Project 1: Gray Paper

Sammi Wong
Matthew Fried
CSCI355: Web and Internet Technology
October 15, 2023

HTML Files

index.html

```
<!--Logo, search bar, shopping cart button-->
<div id="main-header-content">
    <img src="images/logo.png" alt="logo" width="40px" height="40px">
    <a href="index.html" class="logo">S Mart</a>
    <a href="pages/shopping-cart.html" class="shopping-cart">&#128722;</a>
</div>
```

In the header tag, I created a div tag with contents including an image and two anchor tags. The image tag holds a source for the business log and the anchor tags contains reference to the homepage and shopping cart page, respectively. They're each given a name in relation to its purpose.

```
<nav id="main-navigation-bar">
    <!--Include each category-->
    <a href="pages/fresh-produce.html">Fresh Produce</a>
    <a href="pages/meat-and-seafood.html">Meat & Seafood</a>
    <a href="pages/dairy-and-eggs.html">Dairy & Eggs</a>
    <a href="pages/beverage.html">Beverage</a>
    <a href="pages/snack.html">Snack</a>
</nav>
```

The next section of code in the header tag includes a navigation tag that contains five anchor tags. Each anchor tag is a link to different categories of product being sold such as fresh produce, meat & seafood, dairy & eggs, beverage, and snack.

```
<article id="featured">
    <!--Featured (udemy Day 1 - Expanding cards)-->
    <h3>Featured</h3>
    <div class="featured-content">
        <div class="featured-container">
            <div class="panel active"
                style="background-image: url('images/fruits-and-vegetables-featured.jpg')">
                <h3>Fresh Produce</h3>
            </div>
            <div class="panel" style="background-image: url('images/meat-and-seafood-featured.jpg')">
                <h3>Meat & Seafood</h3>
            </div>
            <div class="panel" style="background-image: url('images/dairy-and-eggs-featured.jpg')">
                <h3>Dairy & Eggs</h3>
            </div>
            <div class="panel" style="background-image: url('images/beverage-featured.jpg')">
                <h3>Beverage</h3>
            </div>
            <div class="panel" style="background-image: url('images/snack-featured.jpg')">
                <h3>Snack</h3>
            </div>
        </div>
</article>
```

The next section is written in an article tag with an id of “featured”. I implemented ideas from Brad Traversy’s Udemy course day 1 on expanding cards. I replaced the background image with the pictures I’m using and replaced the header names.

```
<article id="weekly-deals">
    <!--Weekly deals-->
    <h3>Weekly Deals</h3>
    <article class="product-content">
        <div id="carrot" class="fresh-produce product-list">
            <img class="product-image" id="img-carrot" src="images/carrot.jpg">
            <div class="product-name">Carrot</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">34,121</span>
            <div class="price">$2.99</div>
        </div>
        <div id="apple" class="fresh-produce product-list">
            <img class="product-image" id="img-apple" src="images/apple.jpg">

            <div class="product-name">Apple</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">29,583</span>
            <div class="price">$0.99</div>
        </div>
        <div id="chicken" class="meat-and-seafood product-list">
            <img class="product-image" id="img-chicken" src="images/chicken.jpg">

            <div class="product-name">Chicken</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">34,121</span>
            <div class="price">$5.42</div>
        </div>
        <div id="watermelon" class="fresh-produce product-list">
            <img class="product-image" id="img-watermelon" src="images/watermelon.jpg">
            <div class="product-name">Watermelon</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">15,683</span>
            <div class="price">$3.99</div>
        </div>
    </article>
```

I created another section with an article tag. Within this tag, I created another article tag and assigned it with a class named “product-content”. Next, I created a div for each product being displayed within the weekly deals. Each product has an image tag to showcase the picture, a div tag to hold the product name, a span tag to display the rating as stars, another span tag to display the rating in numerical value, and a div tag for the product’s price. I assigned a class attribute for each of these tags since there are multiple products and I want to be able to change all of their properties simultaneously in CSS.

```
<footer>
    <!--faqs, -->
    <a href="pages/faqs.html">FAQs</a>
    <a href="pages/terms-of-services.html">Terms of Services</a>
    <a href="pages/about-us.html">About Us</a>


</footer>
```

At the bottom of the page, I used a footer tag to organize other pages regarding the business, including a faq, terms of services, and about us page. Each of these are placed within an anchor

tag as I want another page to be loaded when they are clicked. Each href value begins with "pages/" because I placed all the html pages in the page folder.

fresh-produce.html, meat-and-seafood.html, dairy-and-eggs.html, beverage.html, snack.html

The setup for the category product pages were all implemented in a similar fashion so I will only be discussing the fresh-produce page.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content=>
    <title>S Mart</title>
    <!--Favicon-->
    <link rel="icon" href="../images/logo.png">
    <link href="../styles/products.css" rel="stylesheet">
</head>
```

There is a head tag at the top of the file with meta tags and links to the stylesheet. The meta tag contains information about the page that will not be present in the window.

```
<header class="header">
    <!--Logo, search bar, shopping cart button-->
    <div class="products-header-content">
        <img src="../images/logo.png" alt="logo" width="40px" height="40px">
        <a href="../index.html" class="logo">S Mart</a>
        <a href="shopping-cart.html" class="shopping-cart">&#128722;</a>
    </div>
    <nav class="navigation-bar">
        <!--Include each category-->
        <a href="fresh-produce.html">Fresh Produce</a>
        <a href="meat-and-seafood.html">Meat & Seafood</a>
        <a href="dairy-and-eggs.html">Dairy & Eggs</a>
        <a href="beverage.html">Beverage</a>
        <a href="snack.html">Snack</a>
    </nav>
</header>
```

Within the header tag, I created a div tag and a navigation tag to contain the header and the navigation bar. This was implemented in the same way as the header and navigation bar in the homepage, but with slight adjustment to the page reference source. This is because each of the category product pages were placed within a page folder so reference to the homepage when clicking on the logo has to begin with "../".

```
<div class="content">
    <h3 class="sub-page-title">Fresh Produce</h3>
    <article class="fresh-produce-content product-content">
        <div id="cabbage" class="fresh-produce product-list">
            <img class="product-image" id="img-cabbage" src="../images/cabbage.jpg">
            <div class="product-name">Cabbage</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">64,187</span>
            <div class="price">4.99</div>
            <button class="add-to-cart-btn ripple">Add to Cart</button>
        </div>

        <div id="carrot" class="fresh-produce product-list">
            <img class="product-image" id="img-carrot" src="../images/carrot.jpg">
            <div class="product-name">Carrot</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">34,121</span>
            <div class="price">2.99</div>
            <button class="add-to-cart-btn ripple">Add to Cart</button>
        </div>

        <div id="eggplant" class="fresh-produce product-list">
            <img class="product-image" id="img-eggplant" src="../images/eggplant.jpg">
            <div class="product-name">Eggplant</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9734;</span>
            <span class="rating-count">63,374</span>
            <div class="price">1.99</div>
            <button class="add-to-cart-btn ripple">Add to Cart</button>
        </div>
```

```
<div id="tomato" class="fresh-produce product-list">
    <img class="product-image" id="img-tomato" src="../images/tomato.jpg">
    <div class="product-name">Tomato</div>
    <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
    <span class="rating-count">83,824</span>
    <div class="price">2.50</div>
    <button class="add-to-cart-btn ripple">Add to Cart</button>
</div>

<div id="lettuce" class="fresh-produce product-list">
    <img class="product-image" id="img-lettuce" src="../images/lettuce.jpg">
    <div class="product-name">Lettuce</div>
    <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
    <span class="rating-count">23,472</span>
    <div class="price">4.59</div>
    <button class="add-to-cart-btn ripple">Add to Cart</button>
</div>

