《Web 前端技术》课程设计

案例名称: 购物网站

班　　级：2022 级软件工程四班

学　　号: 121052022178

姓　　名：何凌强

二零二四年 十二 月 二十二 日

目录

一、前端页面规划和功能设计

1. 页面内容概要说明

本实验旨在开发一个基本的电子商务网站，用户可以浏览商品、将商品添加到购物车、修改购物车内容并最终提交订单。整个网站的功能模块包括：

商品浏览页面

该页面是用户进入网站后首先看到的页面，展示商品的分类和商品列表。用户可以通过点击商品或商品分类按钮，浏览不同类型的商品。每个商品项包含图片、名称、价格和“添加到购物车”按钮。

功能需求

动态加载商品列表，支持根据分类筛选商品。

每个商品项有图片、名称、价格，用户点击后可以将商品加入购物车。

页面底部加入背景宣传视频，提高用户的沉浸感和互动性。

页面右下角增加购物车页面和订单页面的导航按钮

购物车页面

该页面展示用户已添加到购物车的商品，用户可以查看每个商品的详细信息，并能修改商品数量、删除商品或继续购物。购物车底部显示所有商品的合计价格，用户可以点击“结算”按钮进入订单页面。

功能需求

支持显示购物车中商品的详细信息，包括商品名称、图片、价格、数量和总价。

用户可以修改商品数量或删除商品。

动态更新商品总价，并提供结算按钮，用户点击后进入订单页面。

页面右下角增加商品浏览页面和订单页面的导航按钮

订单页面

用户进入订单页面后，可以查看所有选中的商品及其价格，确认订单后可以提交。订单页面展示商品的详细信息，计算商品总价，并在页面底部提供“提交订单”按钮。

功能需求

动态展示购物车中的商品详细信息，包括商品名称、数量、单价、总价等。

计算并显示订单总价。

提供“提交订单”按钮，点击后显示订单成功提示。

页面右下角增加商品浏览页面和订单页面的导航按钮

2. 功能说明

商品浏览页面

商品列表展示：页面通过动态加载商品列表，显示每个商品的名称、图片和价格。商品支持分类展示，用户可以选择商品类别进行筛选。

商品操作：用户可以通过点击商品旁的“添加到购物车”按钮将商品加入购物车，购物车中的商品数量和总价会实时更新。

页面导航：用户可以通过点击右下角的“购物车”按钮进入购物车页面，点击右下角的“订单查看”按钮进入订单页面

购物车页面

商品显示：页面列出了用户已添加到购物车的所有商品。每个商品的详细信息包括商品图片、名称、单价、数量、总价。

数量修改：用户可以修改购物车中商品的数量，数量变化后会实时更新总价。

删除商品：用户可以点击删除按钮，将商品从购物车中移除。

总价计算：页面底部显示购物车中所有商品的总价格，并提供结算按钮。结算按钮引导用户进入订单页面。

页面导航：用户可以通过点击右下角的“订单查看”按钮进入订单页面，点击右下角的“首页”按钮返回商品浏览页面

订单页面

商品信息展示：用户在订单页面查看所有选中商品的详细信息，包括商品名称、数量、单价、总价。

订单总价：页面底部显示该订单的总金额。

提交订单：用户点击“提交订单”按钮后，系统将确认订单信息并显示订单成功提示。

页面导航：用户可以通过点击右下角的“购物车”按钮进入购物车页面，点击右下角的“首页”按钮返回商品浏览页面

3. 操作说明

商品浏览页面

点击**商品类别按钮**，可以过滤并显示对应类别的商品。

点击**添加到购物车按钮**，将商品添加到购物车。

用户可以继续浏览商品，点击不同类别按钮查看不同类型的商品。

用户可以点击**页面导航按钮**，进入不同页面

购物车页面

