        return;
    }

    // 计算并显示结果
    const result = sum(parseInt(startNum), parseInt(endNum));
    document.getElementById('result').value = result;
    document.getElementById('result').classList.add('result-animation');

    showProcess(parseInt(startNum), parseInt(endNum));
}
```

实现说明：

1. 使用等差数列求和公式
2. 自动处理数字顺序
3. 添加输入验证
4. 显示动画效果

4.4 计算过程展示

```javascript
function showProcess(n1, n2) {
    // 如果n1大于n2，交换它们的值
    if (n1 > n2) {
        [n1, n2] = [n2, n1];
    }

    const container = document.getElementById('processContainer');
    const steps = document.getElementById('processSteps');
    container.style.display = 'block';
    steps.innerHTML = '';

    // 显示公式和计算步骤
    const steps_info = [
        {text: `计算公式：(首项 + 末项) × 项数 ÷ 2`, type: 'formula'},
        {text: `(${n1} + ${n2}) × (${n2} - ${n1} + 1) ÷ 2`, type: 'substitution'},
        {text: `= (${n1 + n2}) × ${n2 - n1 + 1} ÷ 2`, type: 'step1'},
        {text: `= ${(n1 + n2) * (n2 - n1 + 1)} ÷ 2`, type: 'step2'},
        {text: `= ${(n1 + n2) * (n2 - n1 + 1) / 2}`, type: 'result'}
    ];

    steps_info.forEach((step, index) => {
        const div = document.createElement('div');
        div.className = 'process-step';
        div.textContent = step.text;
```

```
        steps.appendChild(div);

        // 添加渐进式高亮效果
        setTimeout(() => {
            div.classList.add('highlight');
        }, index * 300);
    });
}
```

实现说明：

1. 分步骤展示计算过程
2. 使用数组存储步骤信息
3. 动态创建 DOM 元素
4. 添加渐进式动画效果

5. 创新点

5.1 界面设计

- 简洁现代的 UI 设计
- 响应式布局适配
- 动态进度条可视化
- 平滑的动画过渡

5.2 功能创新

- 实时输入验证
- 区间范围可视化
- 计算过程动态展示

- 错误提示动画

5.3 技术创新

- 使用 ES6+新特性
- 模块化的代码组织
- 优雅的错误处理
- 高效的算法实现

整体效果展示：

项目 3　消息对话框综合应用

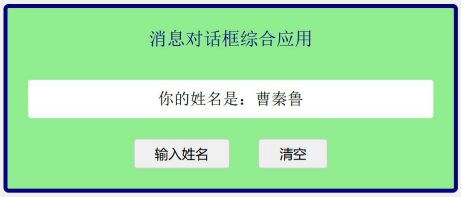
1. 项目概述

本项目是一个基于 Web 技术的消息对话框应用，实现了模态对话框、输入验证、警告提示等功能。项目采用纯前端技术栈，展示了现代 Web 交互设计的基本实现方法。

2. 功能特点

2.1 基础功能

- 姓名输入：通过模态对话框输入姓名
- 输入验证：确保输入不为空
- 结果显示：在主界面显示输入的姓名
- 清空功能：一键清除显示内容



2.2 交互功能　　创新

- 模态对话框：输入时遮罩背景
- 警告提示：输入为空时显示警告

- 键盘支持：支持回车键提交
- 焦点管理：自动聚焦输入框

3. 技术栈

- HTML5：页面结构和语义化标签
- CSS3：
  - Flexbox 布局
  - 过渡动画
  - 定位系统
  - 层叠管理
- JavaScript：
  - DOM 操作
  - 事件处理
  - 输入验证
  - 模态框控制

4. 实现思路详解

4.1 对话框实现

```css
.dialog {
    display: none;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background: white;
    padding: 20px;
    border: 1px solid #ccc;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    border-radius: 5px;
    min-width: 300px;
    z-index: 1000;
}

.overlay {
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0,0,0,0.5);
    z-index: 999;
```

```
}
```

实现说明：

1. 使用 fixed 定位将对话框固定在屏幕中央
2. 使用 transform 进行精确居中
3. 添加 z-index 确保对话框在最上层
4. 使用遮罩层创建模态效果

4.2 对话框控制

```javascript
function showInputDialog() {
    document.getElementById('inputDialog').style.display = 'block';
    document.getElementById('overlay').style.display = 'block';
    document.getElementById('nameInput').value = '';
    document.getElementById('nameInput').focus();
}

function closeDialog(dialogId) {
    document.getElementById(dialogId).style.display = 'none';
    document.getElementById('overlay').style.display = 'none';
}
```

实现说明：

1. 显示/隐藏对话框和遮罩层
2. 清空并聚焦输入框
3. 通用的关闭函数处理不同对话框

4.3 输入处理