<div id="corn" class="fresh-produce product-list">
    <img class="product-image" id="img-corn" src="../images/corn.jpg">
    <div class="product-name">Corn</div>
    <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
    <span class="rating-count">55,642</span>
    <div class="price">3.99</div>
    <button class="add-to-cart-btn ripple">Add to Cart</button>
</div>
```

```
<div id="bitterMelon" class="fresh-produce product-list">
    <img class="product-image" id="img-bitter-melon" src="../images/bitter-melon.jpg">
    <div class="product-name">Bitter Melon</div>
    <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
    <span class="rating-count">12,359</span>
    <div class="price">6.59</div>
    <button class="add-to-cart-btn ripple">Add to Cart</button>
</div>

<div id="gourd" class="fresh-produce product-list">
    <img class="product-image" id="img-gourd" src="../images/gourd.jpg">
    <div class="product-name">Gourd</div>
    <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9734;</span>
    <span class="rating-count">8,263</span>
    <div class="price">5.00</div>
    <button class="add-to-cart-btn ripple">Add to Cart</button>
</div>

<div id="cauliflower" class="fresh-produce product-list">
    <img class="product-image" id="img-cauliflower" src="../images/cauliflower.jpg">
    <div class="product-name">Cauliflower</div>
    <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
    <span class="rating-count">25,742</span>
    <div class="price">4.99</div>
    <button class="add-to-cart-btn ripple">Add to Cart</button>
</div>
```

```
        <div id="broccoli" class="fresh-produce product-list">
            <img class="product-image" id="img-broccoli" src="../images/broccoli.jpg">
            <div class="product-name">Broccoli</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">21,631</span>
            <div class="price">4.00</div>
            <button class="add-to-cart-btn ripple">Add to Cart</button>
        </div>

        <div id="apple" class="fresh-produce product-list">
            <img class="product-image" id="img-apple" src="../images/apple.jpg">
            <div class="product-name">Apple</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">29,583</span>
            <div class="price">0.99</div>
            <button class="add-to-cart-btn ripple">Add to Cart</button>
        </div>

        <div id="watermelon" class="fresh-produce product-list">
            <img class="product-image" id="img-watermelon" src="../images/watermelon.jpg">
            <div class="product-name">Watermelon</div>
            <span class="rating-star">&#9733;&#9733;&#9733;&#9733;&#9733;</span>
            <span class="rating-count">15,683</span>
            <div class="price">3.99</div>
            <button class="add-to-cart-btn ripple">Add to Cart</button>
        </div>

    </article>
```

In the next section, I created an article tag and assigned it with a class named “product-content”. Next, I created a div for each product being displayed under the fresh produce header. Each product has an image tag to showcase the picture, a div tag to hold the product name, a span tag to display the rating as stars, another span tag to display the rating in numerical value, a div tag for the product’s price, and a button for adding each product to the shopping cart. I assigned a class attribute for each of these tags since there are multiple products and I want to be able to change all of their properties simultaneously in CSS.

```
<footer>
    <!--faqs, -->
    <a href="faqs.html">FAQs</a>
    <a href="terms-of-services.html">Terms of Services</a>
    <a href="about-us.html">About Us</a>
</footer>
```

At the bottom of the page, I used a footer tag to organize other pages regarding the business, including a faq, terms of services, and about us page. Each of these are placed within an anchor tag as I want another page to be loaded when they are clicked.

```
<script src="../scripts/products.js"></script>
<script src="../scripts/sort.js"></script>
```

I also included two script tags to connect the page to the JavaScript pages for functionality purposes.

shopping-cart.html

The shopping cart page has a header and navigation bar that was implemented in the same way as the other pages.

```
<div class="content">
    <h3 class="shopping-cart-title">Shopping Cart</h3>
    <a href="shopping-cart.html" id="clear-cart">Clear Cart</a>
    <div class="table-content">
        <table id="shopping-cart-table">
            <tr>
                <th>Picture</th>
                <th>Item</th>
                <th>Price</th>
                <th>Quantity</th>
                <th>Subtotal</th>
            </tr>
        </table>
        <aside id="summary">
            <div id="total-cost-content">
                <!--Summary-->
                <h2 id="total-cost-title">Total Cost</h2>
                <div id="total-cost"></div>
                <a href="checkout.html" id="checkout">Proceed to Checkout</a>
            </div>
        </aside>
    </div>
</div>
```

I created a table tag and included an id attribute for “shopping-cart-table”. This is used to display all the products added to cart. I created five table headings, them bening picture, item, price, quantity, and subtotal. There is no table data in the table yet because nothing has been added to the cart yet. When a user adds a product to cart, entries would be created through Product.js file. More details regarding this will be discussed later on. Following the table is an aside tag. This is used to display the total cost of the items added to cart and an anchor tag that leads to the checkout page.

checkout.html

The checkout page has a header and navigation bar that was implemented in the same way as the other pages.

```
<section id="shipping-address" class="checkout-content">
    <h2 class="checkout-subTitles">1. Shipping Address</h2>
    <div id="shipping-address-description">
        <ul>
            <li>All fields cannot be left empty</li>
            <li>Email: Must contains an '@'</li>
            <li>Postcode: Must be 5 digits long</li>
        </ul>
    </div>
    <div id="shipping-address-content">
        <div class="shipping-address-info">
            <label for="firstNameShippingAddress">First Name *</label>
            <input type="text" name="firstNameShippingAddress" id="firstNameShippingAddress" required="required"
                class="shipping-address-input">
        </div>
        <div class="shipping-address-info">
            <label for="lastNameShippingAddress">Last Name *</label>
            <input type="text" name="lastNameShippingAddress" id="lastNameShippingAddress" required="required"
                class="shipping-address-input">
        </div>
        <div class="shipping-address-info">
            <label for="email">Email *</label>
            <input type="email" name="email" id="email" required="required" class="shipping-address-input">
        </div>
        <div class="shipping-address-info">
            <label for="streetAddress">Street Address *</label>
            <input type="text" name="streetAddress" id="streetAddress" required="required"
                class="shipping-address-input">
        </div>
        <div class="shipping-address-info">
            <label for="town-city">Town/City *</label>
            <input type="text" name="town-city" id="town-city" required="required"
                class="shipping-address-input">
        </div>
        <div class="shipping-address-info">
            <label for="postcode">Postcode *</label>
            <input type="text" name="postcode" id="postcode" required="required" class="shipping-address-input">
        </div>
        <div class="shipping-address-info">
            <label for="country">Country *</label>
            <input type="text" name="country" id="country" required="required" class="shipping-address-input">
        </div>
        <div class="shipping-address-info">
            <label for="state-county">State/County *</label>
            <input type="text" name="state-county" id="state-county" required="required"
                class="shipping-address-input">
        </div>
    </div>
</section>
```

For the checkout page, the first section is an unordered list with list items to inform the customer about the constraints when filling out the fields. The shipping address content is placed within a div tag. Each of the user's personal information is wrapped around a div tag. The first div tag is used to add the customer's first name; it has a "text" type so users can input information into it and I made it required so it cannot be left empty. Last name is implemented in a similar fashion. Next, I had a div tag for email and with type "email" so the input must include an "@" character. The following div tags are for street address, town/city, postcode, country, and state/county which all have a "text" type and a required attribute.