用户可以查看已添加的商品，

点击**删除按钮**，将商品数量从购物车中减一，数量变化后总价自动更新。

点击**结算按钮**，用户跳转到订单页面进行确认和支付。

用户可以点击**页面导航按钮**，进入不同页面

订单页面

用户确认订单详情，查看所有商品的名称、单价、数量和总价。

用户点击**提交订单按钮**，完成订单提交，页面显示“订单提交成功”或相关反馈。

用户可以点击**页面导航按钮**，进入不同页面

4. 技术实现概述

本实验的前端页面设计与功能开发主要通过以下技术实现：

HTML5：用于页面的基本结构和内容布局。

CSS3：用于页面的样式设计，确保页面响应式布局和美观。

JavaScript（ES6+）：用于页面交互和动态功能开发，特别是商品的动态加载、购物车更新、订单计算等。

DOM 操作：利用 JavaScript 操作 DOM 实现页面内容的动态更新，如购物车商品的数量修改、删除、总价更新等。

事件监听：通过 JavaScript 添加事件监听器，响应用户的点击、修改等操作，实现交互效果。

5. 页面之间的交互流程

商品浏览页面

用户进入首页时，页面展示所有商品，用户可通过点击不同的商品类别按钮筛选商品。

点击商品旁边的添加到购物车按钮，商品将被加入购物车并更新页面中的购物车图标。

购物车页面

用户点击购物车按钮后，进入购物车页面，展示购物车中的商品。

用户可以修改商品数量或删除商品，页面会实时更新总价。

用户点击结算按钮后，将商品数据提交单并跳转到订单页面。

订单页面

用户在订单页面查看所有已购买商品的详细信息，确认无误后点击提交订单按钮，

系统将提交订单并显示“订单提交成功”的提示。

二、前端页面设计

1. 设计思路

为提高用户体验，本实验将采用现代化的网页设计风格，简洁大方，功能清晰，确保用户能够顺畅地完成购物操作。整个页面设计着重于以下几个方面：

清晰的导航：在每个页面的顶部提供导航栏，帮助用户在商品浏览、购物车和订单页面之间轻松切换。

简洁的布局：通过弹性布局（Flexbox）让页面整洁，便于展示商品和其他信息。

响应式设计：确保页面在不同设备（如 PC、平板、手机）上都有良好的显示效果，适配不同的屏幕尺寸。

2. 页面布局设计

商品浏览页面

页头部分包括网站标题、导航栏，导航栏包含链接到购物车和订单页面。

背景视频位于页面顶部，用于展示商品宣传片或活动信息。

分类按钮位于宣传视频和商品的中间，用户点击按钮时，商品区域动态更新显示符合筛选条件的商品。

商品展示部分采用 Flexboxd 布局，每个商品在商品容器种可以灵活排列，展示商品图片、名称和价格。

购物车页面

页头部分与商品浏览页面一致

购物车商品列表采用 flex 垂直布局，每一行显示商品的图片、名称、数量、单价、总价和删除按钮。

购物车底部显示商品总价和结算按钮，用户点击结算后进入订单页面。

页面包含导航按钮。

订单页面

页头部分与其他页面一致，包含网站标题和导航栏。

订单信息部分通过 flex 垂直布局展示每个商品的详细信息（名称、数量、单价、总价）。

页面底部展示订单总金额，并提供提交订单按钮。

页面包含导航按钮。

3. 各部分内容及展示形式

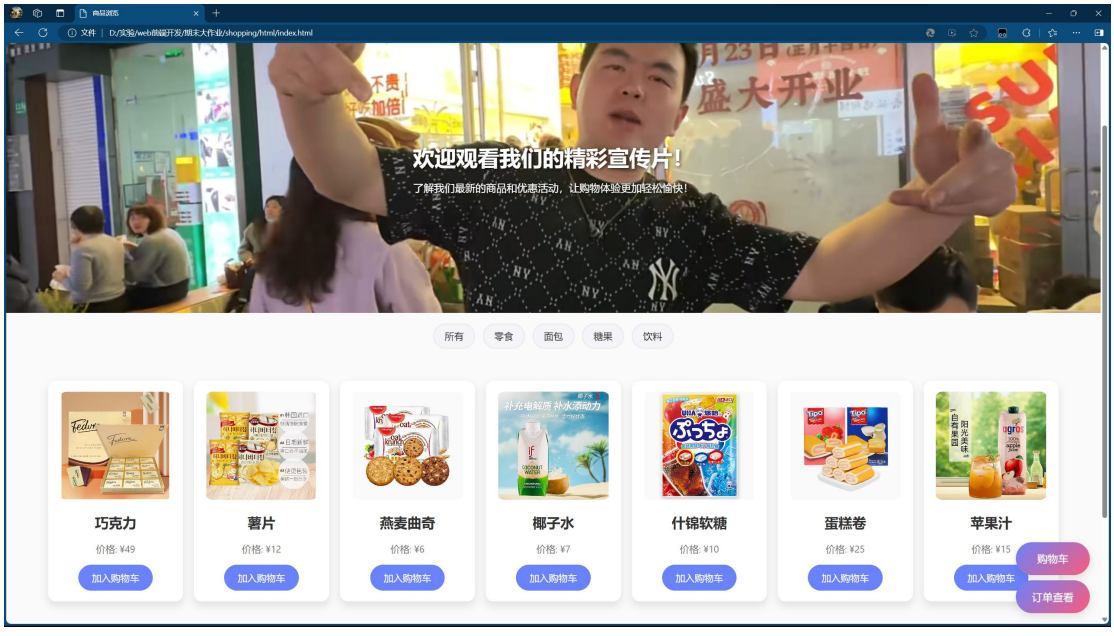
商品浏览页面

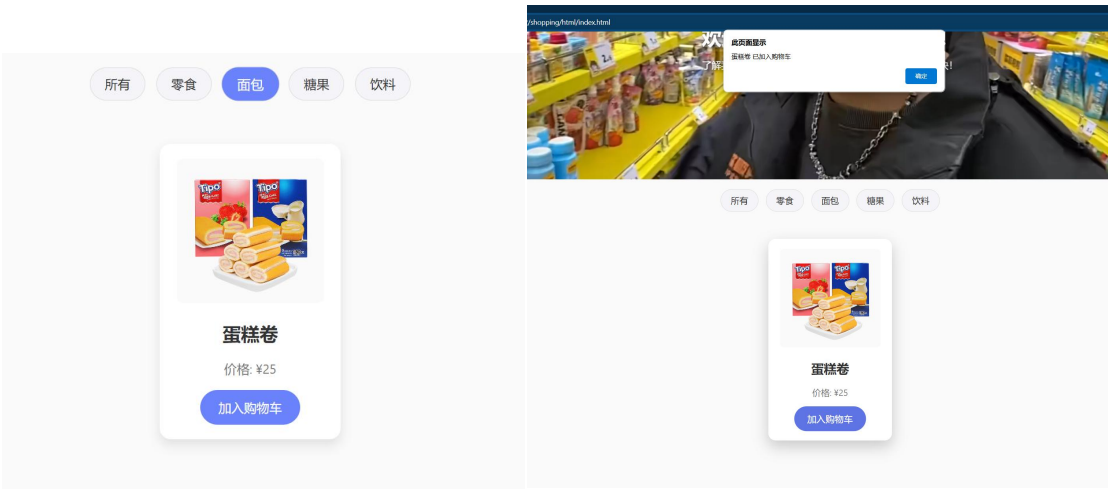
商品分类：使用<button>标签实现类别筛选，点击按钮时通过 JavaScript 动态更新商品展示区。