```javascript
function submitName() {
    const nameInput = document.getElementById('nameInput');
    const name = nameInput.value.trim();

    if (!name) {
        showWarning('请你输入姓名！');
        return;
    }

    const displayName = document.getElementById('displayName');
    displayName.textContent = `你的姓名是：${name}`;
    displayName.style.display = 'block';

    closeDialog('inputDialog');
}
```

实现说明：

1. 获取并验证输入值
2. 显示警告或更新显示
3. 关闭输入对话框
4. 更新主界面显示

4.4 键盘支持

```javascript
document.getElementById('nameInput').addEventListener('keypress', function(e) {
    if (e.key === 'Enter') {
        submitName();
    }
});
```

实现说明：

1. 监听键盘事件
2. 支持回车键提交
3. 提升用户体验

5. 创新点

5.1 界面设计

- 简洁现代的 UI 设计
- 响应式布局适配
- 优雅的动画效果
- 清晰的视觉层次

5.2 交互创新

- 模态对话框设计
- 键盘快捷操作
- 焦点自动管理
- 实时输入验证

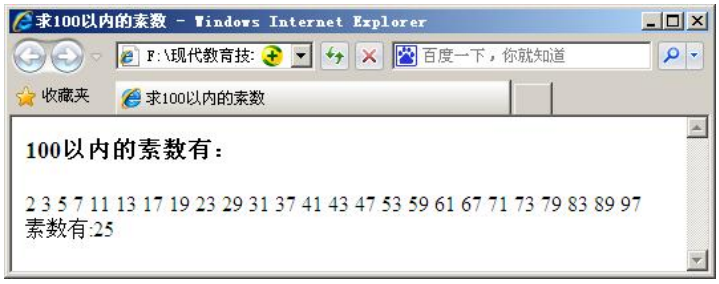
5.3 技术创新

- 组件化的代码组织
- 优雅的错误处理
- 灵活的对话框系统
- 可复用的功能模块

课外拓展训练

1. 编写 JavaScript 程序实现“求 100 以内的素数”

目标效果



我实现的效果：

1. 项目概述

本项目是一个基于 Web 技术的素数计算工具，能够自动计算并展示 100 以内的所有素数，并提供相关的统计信息。项目采用纯前端技术栈，实现了一个直观、高效的素数展示系统。

2. 功能特点

2.1 基础功能

- 自动计算：页面加载时自动计算 100 以内的素数
- 手动触发：支持通过按钮重新计算
- 结果展示：以卡片形式展示每个素数
- 统计分析：显示素数的数量、最大值、最小值和平均值

2.2 界面特点

- 响应式设计：适配不同屏幕尺寸
- 视觉反馈：按钮悬停效果
- 清晰布局：结果和统计信息分区显示
- 美观样式：现代化的卡片设计

3. 技术栈
- HTML5：页面结构和语义化标签
- CSS3：
  - Flexbox 布局
  - 盒子阴影效果
  - 圆角边框
  - 过渡动画
- JavaScript：
  - DOM 操作
  - 数组处理
  - 数学计算
  - 事件处理

4. 核心算法实现

4.1 素数判断算法
```javascript
function isPrime(num) {
    if (num < 2) return false;
    if (num === 2) return true;
    if (num % 2 === 0) return false;

    const sqrt = Math.sqrt(num);
    for (let i = 3; i <= sqrt; i += 2) {
        if (num % i === 0) return false;
    }

    return true;
}
```
算法优化说明：
1. 快速排除小于 2 的数字
2. 特殊处理数字 2
3. 排除偶数提高效率
4. 只检查到平方根
5. 步长为 2 减少循环次数

4.2 素数计算与展示
```javascript
function calculatePrimes() {
    const primes = [];
    for (let i = 2; i <= 100; i++) {
        if (isPrime(i)) {
            primes.push(i);
```

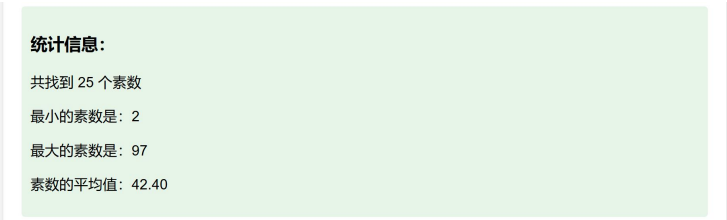
```
        }
    }

    // 显示结果
    resultDiv.innerHTML = '<h3>找到的素数：</h3>' +
        primes.map(num => `<span
class="prime-number">${num}</span>`).join(' ');