```
<section id="payment-method" class="checkout-content">
    <h2 class="checkout-subTitles">2. Payment Method</h2>
    <div id="payment-method-description">
        <ul>
            <li>All fields cannot be left empty</li>
            <li>Credit card number: Must be 16 digits long</li>
            <li>Security code: Must be 3 digits long</li>
        </ul>
    </div>
    <div id="payment-method-content">
        <div class="payment-method-info">
            <input type="radio" name="payment-type" value="credit-card" checked="checked"
                class="payment-type">Credit Card
            <input type="radio" name="payment-type" value="paypal" class="payment-type">PayPal
        </div>
        <div class="payment-method-info">
            <label for="firstNamePaymentMethod">First Name *</label>
            <input type="text" name="firstNamePaymentMethod" id="firstNamePaymentMethod" required="required"
                class="payment-method-input">
        </div>
        <div class="payment-method-info">
            <label for="lastNamePaymentMethod">Last Name *</label>
            <input type="text" name="lastNamePaymentMethod" id="lastNamePaymentMethod" required="required"
                class="payment-method-input">
        </div>
        <div class="payment-method-info">
            <label for="creditCardNumber">Credit Card Number *</label>
            <input type="text" name="creditCardNumber" id="creditCardNumber" required="required"
                class="payment-method-input">
        </div>
        <div class="payment-method-info">
            <label for="securityCode">Security Code *</label>
            <input type="text" name="securityCode" id="securityCode" required="required"
                class="payment-method-input">
        </div>
        <div class="payment-method-info">
            <label for="cardExpiration">Card Expiration *</label>
            <input type="date" name="cardExpiration" id="cardExpiration" required="required"
                class="payment-method-input">
        </div>
</section>
```

The next section is the payment section. Similar to the previous section I used an unordered list with list items to inform the customer about the constraints when filling out the fields. The first div tag uses a radio button for customers to select between PayPal and credit card payment. Subsequently, I included div tags for first name, last name, credit card number, and security code which all have a “text” type and a required attribute. Lastly, I included an input tag for card expiration with “date” type and a required attribute.

```
<section id="Confirm Order" class="checkout-content">
    <h2 class="checkout-subTitles">3. Confirm Order</h2>
    <input type="checkbox" name="updates" value="updates" id="updates-text">
    <label for="updates" id="updates-text">Receive updates and notifications?</label>
    <div id="checkout-links">
        <a href="../index.html" id="home">Continue Shopping</a>
        <a href="purchase-loading.html" id="purchaseBtn">Purchase</a>
    </div>
</section>
```

The next section includes an input tag with “checkbox” type. I decided to use a checkbox as opposed to a radio type so users can deselect it. Following it are two anchor tags, one that returns to the home page and another to proceed to the purchase loading page.

purchase-loading.html

```
<body>
    <div class="kinetic"></div>
    <div id="loading-description-content">
        <div class="loading-description">Purchasing...</div>
        <div class="loading-description">Do Not Refresh</div>
    </div>
</body>
```

In the body tag of the purchase-loading page, I borrowed ideas from Brad Traversy's Udemy course on kinetic loader. I added two div tags to display a "Purchasing" and "Do Not Refresh" text.

thank-you.html

```
<body>
    <div id="content">
        <h1 id="thank-you-text">Thank You for Your Purchase!</h1>
        <a href="../index.html" id="home-page-link">Continue Shopping</a>
    </div>
</body>
```

In the body tag of the thank-you page, I included header 1 and anchor tag wrapped around a div tag.

about-us.html

```
<div class="content">
    <h3 class="about-us-title">About Us</h3>
    <div id="author">By: Sammi Wong</div>
    <div id="counter-content">
        <div class="counter-container">
            <i class="fab fa-twitter fa-3x"></i>
            <div class="counter" data-target="12000"></div>
            <span>Twitter Followers</span>
        </div>

        <div class="counter-container">
            <i class="fab fa-youtube fa-3x"></i>
            <div class="counter" data-target="5000"></div>
            <span>YouTube Subscribers</span>
        </div>

        <div class="counter-container">
            <i class="fab fa-facebook fa-3x"></i>
            <div class="counter" data-target="7500"></div>
            <span>Facebook Fans</span>
        </div>
    </div>
```

In the div tag of the about-us page, I borrowed ideas from Brad Traversy's Udemy course on incrementing counters. This includes a data-target attribute so the value will be incremented to that value. Each of the div tags also include a span tag to include the text on the same line.

```
<div>
    <ul id="about-us-content">
        <li> Welcome to S Mart, where passion meets produce and community flourishes! At our
            store, we're more than just a place to grab your groceries - we're a family committed to
            bringing
            you
            the freshest, finest, and friendliest shopping experience.
        </li>
        <li>
            Founded with a vision to create a hub of culinary delight, S Mart sprouted from a
            love for quality ingredients and a desire to nourish our community. We believe in the power of
            good
            food to bring people together, share stories, and create lasting memories.
        </li>
        <li>
            Every apple, tomato, and loaf of bread at S Mart is handpicked with care. Our
            commitment to freshness extends from our local farmers to your family's table. We're not just
            providing groceries; we're delivering a promise of quality.
        </li>
        <li>
            We're not just a store; we're your neighbors, friends, and fellow food enthusiasts. Engaging
            with
            our community is at the heart of what we do. Join us for events, cooking classes, and tastings
            as we
            celebrate the diverse flavors that make our community unique.
        </li>
        <li>
            As stewards of the environment, we're on a mission to reduce our ecological footprint. From
            eco-friendly packaging to supporting sustainable farming practices, S Mart is committed to
            nourishing the planet as much as our customers.
        </li>
    </ul>
</div>
```

The next section is information about the business. I implemented this through an unordered list with an id attribute of "about-us-content" and multiple listed items about the business.

faqs.html

```
<div id="faqs-content">
    <h1>Frequently Asked Questions</h1>
    <div class="faq-container">
        <div class="faq active">
            <h3 class="faq-title">
                Is is possible for products to be refunded?
            </h3>

            <p class="faq-text">
                Depending on the circumstance, products can be refunded within two hours of purchase.
            </p>

            <button class="faq-toggle">
                <i class="fas fa-chevron-down"></i>
                <i class="fas fa-times"></i>
            </button>
        </div>

        <div class="faq">
            <h3 class="faq-title">
                When will I receive the order?
            </h3>
            <p class="faq-text">
                Order typically arrives in five business days.
            </p>
            <button class="faq-toggle">
                <i class="fas fa-chevron-down"></i>
                <i class="fas fa-times"></i>
            </button>
        </div>

        <div class="faq">
            <h3 class="faq-title">
                Is there a physical location of the store?
            </h3>
            <p class="faq-text">
                No, S Mart is a fully online grocery service.
            </p>
            <button class="faq-toggle">
                <i class="fas fa-chevron-down"></i>
                <i class="fas fa-times"></i>
            </button>
        </div>

        <div class="faq">
            <h3 class="faq-title">
                How can I get involved?
            </h3>
            <p class="faq-text">
                S Mart is currently not looking for any employees, thank you for your interest.
            </p>
            <button class="faq-toggle">
                <i class="fas fa-chevron-down"></i>
                <i class="fas fa-times"></i>
            </button>
        </div>

        <div class="faq">
            <h3 class="faq-title">
                Will holidays affect my food arrival time?
            </h3>
            <p class="faq-text">
                No, S Mart delivers even on holidays.
            </p>
            <button class="faq-toggle">
                <i class="fas fa-chevron-down"></i>
                <i class="fas fa-times"></i>
            </button>
        </div>
    </div>
```

I referenced Brad Traversy's Udemy course on faq-collapse to implement a frequently asked question page. Each question card includes a header 3 which contains a question, a paragraph tag for the answer, and two empty italic tags used for the drop down and close symbol. They all have class attributes so their properties can be accessed easily in CSS and JavaScript.

terms-of-services.html