商品展示：使用 元素作为商品项的容器，每个商品项内部包括商品图片、名称、价格和“添加到购物车”按钮。

宣传视频：通过 <video> 标签来嵌入视频播放器。视频可以自动播放、循环播放并静音，以增强页面的互动性和吸引力。

导航按钮：按钮是固定悬浮在页面的右侧，使用 <a> 标签实现与其他页面的链接。当用户点击这些按钮时，会跳转到指定的页面，



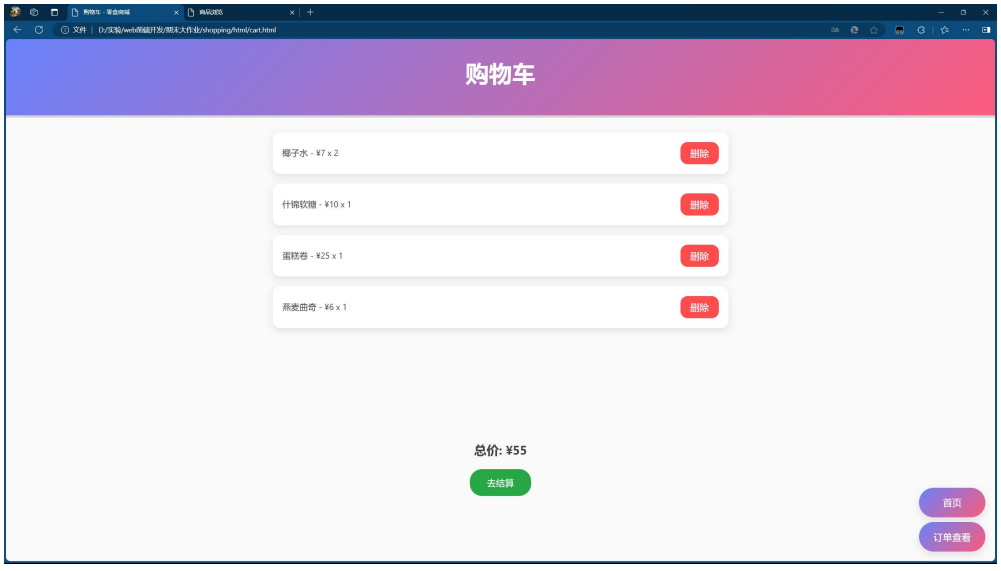


购物车页面

商品列表：通过<table>标签展示购物车中的商品，使用<tr>和<td>来显示商品的详细信息。

总价：通过 JavaScript 动态计算商品总价并显示在页面底部。

导航按钮：按钮是固定悬浮在页面的右侧或顶部，使用 <a> 标签实现与其他页面的链接。当用户点击这些按钮时，会跳转到指定的页面，



订单页面

订单信息：订单信息通过<ul>和<li>标签展示每个商品的详细信息。

订单总价：通过<p>标签展示总金额。

导航按钮：按钮是固定悬浮在页面的右侧或顶部，使用 <a> 标签实现与其他页面的链接。当用户点击这些按钮时，会跳转到指定的页面，

4. 颜色方案

Common.css

```css
/* 通用样式 */
/* 通配符选择器 */
* {
    margin: 0;
    /* 移除所有元素的默认外边距 */
    padding: 0;
    /* 移除所有元素的默认内边距 */
    box-sizing: border-box;
    /* 使元素的宽度和高度包括内边距和边框 */
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    /* 设置全局字体 */
}

/* 类选择器 */
body {
    background: #fafafa;
    /* 设置背景颜色为浅灰色 */
    color: #444;
    /* 设置文本颜色为深灰色 */
    font-size: 16px;
    /* 设置字体大小为 16 像素 */
    line-height: 1.6;
    /* 设置行高为 1.6 倍字体大小 */
}
```

```css
/* 头部样式 */
header {
    background: linear-gradient(135deg, #6a82fb, #fc5c7d);
    /* 设置背景为 135 度线性渐变 */
    padding: 30px 0;
    /* 设置上下内边距为 30 像素，左右为 0 */
    text-align: center;
    /* 设置文本居中对齐 */
    border-bottom: 5px solid #d1d1d1;
    /* 设置底部边框为 5 像素实线，颜色为浅灰色 */
}
```

```css
/* 后代选择器 */
header h1 {
    color: white;
    /* 设置标题颜色为白色 */
    font-size: 3em;
    /* 设置字体大小为 3 倍默认大小 */
    font-weight: 700;
    /* 设置字体粗细为 700（加粗） */
    margin-bottom: 20px;
    /* 设置底部外边距为 20 像素 */
}
```

```css
footer {
    background-color: #fafafa;
    /* 设置页脚背景颜色 */
    padding: 20px 0;
    /* 设置页脚上下内边距为 20 像素，左右为 0 */
    text-align: center;
    /* 设置页脚文本居中对齐 */
    /* border-top: 5px solid #d1d1d1; */
    /* 设置顶部边框为 5 像素实线，颜色为浅灰色 */
    font-size: 0.9em;
    /* 设置页脚字体大小 */
    color: #777;
    /* 设置页脚文本颜色 */
}
```

```css
footer p {
    text-align: center;
    /* 设置文本居中对齐 */
    margin-top: 200px;
}
```

```css
/* 页脚按钮样式 */
footer button {
    background-color: #28a745;
    /* 按钮背景色设置为绿色 */
    padding: 15px 35px;
    /* 设置按钮的内边距，上下 15px，左右 35px */
    font-size: 1.3em;
    /* 设置按钮文本的字体大小 */
```

```
    border-radius: 25px;
    /* 设置按钮圆角，形成圆形边角 */
    cursor: pointer;
    /* 鼠标悬停时显示为手型光标 */
    transition: background-color 0.3s ease, transform 0.3s ease;
    /* 设置按钮状态变化的过渡效果 */
}
```

```
/* 页脚按钮悬停效果 */
footer button:hover {
    background-color: #218838;
    /* 鼠标悬停时按钮背景色变为更深的绿色 */
    transform: scale(1.05);
    /* 鼠标悬停时按钮略微放大（缩放 1.05 倍） */
}
```

```
button {
    background-color: #6a82fb;
    /* 设置按钮背景颜色 */
    color: white;
    /* 设置按钮文字颜色 */
    border: none;
    /* 移除按钮边框 */
    padding: 12px 25px;
    /* 设置按钮内边距 */
    font-size: 1.1em;
    /* 设置按钮字体大小 */
    border-radius: 25px;
    /* 设置按钮圆角 */
    cursor: pointer;
    /* 设置鼠标指针为手型 */
    transition: background-color 0.3s ease, transform 0.3s ease;
    /* 设置背景颜色和变换的过渡效果 */
}
```

```
button:hover {
    background-color: #5e74e4;
    /* 鼠标悬停时改变按钮背景颜色 */
    transform: scale(1.05);
    /* 鼠标悬停时放大按钮 */
}
```

```css
/* 悬浮按钮样式 */
#floating-buttons {
    position: fixed;
    right: 20px;
    /* 距离右边 20px */
    bottom: 20px;
    /* 距离底部 20px */
    display: flex;
    flex-direction: column;
    /* 垂直排列按钮 */
    gap: 10px;
    /* 按钮间隔 10px */
    z-index: 1000;
    /* 确保按钮在最上层 */
}
```

```css
.floating-btn {
    background: linear-gradient(135deg, #6a82fb, #fc5c7d);
    /* 渐变背景：紫色到粉色 */
    color: white;
    /* 白色文字 */
    padding: 15px 30px;
    font-size: 1.2em;
    border-radius: 30px;
    /* 圆角更加圆润 */
    text-decoration: none;
    text-align: center;
    box-shadow: 0 6px 15px rgba(0, 0, 0, 0.1);
    /* 更柔和的阴影效果 */
    transition: background 0.3s ease, transform 0.3s ease, box-shadow 0.3s ease;
    display: block;
}
```

```css
/* 伪类选择器
hover： 选择鼠标悬停时的元素。
active： 选择用户点击时的元素。
focus：选择获得焦点的元素。
*/
```

```css
.floating-btn:hover {
    background: linear-gradient(135deg, #5e74e4, #f45c9d);
    /* 悬浮时的渐变色 */
    transform: scale(1.05);
```

```
    /* 放大效果 */
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.2);
    /* 更明显的阴影效果 */
}
```

```
.floating-btn:active {
    transform: scale(1);
    /* 按下时恢复原状 */
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
    /* 按下时减少阴影 */
}

/* 响应式布局 */
@media (max-width: 768px) {
    /* 当屏幕宽度小于 768px 时，进行布局调整 */
    #product-container {
        flex-direction: column;
        /* 商品列表展示改为垂直方向排列 */
        align-items: center;
        /* 商品列表容器内的商品项居中对齐 */
    }
    .product {
        width: 80%;
        /* 商品项宽度调整为原宽度的 80% */
    }
    #cart-container {
        padding: 20px;
        /* 购物车容器内边距调整为 20px */
    }
    .cart-item {
        flex-direction: column;
        /* 购物车项的布局改为垂直排列 */
        align-items: flex-start;
        /* 购物车项内容左对齐 */
    }
    footer {
        font-size: 1em;
        /* 页脚字体大小调整为 1em */
    }
    footer button {
        width: 100%;
        /* 页脚按钮宽度调整为 100% */
    }
}
```