    // 显示统计
    statsDiv.innerHTML = `
        <h3>统计信息：</h3>
        <p>共找到 ${primes.length} 个素数</p>
        <p>最小的素数是：${primes[0]}</p>
        <p>最大的素数是：${primes[primes.length - 1]}</p>
        <p>素数的平均值：${(primes.reduce((a, b) => a + b, 0) /
primes.length).toFixed(2)}</p>
    `;
}
```

实现说明：

1. 使用数组存储素数
2. 使用 map 方法生成 HTML
3. 计算统计信息
4. 格式化显示结果



5. 样式设计

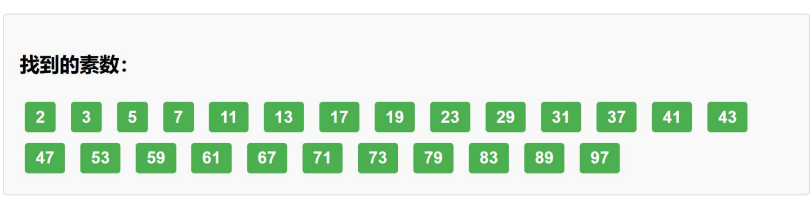
5.1 布局设计

```css
body {
    font-family: Arial, sans-serif;
    max-width: 800px;
    margin: 20px auto;
    padding: 0 20px;
    background-color: #f5f5f5;
}
```

设计说明：

1. 居中布局
2. 最大宽度限制
3. 适当的内外边距
4. 柔和的背景色

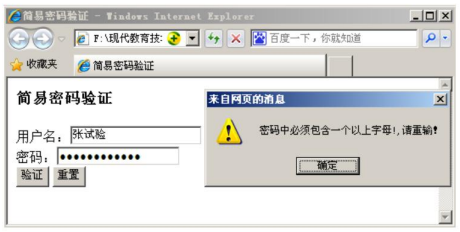
5.2 素数展示样式
```css
.prime-number {
    display: inline-block;
    padding: 5px 10px;
    margin: 5px;
    background-color: #4CAF50;
    color: white;
    border-radius: 3px;
    font-weight: bold;
}
```
设计说明：
1. 卡片式展示
2. 醒目的配色
3. 合适的间距
4. 圆角美化



结果展示：

2. 编写 JavaScript 程序实现简易密码验证

目标样式



我的效果展示：

1. 项目概述

本项目是一个基于 Web 技术的密码验证工具，提供实时的密码强度检测和规则验证功能。项目采用纯前端技术栈，实现了一个直观、用户友好的密码验证系统。

2. 功能特点

2.1 基础功能

- 密码输入：安全的密码输入框
- 规则验证：实时检查密码规则
- 强度显示：动态密码强度指示器
- 状态反馈：清晰的验证状态展示

2.2 验证规则 创新

- 长度要求：密码至少 8 位
- 字符组合：
  - 必须包含大写字母
  - 必须包含小写字母
  - 必须包含数字
  - 必须包含特殊字符（!@#$%^&*）

2.3 界面特点
- 响应式设计：适配不同屏幕尺寸
- 实时反馈：即时显示验证结果
- 视觉指示：使用图标和颜色区分状态
- 强度条：直观展示密码强度

3. 技术栈
- HTML5：
  - 语义化标签
  - 表单控件
  - 安全属性
- CSS3：
  - Flexbox 布局
  - 过渡动画
  - 伪元素
  - 响应式设计
- JavaScript：
  - DOM 操作
  - 事件处理
  - 正则表达式
  - 状态管理

4. 核心代码实现

4.1 密码验证规则
```javascript
const passwordRules = {
    length: password => password.length >= 8,
    uppercase: password => /[A-Z]/.test(password),
    lowercase: password => /[a-z]/.test(password),
    number: password => /[0-9]/.test(password),
    special: password => /[!@#$%^&*]/.test(password)
};
```
实现说明：
1. 使用对象存储验证规则
2. 每个规则返回布尔值

3. 使用正则表达式匹配字符
4. 函数式编程方法

4.2 密码强度验证
```javascript
function validatePassword(password) {
    let strength = 0;
    let validCount = 0;

    // 检查每个规则
    for (let rule in passwordRules) {
        const isValid = passwordRules[rule](password);
        requirements[rule].className = 'requirement ' +
(isValid ? 'valid' : 'invalid');
        if (isValid) {
            validCount++;
            strength += 20; // 每个规则占20%的强度
        }
    }

    // 更新强度条
    strengthBar.style.width = strength + '%';
    if (strength > 70) {
        strengthBar.style.backgroundColor = '#28a745';
    } else if (strength > 40) {
        strengthBar.style.backgroundColor = '#ffc107';
    } else {
        strengthBar.style.backgroundColor = '#dc3545';
    }

    // 所有规则都满足时启用提交按钮
    submitBtn.disabled = validCount !== 5;
}
```

实现说明：
1. 计算密码强度
2. 更新 UI 状态
3. 控制提交按钮
4. 动态更新强度条

5. 样式设计

5.1 密码强度条 创新
```css
```