```
<article>
    <h3 id="terms-of-services-title">Terms of Services</h3>
    <ol id="terms-of-services">
        <li>By accessing or purchasing products from S Mart, you agree to comply with and be bounded by these
            terms of services.</li>
        <li>All orders are subject to availability and acceptance. S Mart reserves the right to refuse or cancel
            any order for any reason.</li>
        <li>We strive to provide high-quality products. If you receive a product that is damaged or defective,
            please contact us within 5 days for a replacement or refund.</li>
        <li>You agree not to engage in any conduct that may disrupt or interfere with the normal operation of S
            Mart, including but not limited to hacking, transmitting viruses, or violating any applicable laws.
        </li>
        <li>Our privacy policy governs the collection, use, and disclosure of your personal information. By
            Using our services, you consent to the terms of our privacy policy.</li>
        <li>S Mart reserves the right to modify these terms of services at any time. It is your responsbility to
            review these terms periodically for changes.</li>
        <li>These terms of services are governed by and construed in accordance with the laws.</li>
        <li>If you have an questions of concerns about these terms of services, please contact us.</li>
    </ol>
</article>
```

In the terms of services page, it displays a list of terms when using the website. I implemented through an ordered list tag so the marks would be numbered.

CSS Files

home-page.css

```
/*Navaigation Bar*/
#main-header {
    position: fixed;
    left: 0;
    top: 0;
    background-color: #6ab767;
    width: 100%;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    display: flex;
    flex-direction: column;
    gap: 10px;
    justify-content: center;
    align-content: center;
    z-index: 100;

}
```

Used on the tag with “main-header” as its id, which is the header and navigation bar. Position is set to fixed so it would remain present at the top of the page even if the user is scrolling. It has a display of flex because there I want the header and navigation to be stacked on top of one another. I centered the content so the content would be in the center. I set the z-index to 100 so it would be stacked in front of all other content on the page.

```
#main-header-content {
    display: flex;
    justify-content: space-between;
    gap: 10px;
}
```

The selector is referring to the header at the very top of the page. It has a flex display of flex since there is only one row (default). Justify content is set to space between so the content within is spaced evenly. It has a gap of value 10px, indicating the space between.

```
.logo {
    font-size: 20px;
    text-align: center;
    color: #597d37;
    text-decoration: none;
    font-size: 30px;
}
```

This is the class selector referring to the word “S Mart” in the middle of the header. I adjusted the font-size, color, and set its text-decoration to none to remove the underline below the link.

```
.shopping-cart {
    text-align: center;
    color: white;
    text-decoration: none;
    font-size: 30px;
    transform: scaleX(-1);
}
```

This selector is referring to the cart image in the header. I centered the image, changed its color to white, removed the underline beneath the link, and set its font-size. I also included a transform property with a value of scaleX(-1) to flip the cart symbol.

```
#main-navigation-bar {
    display: flex;
    justify-content: space-evenly;
    gap: 10px;
}
```

The selector is referring to the navigation bar. It has a flex display of flex since there is only one row (default). Justify content is set to space between so the content within is spaced evenly. It has a gap of value 10px, indicating the space between.

```
#main-navigation-bar>a {
    background-color: #75bd85;
    width: 20%;
    font-size: 20px;
    text-align: center;
    color: white;
    text-decoration: none;
    padding: 10px;
}
```

The selector is referring to the links that are the direct child of the main-navigation-bar. I changed the background color, width, font size, centered the text, the font color, removed the underline, and added a 10px padding.

```
#main-navigation-bar>a:hover {
    background-color: #bddfcf;
}
```

Each time the product categories are hovered over, the background-color would be changed. This way, customers can easily tell that they are hovering over the link.

```
/*Main content*/
#main-content {
    position: relative;
    top: 100px;
    display: flex;
    flex-direction: column;
    gap: 10px;
    justify-content: center;
    align-content: center;
    text-align: center;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    color: #597d37;

}
```

The selector is referring to everything that follows the header and navigation bar. It has a property of position set to relative so I can adjust its location in relation to the container it's in. Property top of 100px moves the content 100px down so it would not be blocked by the

navigation bar. It has a display of flex so the contents within such as featured and weekly deals would be displayed in a column. It is centered vertically and horizontally with align-content and justify-content. I used Impact as the font and changed the color.

```
#shop-title {
    justify-content: center;
    font-size: 100px;
    background-image: url("../images/cover-page-background.jpg");
    height: 300px;
}
```

This refers to the container that holds the shop title. I centered the text and set the size to 100px. I included a background image which is found in the image folder and set the height to 300px.

```
article {
    height: 100%;
    width: 100%;
    background-color: #cfe3be;
    font-size: 30px;

}
```

This refers to all article tags. I set the height to 100% and the width to 10% so it would extend to its max width and height within its container.

```
/*featured*/

@import url('https://fonts.googleapis.com/css?family=Muli&display=swap');

* {
    box-sizing: border-box;
}

.featured-content {
    font-family: 'Muli', sans-serif;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 100vh;
    overflow: hidden;
    margin: 0;
}

.featured-container {
    display: flex;
    width: 90vw;
}

.panel {
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    height: 80vh;
    border-radius: 50px;
    color: #fff;
    cursor: pointer;
    flex: 0.5;
    margin: 10px;
    position: relative;
    transition: all 700ms ease-in;
}
```

```
.panel h3 {
    font-size: 24px;
    position: absolute;
    bottom: 20px;
    left: 20px;
    margin: 0;
    opacity: 0;
}

.panel.active {
    flex: 5;
}

.panel.active h3 {
    opacity: 1;
    transition: opacity 0.3s ease-in 0.4s;
}

@media (max-width: 480px) {
    .container {
        width: 100vw;
    }

    .panel:nth-of-type(4),
    .panel:nth-of-type(5) {
        display: none;
    }
}
```

This is the CSS information used in Brad Traversy's Expanding Card video. I used it to display the images in my featured section so the images would expand when clicked.

```css
/*Weekly Deals*/
.product-content {
    display: grid;
    grid-template-columns: auto auto auto auto;
    grid-template-rows: auto auto auto;
    gap: 20px;
    background-color: #cfe3be;
    padding: 20px;
}

.product-list {
    background-color: white;
    padding: 10px;
    text-align: center;
    height: 200px;
    box-shadow: 5px 5px 5px #5a9b57;
    border-radius: 5%;
}

.product-image {
    width: 30%;
    height: 30%;
}

.product-name {
    font-size: 30px;
}
```

```css
.rating-star {
    color: #fdfd96;
    text-shadow: 0 0 3px orange;
}

.rating-count {
    color: black;
    font-size: 20px;
}

.price {
    font-size: 25px;
    color: black;
}
```

This CSS section impacts the layout of my weekly deals. The class product-content has a grid display with four columns and 3 rows. However, only the first 4 boxes in the first row are filled because I only have four products within this section. Product-list class refers to the individual products. I changed the background color to white, aligned the text, set a box shadow, and gave it a border-radius property to make the border round. The other selectors refer to each information regarding the product such as its font size and color. I used the class selector so the information can be adjusted for all products with these classes.

```css
/*Footer*/
footer {
    background-color: #75bd85;
    width: 100%;
    font-size: 20px;
    text-align: center;
    padding: 10px;
    display: flex;
    justify-content: space-evenly;
}
```

This is referring to the footer tag. I added properties to change its background color, width, font and size. I also added a flex display so the links within can be layouted horizontally.

```css
footer>a {
    text-decoration: none;
    color: white;
}
```

This selector is referring to all anchor tags that are the direct child of a footer tag. I set the text decoration to none so there wouldn't be an underline beneath the links and changed the font color to white.

products.css

The selectors and properties used in this file are almost the same as the ones used in the homepage.css file for the header, navigation bar, product content, and footer with slight modifications based on the class and id names used.