Index. css

```
/* id 选择器 */
/* 商品列表展示容器样式 */
#product-container {
    display: flex;
    /* 使用 Flexbox 布局，使子元素可以灵活排列 */
    flex-wrap: wrap;
    /* 允许子元素在容器内换行 */
    justify-content: center;
    /* 将子元素在主轴（水平轴）上居中对齐 */
    height: fit-content;
    /* 使容器高度适应内容 */
    padding: 40px 20px;
    /* 为容器添加内边距，上下各 40 像素，左右各 20 像素 */
    gap: 20px;
    /* 子元素之间的间距为 20 像素 */
}

/* 商品列表展示 */
.product {
    background-color: white;
    /* 设置背景颜色为白色 */
    border-radius: 15px;
    /* 设置边框圆角为 15 像素 */
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
    /* 设置阴影效果 */
    width: 250px;
    /* 设置宽度为 250 像素 */
    padding: 20px;
    /* 设置内边距为 20 像素 */
    text-align: center;
    /* 设置文本居中对齐 */
    transition: all 0.3s ease-in-out;
    /* 设置过渡效果 */
    overflow: hidden;
    /* 隐藏溢出内容 */
}
```

```
/* 伪类选择器
hover：  选择鼠标悬停时的元素。
active：  选择用户点击时的元素。
focus：选择获得焦点的元素。
```

```
*/
.product:hover {
    transform: translateY(-5px);
    /* 鼠标悬停时向上移动 5 像素 */
    box-shadow: 0 12px 30px rgba(0, 0, 0, 0.2);
    /* 鼠标悬停时增加阴影效果 */
}
```

```
.product img {
    width: 100%;
    /* 设置图片宽度为 100% */
    height: 200px;
    /* 设置图片高度为 200 像素 */
    object-fit: cover;
    border-radius: 15px;
    margin-bottom: 15px;
}
```

```
.product h3 {
    font-size: 1.6em;
    /* 设置标题字体大小为 1.6 倍默认大小 */
    color: #333;
    /* 设置标题颜色为深灰色 */
    margin-bottom: 15px;
    /* 设置底部外边距为 15 像素 */
}
```

```
.product p {
    font-size: 1.1em;
    color: #777;
    margin-bottom: 15px;
}
```

```
/* 分类按钮样式 */
#category-container {
    display: flex;
    /* 使用 Flexbox 布局 */
    justify-content: center;
    /* 分类按钮水平居中对齐 */
    gap: 15px;
    /* 设置分类按钮之间的间距为 15px */
    padding: 20px 0;
```

```
    /* 上下内边距为 20px */
}
```

```
/* 分类按钮样式 */
#category-container button {
    background-color: #f4f4f9;
    /* 设置按钮的背景色为淡灰色 */
    color: #333;
    /* 设置字体颜色为深灰色 */
    border: 1px solid #ddd;
    /* 设置按钮边框为浅灰色 */
    padding: 10px 20px;
    /* 设置按钮内边距，上下 10px，左右 20px */
    font-size: 1.1em;
    /* 设置按钮字体大小为 1.1em */
    border-radius: 30px;
    /* 设置按钮圆角，圆角半径为 30px */
    cursor: pointer;
    /* 鼠标悬停时显示为手型光标 */
    transition: background-color 0.3s ease, transform 0.2s ease;
    /* 设置按钮状态变化的过渡效果 */
}
```

```
/* 分类按钮悬停效果 */
#category-container button:hover {
    background-color: #6a82fb;
    /* 鼠标悬停时按钮背景色变为蓝色 */
    color: white;
    /* 鼠标悬停时按钮字体颜色变为白色 */
    transform: scale(1.05);
    /* 鼠标悬停时按钮略微放大（缩放 1.05 倍） */
}
```

```
/* 背景视频部分 */
#promo-video {
    position: relative;
    /* 设置相对定位 */
    text-align: center;
    /* 设置文本居中对齐 */
    width: 100%;
    /* 设置宽度为 100% */
    height: 50vh;
    /* 设置高度为视口高度的 50% */
```

```
    overflow: hidden;
    /* 超出部分隐藏起来 */
    background-color: #000;
    /* 设置背景颜色为黑色 */
}
```

```
.video-container {
    position: absolute;
    /* 设置绝对定位 */
    top: 0;
    /* 设置顶部位置为 0 */
    left: 0;
    /* 设置左侧位置为 0 */
    width: 100%;
    /* 设置宽度为 100% */
    height: 100%;
    /* 设置高度为 100% */
}
```

```
video {
    position: absolute;
    /* 设置绝对定位 */
    top: 0;
    /* 设置顶部位置为 0 */
    left: 0;
    /* 设置左侧位置为 0 */
    width: 100%;
    /* 设置宽度为 100% */
    height: 100%;
    /* 设置高度为 100% */
    object-fit: cover;
    /* 让视频填充整个容器 */
    z-index: 1;
    /* 保证视频在所有元素的前面 */
}
```

```
/* 视频描述文本样式 */
.video-description {
    position: absolute;
    /* 设置绝对定位 */
    top: 50%;
    /* 设置顶部位置为 50% */
    left: 50%;
```

```
    /* 设置左侧位置为 50% */
    transform: translate(-50%, -50%);
    /* 通过平移使元素居中 */
    color: white;
    /* 设置文本颜色为白色 */
    text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);
    /* 设置文本阴影 */
    text-align: center;
    /* 设置文本居中对齐 */
    z-index: 1;
    /* 保证文本在视频的前面 */
}
```

```
.video-description h2 {
    font-size: 2.5em;
    /* 设置标题字体大小为 2.5 倍默认大小 */
    font-weight: bold;
    /* 设置标题字体加粗 */
    margin-bottom: 10px;
    /* 设置底部外边距为 10 像素 */
}
```

```
.video-description p {
    font-size: 1.2em;
    /* 设置段落字体大小为 1.2 倍默认大小 */
    line-height: 1.5;
    /* 设置行高为 1.5 倍字体大小 */
}
```