```css
.strength-meter {
    height: 4px;
    background-color: #eee;
    margin: 10px 0;
    border-radius: 2px;
}

.strength-meter div {
    height: 100%;
    width: 0;
    background-color: #dc3545;
    border-radius: 2px;
    transition: all 0.3s;
}
```

设计说明：

1. 简洁的进度条设计
2. 平滑的过渡动画
3. 颜色反馈机制
4. 圆角美化效果

5.2 验证状态样式

```css
.requirement.valid {
    color: #28a745;
}
.requirement.valid::before {
    content: "✓ ";
    color: #28a745;
}
.requirement.invalid {
    color: #dc3545;
}
```

```
.requirement.invalid::before {
    content: "✗ ";
    color: #dc3545;
}
```

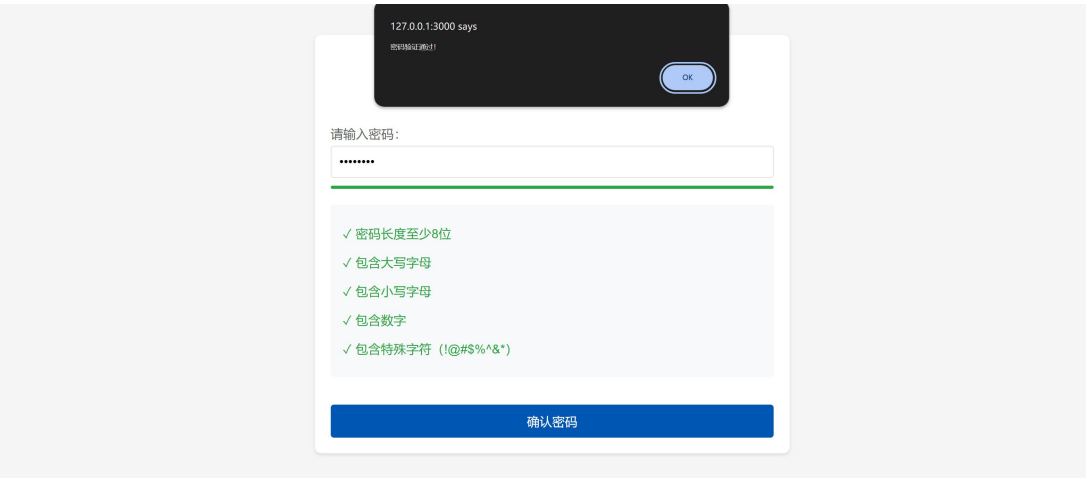
设计说明：
1. 使用伪元素添加图标
2. 颜色区分状态
3. 清晰的视觉反馈
4. 统一的样式主题

6. 使用说明
1. 打开页面，看到密码输入框和验证规则列表
2. 开始输入密码，观察实时反馈：
   - 规则验证状态实时更新
   - 密码强度条动态变化
   - 提交按钮状态自动切换
3. 满足所有规则后，提交按钮变为可用
4. 点击提交按钮完成验证

7. 安全特性
1. 输入安全：
   - 使用 password 类型输入框
   - 禁用自动完成功能
   - 提交后自动清空

2. 验证机制：
   - 客户端实时验证
   - 多重规则组合
   - 强度评估系统

效果展示：



实验收获：

技术要点总结

1. HTML5 应用
- 语义化标签的使用增强了代码可读性
- 表单控件的合理应用提升了用户体验
- 响应式设计确保了跨设备兼容性
- 模态框等现代 UI 组件的实现

2. CSS3 特性
- Flexbox 布局实现了灵活的页面结构
- 过渡动画增强了交互体验
- 伪元素的巧妙运用简化了标签结构
- 响应式设计适配各种屏幕尺寸
- 优雅的视觉反馈设计

3. JavaScript 技术
- DOM 操作实现动态内容更新
- 事件处理实现用户交互
- 正则表达式进行输入验证
- 模块化的代码组织提高可维护性
- 算法的优化实现

通过本次 Web 开发实验，我深入学习了前端开发的三大核心技术（HTML5、CSS3 和 JavaScript），并通过三个实际项目的开发加深了对这些技术的理解和应用。在消息对话框项目中，我学会了模态框的实现和用户交互处理；在素数计算器项目中，我掌握了 JavaScript 算法优化和数据展示技巧；在密码验证器项目中，我理解了实时验证和安全性考虑的重要性。通过这些项目，我不仅提升了编程技能，还学会了如何从用户体验的角度思考问题，如何进行代码组织和优化，以及如何编写规范的技术文档。最重要的是，我认识到了理论知识与实践相结合的重要性，体会到了解决实际问题时需要全面考虑各种因素。这次实验让我对 Web 开发产生了更浓厚的兴趣，也为我未来继续深入学习 Web 技术打下了坚实的基础。

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。