```css
/*Button ripple effect*/
button {
    background-color: #bddfcf;
    color: #fff;
    border: 3px #6ab767 solid;
    font-size: 14px;
    text-transform: uppercase;
    letter-spacing: 2px;
    padding: 10px 10px;
    overflow: hidden;
    position: relative;
    border-radius: 10%;
    cursor: pointer;
}

button:focus {
    outline: none;
}

button .circle {
    position: absolute;
    background-color: #597d37;
    width: 500px;
    height: 500px;
    border-radius: 50%;
    transform: translate(-100%, -100%) scale(0);
    animation: scale 0.5s ease-out;
}

@keyframes scale {
    to {
        transform: translate(-50%, -50%) scale(3);
        opacity: 0;
    }
}
```

This selector was based on Brad Traversy's button ripple effect. It allows for some effect to occur when the add to cart button is clicked. @keyframes is an animation selector that allows the circle to spread out when the button is clicked.

shopping-cart.css

The selectors and properties used in this file are almost the same as the ones used in the files for the header and navigation bar with slight modifications based on the class and id names used.

```css
.content {
    position: relative;
    top: 100px;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    color: #597d37;
}
```

The selector is referring to everything that follows the header and navigation bar. It has a property of position set to relative so I can adjust its location in relation to the container it's in. Property top of 100px moves the content 100px down so it would not be blocked by the

navigation bar. Its font is set to Impact (if the user has it installed) and the font color is set to #597d37.

```
#clear-cart {
    background-color: #75bd85;
    width: 20%;
    font-size: 20px;
    text-align: center;
    color: white;
    text-decoration: none;
    padding: 10px;
    box-shadow: 5px 5px 5px #5a9b57;
}
```

This selector refers to the clear cart button. I added a property for its background color, width, font size, and font color. I removed the underline and added a box shadow.

```
.table-content {
    display: grid;
    grid-template-columns: 75% 20%;
    grid-template-rows: auto;
    gap: 5%;

}
```

This selector is used for the table and aside tag. The table would take up 75% of the screen, the aside tag takes up 20%, and the remaining is used for the gap in between. There is only one row so I set it to auto. An alternative approach would be using flex since there is only one row.

```
.shopping-cart-title {
    font-size: 50px;
    text-align: center;
}

#shopping-cart-table {
    background-color: #cfe3be;
    padding: 30px;
    box-shadow: 5px 5px 5px #5a9b57;
}

#shopping-cart-table .table-row {
    background-color: #bddfcf;
    text-align: center;
    font-size: 30px;
    color: #6ab767;
}

#shopping-cart-table th {
    background-color: #6ab767;
    font-size: 30px;
    padding: 5px;
}

.quantity-cell {
    display: flex;
    gap: 10px;
    justify-content: center;
}

.quantity-btn {
    background-color: #cfe3be;
    border-color: #6ab767;
    border-width: 3px;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    color: #597d37;
    padding: 10px;
    border-radius: 10%;
    cursor: pointer;
}
```

These selectors are used to make modifications to the table containing the products added to cart and the contents within. The selector .table-row modifies is a class selector that modifies each

row in the table. I added a background color property, aligned the text, and set the font size and color. For each table header, I included a background color, font size, and padding property. The quantity cell class has three elements so I used flex display to layout the elements. The quantity-btn class selector is used to modify the increase and decrease button. Most of the selectors are explained already besides “cursor: pointer” which changes the cursor design when the user hovers over the buttons.

```
#total-cost-content {
    background-color: #cfe3be;
    padding: 10px;
    border-radius: 10%;
    box-shadow: 5px 5px 5px #5a9b57;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-content: center;
}

#total-cost {
    font-size: 40px;
}

#checkout {
    background-color: #597d37;
    text-align: center;
    color: white;
    text-decoration: none;
    padding: 5px;
    border-radius: 10%;
    animation-name: flickeringbutton;
    animation-duration: 3s;
    animation-iteration-count: infinite;
}

@keyframes flickeringbutton {
    from {
        background-color: #6ab767;
    }

    to {
        background-color: #597d37;
    }
}
```

These selectors are used to modify the design for the summary container. Most of the properties used are used within other selectors as well. I included an animation on the checkout anchor tag so the background color flickers between two colors every three seconds.

checkout.css

```
#description {
    color: red;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    font-size: 20px;
}

.checkout-container {
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.checkout-content {
    height: 100%;
    width: 100%;
    background-color: #cfe3be;
    font-size: 30px;
    padding: 20px;

}

#checkout-title {
    font-size: 50px;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    color: #597d37;
    text-align: center;


}

.checkout-subTitles {
    font-size: 30px;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    color: #597d37;
}
```

These selectors are used to modify the containers used on the checkout page. I mainly added properties to change the color, font family, and font size.

```
#shipping-address-content {
    display: grid;
    grid-template-columns: auto auto;
    grid-template-rows: auto auto auto auto auto auto auto;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    font-size: 20px;
    color: #6ab767;
    gap: 10px;
    padding: 20px;
}

.shipping-address-info {
    width: 100%;
}

.shipping-address-input {
    color: #597d37;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    width: 50%;
    display: flex;
    flex-direction: row;
}
```

These selectors are used to alter the shipping address content. I used a grid display with 2 columns and 7 rows to organize the content within the shipping address. Additionally, I added a flex display on the input with a flex direction of row.

```
#payment-method-content {
    display: grid;
    grid-template-columns: auto auto;
    grid-template-rows: auto auto auto;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    font-size: 20px;
    color: #6ab767;
    gap: 10px;
    padding: 20px;
}

.payment-method-info {
    width: 100%;

}

.payment-method-input {
    color: #597d37;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    width: 50%;
    display: flex;
    flex-direction: row;
}
```

These selectors are used to alter the payment method content. The properties used are similar to the ones used for shipping address content. However, the grid display has two columns and three rows instead.

```
#updates-text {
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    font-size: 20px;
    color: #6ab767;
    gap: 10px;
    padding: 20px;
}
```

This is the selector for the update text. I changed the font family, size, and color of the text.

```css
#payment-method-description,
#shipping-address-description {
    color: red;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    font-size: 10px;
}

.invalid-input {
    border-color: red;
}

.valid-input {
    border-color: #6ab767;
}

.payment-type,
#updates-text {
    cursor: pointer;
}
```

These are the selectors to modify the border color when the input is invalid based on the conditions I set in its corresponding JavaScript file. When the input is invalid, the border will be set to red.

purchase-loading.css

```css
* {
    box-sizing: border-box;
}

body {
    background-color: #5a9b57;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 100vh;
    overflow: hidden;
    margin: 0;
}

.kinetic {
    position: relative;
    height: 80px;
    width: 80px;
}

.kinetic::after,
.kinetic::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 0;
    height: 0;
    border: 50px solid transparent;
    border-bottom-color: #fff;
    animation: rotateA 2s linear infinite 0.5s;
}

.kinetic::before {
    transform: rotate(90deg);
    animation: rotateB 2s linear infinite;
}

@keyframes rotateA {

    0%,
    25% {
        transform: rotate(0deg);
    }

    50%,
    75% {
        transform: rotate(180deg);
    }

    100% {
        transform: rotate(360deg);
    }
}

@keyframes rotateB {

    0%,
    25% {
        transform: rotate(90deg);
    }

    50%,
    75% {
        transform: rotate(270deg);
    }

    100% {
        transform: rotate(450deg);
    }
}

#loading-description-content {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-content: center;
    text-align: center;
    position: absolute;
    left: 100;
    top: 0;
}

.loading-description {
    font-size: 50px;
    color: white;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;

}
```

This is the CSS file used in Brad Traversy's Kinetic loader video. The most prominent part is the animations he used to change the top and bottom triangle. He transforms the triangle by making it rotate a certain degree at each interval, giving the triangle a loading effect.

thank-you.css

```css
body {
    background-color: #5a9b57;
}

#content {
    background-color: #cfe3be;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-content: center;
    padding: 30px;
    text-align: center;
    position: fixed;
    top: 30%;
    left: 30%;
}

#thank-you-text {
    color: #597d37;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    text-align: center;
}

#home-page-link {
    background-color: #597d37;
    text-align: center;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    font-size: 30px;
    color: white;
    text-decoration: none;
    padding: 5px;
    border-radius: 10%;
}
```

This is the CSS file for the thank-you.html file. I mainly changed the color, aligned the text, removed underlines for the link, and assigned a font family to the elements on this page.

about-us.css

The header and navigation styles used are the same as the ones used in the other CSS files.