Cart.css

```css
/* id 选择器 */
/* 商品列表展示容器样式 */
#product-container {
    display: flex;
    /* 使用 Flexbox 布局，使子元素可以灵活排列 */
    flex-wrap: wrap;
    /* 允许子元素在容器内换行 */
    justify-content: center;
    /* 将子元素在主轴（水平轴）上居中对齐 */
    height: fit-content;
    /* 使容器高度适应内容 */
    padding: 40px 20px;
    /* 为容器添加内边距，上下各 40 像素，左右各 20 像素 */
    gap: 20px;
    /* 子元素之间的间距为 20 像素 */
}

/* 商品列表展示 */
.product {
    background-color: white;
    /* 设置背景颜色为白色 */
    border-radius: 15px;
    /* 设置边框圆角为 15 像素 */
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
    /* 设置阴影效果 */
    width: 250px;
    /* 设置宽度为 250 像素 */
    padding: 20px;
    /* 设置内边距为 20 像素 */
    text-align: center;
    /* 设置文本居中对齐 */
    transition: all 0.3s ease-in-out;
    /* 设置过渡效果 */
    overflow: hidden;
    /* 隐藏溢出内容 */
}
```

```
/* 伪类选择器
hover：  选择鼠标悬停时的元素。
active：  选择用户点击时的元素。
focus：选择获得焦点的元素。
```

```
*/
.product:hover {
    transform: translateY(-5px);
    /* 鼠标悬停时向上移动 5 像素 */
    box-shadow: 0 12px 30px rgba(0, 0, 0, 0.2);
    /* 鼠标悬停时增加阴影效果 */
}
```

```
.product img {
    width: 100%;
    /* 设置图片宽度为 100% */
    height: 200px;
    /* 设置图片高度为 200 像素 */
    object-fit: cover;
    border-radius: 15px;
    margin-bottom: 15px;
}
```

```
.product h3 {
    font-size: 1.6em;
    /* 设置标题字体大小为 1.6 倍默认大小 */
    color: #333;
    /* 设置标题颜色为深灰色 */
    margin-bottom: 15px;
    /* 设置底部外边距为 15 像素 */
}
```

```
.product p {
    font-size: 1.1em;
    color: #777;
    margin-bottom: 15px;
}
```

```
/* 分类按钮样式 */
#category-container {
    display: flex;
    /* 使用 Flexbox 布局 */
    justify-content: center;
    /* 分类按钮水平居中对齐 */
    gap: 15px;
    /* 设置分类按钮之间的间距为 15px */
    padding: 20px 0;
```

```
    /* 上下内边距为 20px */
}
```

```
/* 分类按钮样式 */
#category-container button {
    background-color: #f4f4f9;
    /* 设置按钮的背景色为淡灰色 */
    color: #333;
    /* 设置字体颜色为深灰色 */
    border: 1px solid #ddd;
    /* 设置按钮边框为浅灰色 */
    padding: 10px 20px;
    /* 设置按钮内边距，上下 10px，左右 20px */
    font-size: 1.1em;
    /* 设置按钮字体大小为 1.1em */
    border-radius: 30px;
    /* 设置按钮圆角，圆角半径为 30px */
    cursor: pointer;
    /* 鼠标悬停时显示为手型光标 */
    transition: background-color 0.3s ease, transform 0.2s ease;
    /* 设置按钮状态变化的过渡效果 */
}
```

```
/* 分类按钮悬停效果 */
#category-container button:hover {
    background-color: #6a82fb;
    /* 鼠标悬停时按钮背景色变为蓝色 */
    color: white;
    /* 鼠标悬停时按钮字体颜色变为白色 */
    transform: scale(1.05);
    /* 鼠标悬停时按钮略微放大（缩放 1.05 倍） */
}
```

```
/* 背景视频部分 */
#promo-video {
    position: relative;
    /* 设置相对定位 */
    text-align: center;
    /* 设置文本居中对齐 */
    width: 100%;
    /* 设置宽度为 100% */
    height: 50vh;
    /* 设置高度为视口高度的 50% */
```

```
    overflow: hidden;
    /* 超出部分隐藏起来 */
    background-color: #000;
    /* 设置背景颜色为黑色 */
}
```

```
.video-container {
    position: absolute;
    /* 设置绝对定位 */
    top: 0;
    /* 设置顶部位置为 0 */
    left: 0;
    /* 设置左侧位置为 0 */
    width: 100%;
    /* 设置宽度为 100% */
    height: 100%;
    /* 设置高度为 100% */
}
```

```
video {
    position: absolute;
    /* 设置绝对定位 */
    top: 0;
    /* 设置顶部位置为 0 */
    left: 0;
    /* 设置左侧位置为 0 */
    width: 100%;
    /* 设置宽度为 100% */
    height: 100%;
    /* 设置高度为 100% */
    object-fit: cover;
    /* 让视频填充整个容器 */
    z-index: 1;
    /* 保证视频在所有元素的前面 */
}
```

```
/* 视频描述文本样式 */
.video-description {
    position: absolute;
    /* 设置绝对定位 */
    top: 50%;
    /* 设置顶部位置为 50% */
    left: 50%;
```

```
    /* 设置左侧位置为 50% */
    transform: translate(-50%, -50%);
    /* 通过平移使元素居中 */
    color: white;
    /* 设置文本颜色为白色 */
    text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);
    /* 设置文本阴影 */
    text-align: center;
    /* 设置文本居中对齐 */
    z-index: 1;
    /* 保证文本在视频的前面 */
}
```

```
.video-description h2 {
    font-size: 2.5em;
    /* 设置标题字体大小为 2.5 倍默认大小 */
    font-weight: bold;
    /* 设置标题字体加粗 */
    margin-bottom: 10px;
    /* 设置底部外边距为 10 像素 */
}
```

```
.video-description p {
    font-size: 1.2em;
    /* 设置段落字体大小为 1.2 倍默认大小 */
    line-height: 1.5;
    /* 设置行高为 1.5 倍字体大小 */
}
```

Order.css

```css
/* 订单页面样式 */

/* 订单详情容器 */
#order-details {
    max-width: 1000px;
    /* 限制订单详情最大宽度为 1000px */
    margin: 30px auto;
    /* 上下间距 30px，左右自动居中 */
    padding: 20px;
    /* 内边距设置为 20px */
    background-color: #fff;
    /* 背景色为白色 */
    border-radius: 15px;
    /* 设置圆角，圆角半径为 15px */
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
    /* 添加阴影效果，使其浮动 */
}
```

```css
/* 订单项样式 */
.order-item {
    padding: 15px;
    /* 每个订单项的内边距设置为 15px */
    margin-bottom: 15px;
    /* 每个订单项的底部外边距为 15px */
    font-size: 1.1em;
    /* 字体大小设置为 1.1em */
    background-color: #f9f9f9;
    /* 背景色为浅灰色 */
    border-radius: 12px;
    /* 设置圆角，圆角半径为 12px */
    display: flex;
    /* 使用 Flexbox 布局 */
    justify-content: space-between;
    /* 在主轴上将内容分散对齐 */
    align-items: center;
    /* 在交叉轴上将内容居中对齐 */
    box-shadow: 0 3px 6px rgba(0, 0, 0, 0.1);
    /* 添加阴影效果 */
}
```

```css
/* 订单总价样式 */
#order-total {
    font-size: 1.6em;
    /* 字体大小设置为 1.6em */
    font-weight: 700;
    /* 设置为加粗 */
    text-align: center;
    /* 文字居中 */
    color: #444;
    /* 字体颜色为深灰色 */
    padding-top: 20px;
    /* 上边距为 20px */
    margin-top: 200px;
}
```

三、前端页面布局

1. 布局技术

Flexbox：用于商品展示区域和购物车商品列表的布局。Flexbox 布局简洁且易于响应式调整，适应不同屏幕宽度。

绝对定位：用于背景视频，确保元素显示在页面的最上方

固定定位：悬浮按钮，确保这些元素始终在页面右下方中显示。

2. 布局实现

商品浏览页面布局

背景视频、分类按钮、商品列表使用 Flexbox 布局进行展示。商品项以行列的形式展示，每个商品项包括图片、标题、价格以及“添加到购物车”按钮。

```
/* 背景视频部分 */
#promo-video {
    position: relative;
    /* 设置相对定位 */
    text-align: center;
    /* 设置文本居中对齐 */
    width: 100%;
    /* 设置宽度为 100% */
    height: 50vh;
    /* 设置高度为视口高度的 50% */
    overflow: hidden;
    /* 超出部分隐藏起来 */
    background-color: #000;
    /* 设置背景颜色为黑色 */
}
```

```css
/* 分类按钮容器样式 */
#category-container {
    display: flex;
    /* 使用 Flexbox 布局 */
    justify-content: center;
    /* 分类按钮水平居中对齐 */
    gap: 15px;
    /* 设置分类按钮之间的间距为 15px */
    padding: 20px 0;
    /* 上下内边距为 20px */
}
```

```css
/* 商品列表展示容器样式 */
#product-container {
    display: flex;
    /* 使用 Flexbox 布局，使子元素可以灵活排列 */
    flex-wrap: wrap;
    /* 允许子元素在容器内换行 */
    justify-content: center;
    /* 将子元素在主轴（水平轴）上居中对齐 */
    height: fit-content;
    /* 使容器高度适应内容 */
    padding: 40px 20px;
    /* 为容器添加内边距，上下各 40 像素，左右各 20 像素 */
    gap: 20px;
    /* 子元素之间的间距为 20 像素 */
}
```

购物车页面布局

使用 Flexbox 布局对商品信息进行排列，使商品的图片、名称、数量等信息整齐对齐。购物车合计部分居中显示。

```
/* 购物车页面 */
#cart-container {
    padding: 30px;
    /* 设置容器内边距 */
    max-width: 1000px;
    /* 设置容器最大宽度 */
    margin: 0 auto;
    /* 设置容器居中 */
    display: flex;
    /* 使用 Flexbox 布局 */
    flex-direction: column;
    /* 设置子元素垂直排列 */
    gap: 20px;
    /* 设置子元素之间的间距 */
}
```