```css
#counter-content {
    display: flex;
    flex-direction: row;
    justify-content: center;
    color: white;
}

.counter-container {
    display: flex;
    flex-direction: column;
    justify-content: center;
    text-align: center;
    margin: 30px 50px;
}

.counter {
    font-size: 60px;
    margin-top: 10px;
}

@media (max-width: 580px) {
    body {
        flex-direction: column;
    }
}

#author {
    position: relative;
    top: -100px;
    left: 20px;
    font-size: 20px;
}
```

This is the CSS code used in Brad Traversy incrementing counter to display a counter that increases up to the amount it was setted to. The @media selector used will only take into effect when the max-width is set to 580px.

terms-of-services.css

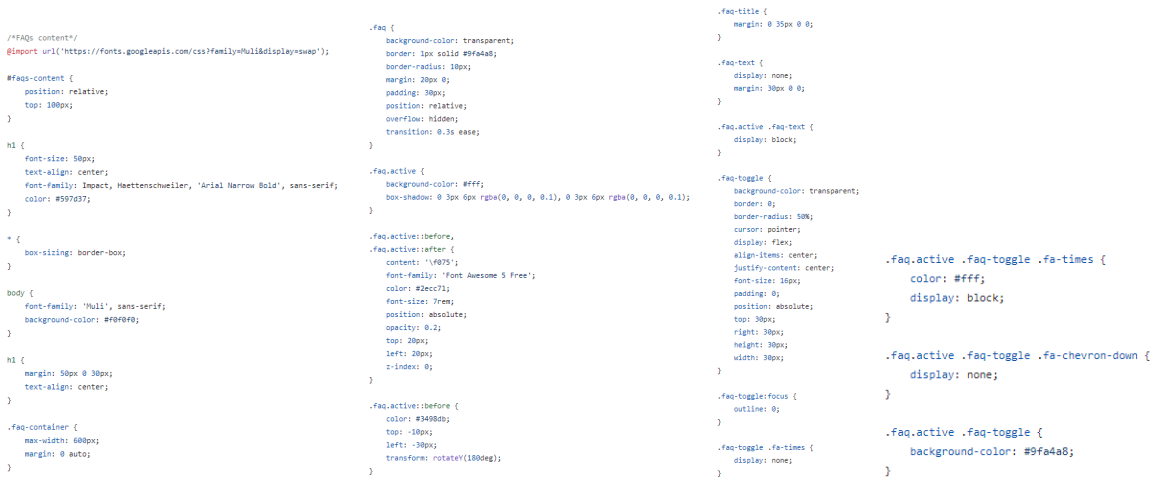
The header and navigation styles used are the same as the ones used in the other CSS files.

```css
/*Main content*/
article {
    position: relative;
    top: 100px;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    color: #597d37;
    background-color: #bddfcf;
    padding: 10px;
}

#terms-of-services-title {
    font-size: 50px;
    text-align: center;
}

article li {
    font-size: 20px;
    color: #6ab767;
}
```

I mainly included properties to style the font size and color for each listed item within the article.

faqs.css

The header and navigation styles used are the same as the ones used in the other CSS files.



```css
/*FAQs content*/
@import url('https://fonts.googleapis.com/css?family=Muli&display=swap');

#faqs-content {
    position: relative;
    top: 100px;
}

h1 {
    font-size: 50px;
    text-align: center;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
    color: #597d37;
}

* {
    box-sizing: border-box;
}

body {
    font-family: 'Muli', sans-serif;
    background-color: #f0f0f0;
}

h1 {
    margin: 50px 0 30px;
    text-align: center;
}

.faq-container {
    max-width: 600px;
    margin: 0 auto;
}

.faq {
    background-color: transparent;
    border: 1px solid #9fa4a8;
    border-radius: 10px;
    margin: 20px 0;
    padding: 30px;
    position: relative;
    overflow: hidden;
    transition: 0.3s ease;
}

.faq.active {
    background-color: #fff;
    box-shadow: 0 3px 6px rgba(0, 0, 0, 0.1), 0 3px 6px rgba(0, 0, 0, 0.1);
}

.faq.active::before,
.faq.active::after {
    content: '\f075';
    font-family: 'Font Awesome 5 Free';
    color: #2ecc71;
    font-size: 7rem;
    position: absolute;
    opacity: 0.2;
    top: 20px;
    left: 20px;
    z-index: 0;
}

.faq.active::before {
    color: #3498db;
    top: -10px;
    left: -30px;
    transform: rotateY(180deg);
}

.faq-title {
    margin: 0 35px 0 0;
}

.faq-text {
    display: none;
    margin: 30px 0 0;
}

.faq.active .faq-text {
    display: block;
}

.faq-toggle {
    background-color: transparent;
    border: 0;
    border-radius: 50%;
    cursor: pointer;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 16px;
    padding: 0;
    position: absolute;
    top: 30px;
    right: 30px;
    height: 30px;
    width: 30px;
}

.faq-toggle:focus {
    outline: 0;
}

.faq-toggle .fa-times {
    display: none;
}

.faq.active .faq-toggle .fa-times {
    color: #fff;
    display: block;
}

.faq.active .faq-toggle .fa-chevron-down {
    display: none;
}

.faq.active .faq-toggle {
    background-color: #9fa4a8;
}
```

This is the file used in Brad Traversy's frequently asked questions CSS file. It mainly utilizes pseudo-classes to modify a specific child. It also uses a toggle which adds and removes styles when clicked.

JavaScript
products.js

```
let cartItems = []; // Initialize an empty cartItems array
let storedItems;
let oldValue = [];
let quantity = 1;
let subtotalArray = [];
```

Creating an empty array for each variable.

```
let addToCartBtn = document.querySelectorAll(".add-to-cart-btn");
let totalCost = document.getElementById("total-cost");

// Loop through each "Add to Cart" button and add the click event listener
addToCartBtn.forEach(myProductBtn => {
    myProductBtn.addEventListener("click", handleAddToCartClick);
});
```

Select all elements with .add-to-cart-btn class and assign it to addToCartBtn. For each node within addToCartBtn, I added an addEventListener that listens to a click event. When the button is clicked, it will call the handleAddToCartClick function.

```
function handleAddToCartClick() {
    //for each node in productBtns, retrieve the name, price, and image

    //retrieve name
    const parentNode = this.parentNode;
    const itemName = parentNode.getAttribute("id");

    //retrieve price
    const priceElement = parentNode.lastChild.previousSibling.previousSibling.previousSibling;
    const price = parseFloat(priceElement.textContent);

    //retrieve image
    const imageElement = parentNode.firstChild.nextSibling;
    const image = imageElement.getAttribute("src");

    //subtotal
    let subtotal = price.toFixed(2) * quantity;

    let product = {
        name: itemName,
        price: price,
        image: image,
        quantity: quantity,
        subtotal: subtotal
    }

    // Add the product to the cartItems array
    cartItems.push(product);

    //check if there are old values that exist in local storage
    if (localStorage.getItem("myCartItems")) {
        oldValue = JSON.parse(localStorage.getItem("myCartItems"));

        let combinedCart = oldValue.concat(cartItems);
        let distinctCombinedCart = distinctArray(combinedCart);

        // combine two arrays of objects (old cart and new cart)
        localStorage.setItem("myCartItems", JSON.stringify(distinctCombinedCart));

    } else {
        //there is no value stored in local storage
        localStorage.setItem("myCartItems", JSON.stringify(cartItems));
    }
}
```

This is the handleAddToCartClick function. I retrieve the name, price, and image for each element node. Then, I pushed each item to the product array with the information I retrieved. Since the HTML is stateless, the buttons I clicked and added to cart would not be saved when the page is refreshed. As a result, I had to use localStorage to store the information. The localStorage stores the array called distinctCombinedCart, which will be explained later on. Stringify is used to remove the quotation marks around the product.

```
//comparing by the object's name. Since you can't compare objects, change it to string first then compare. Then change it
function distinctArray(array) {
    let distinctArray = [];
    let elementString;
    let elementObject;
    let distinctArray2 = [];
    for (let element of array) {
        elementString = JSON.stringify(element);
        //if  distinctArray doesn't include the element or is empty, then push it into the array
        if (!distinctArray.includes(elementString) || distinctArray.length === 0) {
            distinctArray.push(elementString);
        }
    }

    //after each iteration, all unqiue elements in distinctArray will be of type string.
    for (let i = 0; i < distinctArray.length; i++) {
        elementObject = JSON.parse(distinctArray[i]);
        distinctArray2.push[elementObject];
    }
    for (let element of distinctArray) {
        let element2 = JSON.parse(element);
        distinctArray2.push(element2);
    }
    return distinctArray2;
}
```

distinctArray function is used to remove any duplicate elements in the array. I used this function because when a product is added to the cart, the products that were added previously are added again. This is because I used concat in the previous step portion to include more items in the shopping cart. In this function, I only push the element when the array doesn't include the element.

```
//Making the table in shopping-cart.html
let shoppingCartTable = document.getElementById("shopping-cart-table");

if (document.URL.includes("pages/shopping-cart.html")) {
    for (let i = 0; i < storedItems.length; i++) {
        //Creating elements in table and appending it to shoppingCartTable
        let newRow = document.createElement("tr");
        let newDataImgCell = document.createElement("td");
        let newDataImg = document.createElement("img");
        let newDataName = document.createElement("td");
        let newDataPrice = document.createElement("td");
        let newDataQuantityCell = document.createElement("td");
        let newBtnDown = document.createElement("button");
        let newDataQuantity = document.createElement("div");
        let newBtnUp = document.createElement("button");
        let newDataSubtotal = document.createElement("td");

        //append the elements to quantity cell
        newDataQuantityCell.appendChild(newBtnDown);
        newDataQuantityCell.appendChild(newDataQuantity);
        newDataQuantityCell.appendChild(newBtnUp);

        //append the elements to the newRow
        newRow.appendChild(newDataImgCell);
        newRow.appendChild(newDataName);
        newRow.appendChild(newDataPrice);
        newRow.appendChild(newDataQuantityCell);
        newRow.appendChild(newDataSubtotal);

        //append each row element to the table
        shoppingCartTable.appendChild(newRow);

        //adding class attribute for each row
        newRow.setAttribute("class", "table-row")

        //adding class attribute for each subtotal
        newDataSubtotal.setAttribute("class", "table-subtotal")

        //Setting attributes of newDataImg element
        newDataImg.setAttribute("src", storedItems[i].image);
        newDataImg.setAttribute("class", "shopping-cart-images");
        newDataImgCell.appendChild(newDataImg);

        //Setting attributes for the quantity buttons
        newDataQuantityCell.setAttribute("class", "quantity-cell")
        newBtnDown.setAttribute("type", "button");
        newBtnUp.setAttribute("type", "button");
        newBtnDown.setAttribute("class", "decrease-product-quantity quantity-btn");
        newBtnUp.setAttribute("class", "increase-product-quantity quantity-btn");

        //Setting contents in quantity column
        newBtnDown.textContent = "down";
        newDataQuantity.textContent = storedItems[i].quantity;
        newBtnUp.textContent = "up";

        //Setting contents in for item name, price, and subtotal
        newDataName.textContent = storedItems[i].name.toUpperCase();
        newDataPrice.textContent = storedItems[i].price;
        newDataSubtotal.textContent = storedItems[i].subtotal;

        let myDataSubtotal;
        totalCost.textContent = "$" + total(storedItems);

        //setting hegiht and width manually but implemetn this in CSS instead
        newDataImg.setAttribute("height", "100px");
        newDataImg.setAttribute("width", "100px");
```

This is the code used if the current page is the shopping cart page. I placed the code within the product.js file as opposed to creating a separate file because I need to access the product array. For each item that is stored in localStorage, I created an element for row, image, name, price, quantity buttons, and subtotal. Subsequently, I appended the elements to its proper location on the DOM and set attributes for some elements. textContent is used to edit the content that is being shown for a specific element.

```
//change quantity with click event listener
newBtnDown.addEventListener("click", function () {
    handleBtnDownClick(newDataQuantity);
    myDataSubtotal = updateSubtotal(newDataPrice, newDataQuantity);
    newDataSubtotal.textContent = myDataSubtotal;
});
newBtnUp.addEventListener("click", function () {
    handleBtnUpClick(newDataQuantity);
    myDataSubtotal = updateSubtotal(newDataPrice, newDataQuantity);
    newDataSubtotal.textContent = myDataSubtotal;
});
```

Add an event listener for each up and down quantity button. Each time the button is clicked, it will call either the handleBtnDownClick() function or the handleBtnUpClick() function depending on which quantity button was clicked and set it to the variable myDataSubtotal. Next, the text content of newDataSubtotal will display the new value.

```
//each time you click on clear cart, the table will refresh
let clearCart = document.getElementById("clear-cart");
clearCart.addEventListener("click", () => {
    localStorage.clear();
});
```

Clears the localStorage by removing all elements within it.

```
//creating an array of subtotal
let subtotal = document.querySelectorAll(".table-subtotal");

subtotal.forEach(eachSubtotal => {
    subtotalArray.push(eachSubtotal.textContent);
});
```

This is used to retrieve the text of each subtotal within the table.

```
    //Update total cost when quantity buttons are clicked
    let rows = document.querySelectorAll(".table-subtotal");
    let newTotal = 0;
    let quantityBtn = document.querySelectorAll(".quantity-btn");

    quantityBtn.forEach(btn => {
        btn.addEventListener("click", () => {
            rows.forEach(row => {
                newTotal += parseFloat(row.textContent);
            });
            totalCost.textContent = "$" + newTotal.toFixed(2);
            //resets the value
            newTotal = 0;
        });
    });
}
```

For each quantity button that was clicked, it will update the total price displayed in the summary section. toFixed() method is used to set the float value to a limit of 2.

```
//Finds the total cost of items added to cart already.
function total(storedItems) {
    let sum = 0;
    for (let i = 0; i < storedItems.length; i++) {
        sum += storedItems[i].subtotal;
    }
    sum.toFixed(2);
    return sum;
}
```

Find the total cost of items that are added to cart already with a for loop and add all value together and store it in sum.