订单页面布局

使用 flexbox 布局展示订单的商品列表，同时在页面的底部居中显示提交按钮。

```
/* 订单详情容器 */
#order-details {
    max-width: 1000px;
    /* 限制订单详情最大宽度为 1000px */
    margin: 30px auto;
    /* 上下间距 30px，左右自动居中 */
    padding: 20px;
    /* 内边距设置为 20px */
    background-color: #fff;
    /* 背景色为白色 */
    border-radius: 15px;
    /* 设置圆角，圆角半径为 15px */
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
    /* 添加阴影效果，使其浮动 */
}
```

四、前端页面开发实现

1. index.html

页面结构概述

商品浏览页面展示了商品列表和一个背景视频，用户可以在该页面选择商品类别，浏览商品信息，点击“添加到购物车”按钮将商品加入购物车。

主要部分：

头部区域（header）：含标题和背景

背景视频区域（#promo-video）：用于展示宣传视频，增加页面的互动性。

商品分类区域（#category-container）：用于筛选商品类别。

商品展示区域（#product-container）：商品的动态展示区，商品按照类别进行排列，显示图片、名称、价格和添加到购物车按钮。

导航按钮区域（#floating-buttons）：用于页面导航

底部区域（footer）：显示版权信息。

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>商品浏览</title>
    <link rel="stylesheet" href="../styles.css">
</head>
```

```
<body>
    <header>
        <h1>零食购物商城</h1>
    </header>
    <section id="promo-video">
        <div class="video-container">
            <!-- 背景视频播放器 -->
            <video id="promo" autoplay loop muted>
                <!-- autoplay: 自动播放
                    loop: 循环播放
                    muted: 静音 -->
                <source src="../video/宣传片.mp4" type="video/mp4">
                <!-- 如果浏览器不支持 video 标签，显示以下内容 -->
                您的浏览器不支持 HTML5 视频播放。
            </video>
            <!-- 视频描述文本 -->
            <div class="video-description">
                <h2>欢迎观看我们的精彩宣传片！</h2>
                <p>了解我们最新的商品和优惠活动，让购物体验更加轻松愉快！</p>
            </div>
        </div>
    </section>
    <!-- 商品分类导航栏 -->
    <div id="category-container" class="category-container">
        <button onclick="filterProducts('所有')">所有</button>
        <button onclick="filterProducts('零食')">零食</button>
        <button onclick="filterProducts('面包')">面包</button>
        <button onclick="filterProducts('糖果')">糖果</button>
        <button onclick="filterProducts('饮料')">饮料</button>
    </div>
    <!-- 商品展示区域 -->
    <section id="product-container" class="product-container">
        <!-- 动态生成商品列表 -->
    </section>
    <!-- 悬浮按钮 -->
    <div id="floating-buttons">
        <a href="cart.html" class="floating-btn">购物车</a>
        <a href="order.html" class="floating-btn">订单查看</a>
    </div>
    <footer>
        <p>© 2024 web 前端开发大作业</p>
    </footer>
    <script src="../js/product.js">
    </script>
```

```
    </body>
    </html>
```

Product.js

```
    // 商品数据（已更新）
    const products = [
        { id: 1, name: "巧克力", price: 49, img: "../image/巧克力.png", category: "零
食" },
        { id: 2, name: "薯片", price: 12, img: "../image/薯片.png", category: "零食" },
        { id: 3, name: "燕麦曲奇", price: 6, img: "../image/饼干.png", category: "零食
" },
        { id: 4, name: "椰子水", price: 7, img: "../image/椰子水.png", category: "饮料
" },
        { id: 5, name: "什锦软糖", price: 10, img: "../image/什锦软糖.png", category: "
糖果" },
        { id: 6, name: "蛋糕卷", price: 25, img: "../image/蛋糕卷.png", category: "面
包" },
        { id: 7, name: "苹果汁", price: 15, img: "../image/苹果汁.png", category: "饮
料" }
    ];

    // 获取商品展示容器
    const productContainer = document.getElementById("product-container");
    function renderProducts(filteredProducts) {
        productContainer.innerHTML = ''; // 清空当前商品展示区
        filteredProducts.forEach(product => {
            const productElement = document.createElement("div"); // 创建一个 div 元素
来展示商品
            productElement.classList.add("product"); // 为商品元素添加类名
            productElement.innerHTML = `
                <img src="${product.img}" alt="${product.name}">
                <h3>${product.name}</h3>
                <p>价格: ¥${product.price}</p>
                <button onclick="addToCart(${product.id}, '${product.name}',
${product.price})">加入购物车</button>
            `; // 设置商品的图片、名称、价格和加入购物车按钮
            productContainer.appendChild(productElement); // 将商品元素添加到商品展示容
器中
        });
    }
    // 将一个有效的 JSON 字符串解析为 JavaScript 对象
    let cart = JSON.parse(localStorage.getItem("cart")) || [];
    // 从本地存储中获取购物车数据，如果没有则初始化为空数组
```

```
    function addToCart(id, name, price) {
        const itemIndex = cart.findIndex(item => item.id === id); // 查找购物车中是否已有该商品
        // findIndex() 返回的是一个 整数索引值，从 0 开始
        if (itemIndex > -1) {
            cart[itemIndex].quantity++; // 如果已有该商品，增加其数量
        } else {
            cart.push({ id, name, price, quantity: 1 }); // 如果没有该商品，添加到购物车
        }
        // 将 JavaScript 对象转换为 JSON 字符串
        localStorage.setItem("cart", JSON.stringify(cart)); // 更新本地存储中的购物车数据
        alert(`${name} 已加入购物车`); // 提示用户商品已加入购物车
    }
    function filterProducts(category) {
        if (category === '所有') {
            renderProducts(products); // 渲染所有商品
        } else {
            const filteredProducts = products.filter(product => product.category === category);
            // 根据类别筛选商品
            renderProducts(filteredProducts); // 渲染筛选后的商品
        }
    }
    // 页面加载时默认展示所有商品
    window.onload = function () {
        renderProducts(products); // 默认展示所有商品
    };
```

2. cart.html

页面结构概述

购物车页面展示了用户已添加到购物车的商品，用户可以查看商品的详细信息、修改数量或删除商品，还可以查看总价并结算。

主要部分：

头部区域（header）：包含标题和背景。

购物车商品区域（#cart-items）：动态生成购物车中的商品列表，显示商品、名称、数量、单价、总价。

合计区域（#cart-total）：显示购物车中的商品总金额。

导航按钮区域（#floating-buttons）：用与页面导航

结算按钮：用户点击后进入订单页面。

Cart.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>购物车 - 零食商城</title>
    <link rel="stylesheet" href="../styles.css">
</head>
<body>
    <header>
        <h1>购物车</h1>
    </header>
    <section id="cart-container">
        <!-- 购物车商品列表展示 -->
    </section>
```

```
        <div id="total-price">
            总价: <span id="total">¥0</span>
        </div>
        <!-- 悬浮按钮 -->
        <div id="floating-buttons">
            <a href="index.html" class="floating-btn">首页</a>
            <a href="order.html" class="floating-btn">订单查看</a>
        </div>
        <footer>
            <button onclick="goToCheckout()">去结算</button>
        </footer>
        <script src="../js/cart.js"></script>
    </body>
    </html>
```

Cart.js

```
    .// 将一个有效的 JSON 字符串解析为 JavaScript 对象
    let cart = JSON.parse(localStorage.getItem("cart")) || [];
    // 从本地存储中获取购物车数据，如果没有则初始化为空数组

    function updateCart() {
        const cartContainer = document.getElementById("cart-container");
        cartContainer.innerHTML = ""; // 清空购物车容器
        let total = 0; // 初始化总价为 0
        cart.forEach(item => {
            const itemElement = document.createElement("div");
            itemElement.classList.add("cart-item"); // 为每个商品创建一个 div 元素并添加
类名
            itemElement.innerHTML = `
                <span>${item.name} - ¥${item.price} x ${item.quantity}</span>
                <button onclick="removeFromCart(${item.id})">删除</button>
            `; // 设置商品名称、价格、数量和删除按钮
            cartContainer.appendChild(itemElement); // 将商品元素添加到购物车容器中
            total += item.price * item.quantity; // 计算总价
        });
        document.getElementById("total").innerText = `¥${total}`; // 更新总价显示
    }
    function removeFromCart(id) {
        const itemIndex = cart.findIndex(item => item.id === id); // 查找要删除的商品索
引
        if (itemIndex > -1) {
```

```
            if (cart[itemIndex].quantity > 1) {
                // 如果商品数量大于 1，只减少数量
                cart[itemIndex].quantity--;
            } else {
                // 如果商品数量为 1，删除该商品
                cart.splice(itemIndex, 1);
            }
            // 将  JSON 对象转换为 JavaScript  字符串
            localStorage.setItem("cart", JSON.stringify(cart)); // 更新本地存储中的购物车数据
            updateCart(); // 更新购物车显示
        }
    }
    function goToCheckout() {
        if (cart.length === 0) {
            alert("购物车为空，无法结算!"); // 如果购物车为空，提示用户
            return;
        }
        // 将  JSON 对象转换为 JavaScript  字符串
        sessionStorage.setItem("order", JSON.stringify(cart)); // 将购物车数据保存到会话存储
        localStorage.removeItem("cart"); // 清空本地存储中的购物车数据
        window.location.href = "../html/order.html"; // 跳转到订单页面
    }
    window.onload = updateCart; // 页面加载时更新购物车显示
```

3. order.html

页面结构概述

订单页面展示了用户确认的订单内容，包含商品名称、数量、单价、总价等信息，用户可以在此页面提交订单

主要部分：

头部区域（header）：包含标题和背景。

订单详情区域（#order-details）：动态展示订单中所有商品的详细信息。

合计区域（#order-total）：显示订单总价。

提交按钮（#submit-order）：用户点击后提交订单。

导航按钮区域（#floating-buttons）：用与页面导航

Order.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>订单确认 - 零食商城</title>
    <link rel="stylesheet" href="../styles.css">
</head>
<body>
    <header>
        <h1>订单确认</h1>
    </header>
    <section id="order-details">
        <!-- 订单商品信息 -->
    </section>
```

```
        <div id="order-total">
            总价: <span id="orderTotal">¥0</span>
        </div>
        <!-- 悬浮按钮 -->
        <div id="floating-buttons">
            <a href="index.html" class="floating-btn">首页</a>
            <a href="cart.html" class="floating-btn">购物车</a>
        </div>
        <footer>
            <button onclick="submitOrder()">提交订单</button>
        </footer>
        <script src="../js/order.js"></script>
    </body>
    </html>
```

Order.js

```
    let order = JSON.parse(sessionStorage.getItem("order")) || [];
    // 从会话存储中获取订单数据，如果没有则初始化为空数组

    function renderOrder() {
        const orderDetails = document.getElementById("order-details");
        // 获取订单详情的容器元素
        let total = 0;
        // 初始化总价为 0
        if (order.length === 0) {
            orderDetails.innerHTML = "<p>您尚未选择商品。</p>";
            // 如果订单为空，显示提示信息
        } else {
            order.forEach(item => {
                const itemElement = document.createElement("div");
                // 为每个商品创建一个 div 元素
                itemElement.classList.add("order-item");
                // 添加类名
                itemElement.innerHTML = `${item.name} - ¥${item.price} x
${item.quantity}`;
                // 设置商品名称、价格和数量
                orderDetails.appendChild(itemElement);
                // 将商品元素添加到订单详情容器中
                total += item.price * item.quantity;
                // 计算总价
            });
```

```
            document.getElementById("orderTotal").innerText = `¥${total}`;
            // 更新总价显示
        }
    }
    function submitOrder() {
        if (order.length === 0) {
            alert("订单为空，无法提交!");
            // 如果订单为空，提示用户
            return;
        }
        alert("订单已提交！");
        // 提示用户订单已提交
        sessionStorage.removeItem("order");
        // 清空会话存储中的订单数据
        window.location.href = "../html/index.html";
        // 跳转到主页
    }
    window.onload = renderOrder;
    // 页面加载时渲染订单详情
```