```
//Decrease quantity count
function handleBtnDownClick(element) {
    let quantity = parseInt(element.textContent);
    if (quantity <= 1) {
        //do nothing
    } else {
        quantity--;
        element.textContent = quantity;
    }
}

//Increase quantity count
function handleBtnUpClick(element) {
    let quantity = parseInt(element.textContent);
    quantity++;
    element.textContent = quantity;
}
```

These two functions are used to handle the quantity button when you increase or decrease the quantity count. First, I parse the string to an int and increment or decrement it based on which function it is used in. Next, I will set the quantity to the element.

```
//Calculate subtotal by multiplying price and quantity
function updateSubtotal(price, quantity) {
    //first change price to float with parseFloat() and quantity to int with parseInt(), then price * quantity
    let myPrice = parseFloat(price.textContent);
    let myQuantity = parseInt(quantity.textContent);
    let mySubtotal;

    if (!isNaN(myPrice) && !isNaN(myQuantity)) {
        mySubtotal = myPrice * myQuantity;
        return mySubtotal.toFixed(2);
    } else {
        return -1;
    }
}
```

This function is used to calculate the subtotal for each product by multiplying the price and quantity. First, I convert the string to float or int. Then, I check whether the quantity and price is a number. If it is, I multiplied them together and returns the subtotal.

```
//Button ripple effect (udemy day 20)
let buttons = document.querySelectorAll('.ripple')

buttons.forEach(button => {
    button.addEventListener('click', function (e) {
        const x = e.pageX
        const y = e.pageY

        const buttonTop = e.target.offsetTop
        const buttonLeft = e.target.offsetLeft

        const xInside = x - buttonLeft
        const yInside = y - buttonTop

        const circle = document.createElement('span')
        circle.classList.add('circle')
        circle.style.top = yInside + 'px'
        circle.style.left = xInside + 'px'

        this.appendChild(circle)

        setTimeout(() => circle.remove(), 500)
    })
})
```

This is the function used in Brad Traversy's button ripple effect video. When a button is clicked, it creates a circle that will spread and eventually reach zero opacity.

checkout.js

```
//shipping address variables
let firstNameInShippingAddress = document.getElementById("firstNameShippingAddress");
let lastNameInShippingAddress = document.getElementById("firstNameShippingAddress");
let email = document.getElementById("email");
let stressAddress = document.getElementById("streetAddress");
let townCity = document.getElementById("town-city");
let postcode = document.getElementById("postcode");
let country = document.getElementById("country");
let stateCounty = document.getElementById("state-county");

//payment method variables
let firstNamePaymentMethod = document.getElementById("firstNamePaymentMethod");
let lastNamePaymentMethod = document.getElementById("lastNamePaymentMethod");
let creditCardNumber = document.getElementById("creditCardNumber");
let securityCode = document.getElementById("securityCode");
let cardExpiration = document.getElementById("cardExpiration");

let purchaseBtn = document.getElementById("purchaseBtn");
```

These lines are used to assign the variables to the corresponding element in the DOM.

```
purchaseBtn.addEventListener("click", function (event) {
    if (validateShippingAddress() === false || validatePaymentMethod() === false) {
        event.preventDefault();
    }
});
```

When the shipping address or payment method is false, it will prevent the user from proceeding to the next page when clicking on the "Proceed to Checkout" button.

```
function validateShippingAddress() {
    if (firstNameInShippingAddress.value === '') {
        //checks if shipping address first name is empty
        firstNameInShippingAddress.classList.add("invalid-input");
        return false;
    } else if (lastNameInShippingAddress.value === '') {
        //checks if shipping address last name is empty
        lastNameInShippingAddress.classList.add("invalid-input");
        return false;
    } else if (!email.value.includes("@")) {
        //checks if email has an @
        email.classList.add("invalid-input");
        return false;
    } else if (stressAddress.value === '') {
        //checks if shipping address street address is empty
        stressAddress.classList.add("invalid-input");
        return false;
    } else if (townCity.value === '') {
        //checks if shipping address town/city is empty
        townCity.classList.add("invalid-input");
        return false;
    } else if (postcode.value.length !== 5 || isNaN(postcode.value)) {
        //checks if postcode's length is 5 and comprised of numbers only
        postcode.classList.add("invalid-input");
        return false;
    } else if (country.value === '') {
        //checks if shipping address country is empty
        country.classList.add("invalid-input");
        return false;
    } else if (stateCounty.value === '') {
        //checks if shipping address country is empty
        stateCounty.classList.add("invalid-input");
        return false;
    }
    return true;
}
```

This is the function used to validate each input within the shipping address container. Most of these check whether the input is empty. The email input checks whether the user inputted a “@” character in their email. Postcode checks whether the input is a number and if the length is 5.

```
function validatePaymentMethod() {
    if (firstNamePaymentMethod.value === '') {
        //checks if payment method first name is empty
        firstNamePaymentMethod.classList.add("invalid-input");
        return false;
    } else if (lastNamePaymentMethod.value === '') {
        //checks if payment method last name is empty
        lastNamePaymentMethod.classList.add("invalid-input");
        return false;
    } else if (creditCardNumber.value.length !== 16 || isNaN(creditCardNumber.value)) {
        //checks if credit card number is length is 16 and comprised of numbers only
        creditCardNumber.classList.add("invalid-input");
        return false;
    } else if (securityCode.value.length !== 3 || isNaN(securityCode.value)) {
        //checks if security code is length is 3 and comprised of numbers only
        securityCode.classList.add("invalid-input");
        return false;
    } else if (cardExpiration.value.length == '') {
        //checks if card expiration is empty
        cardExpiration.classList.add("invalid-input");
        return false;
    }
    return true;
}
```

This is the function used to validate each input within the payment method container. Again, most of the if-else statement checks whether the input is empty. Credit card number input checks whether the length is 16 and is a number. Security code checks whether the length is 3 and is a valid number.

home-page.js

```
const panels = document.querySelectorAll('.panel')

panels.forEach(panel => {
    panel.addEventListener('click', () => {
        removeActiveClasses()
        panel.classList.add('active')
    })
})

function removeActiveClasses() {
    panels.forEach(panel => {
        panel.classList.remove('active')
    })
}
```

This is the JavaScript code used in Brad Traversy's Expanding Card video. When a panel is clicked, it will add or remove the class active from the panel.

about-us.js

```
const counters = document.querySelectorAll('.counter')

counters.forEach(counter => {
    counter.innerText = '0'

    const updateCounter = () => {
        const target = +counter.getAttribute('data-target')
        const c = +counter.innerText

        const increment = target / 200

        if (c < target) {
            counter.innerText = `${Math.ceil(c + increment)}`
            setTimeout(updateCounter, 1)
        } else {
            counter.innerText = target
        }
    }

    updateCounter()
})
```

This is the JavaScript code used in Brad Traversy's Incrementing Counter video. It will increase the count to a specific value.

faqs.js

```
const toggles = document.querySelectorAll('.faq-toggle')

toggles.forEach(toggle => {
    toggle.addEventListener('click', () => {
        toggle.parentNode.classList.toggle('active')
    })
})
```

This is the JavaScript code used in Brad Traversy's frequently asked questions video. It has a toggle property and for each element with the class .faq-toggle, it will add or remove an active class to the element.

thank-you.js

```
//returns to home page and clears all item in shopping cart
let clearCart = document.getElementById("home-page-link");
clearCart.addEventListener("click", () => {
    localStorage.clear();
});
```

I included a variable that is linked to the “continue shopping” button. When the button is clicked, it will remove all the products in the localStorage, hence, clearing off all items listed in the shopping cart.