

GROUP PROJECT

INDEX.HTML :-

```
194     <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
195     <script>
196         $(document).ready(function() {
197             // Hover effect for Log In and Sign Up links
198             $("#login-link, #signup-link").hover(function() {
199                 $(this).css("color", "red");
200             }, function() {
201                 $(this).css("color", "white");
202             });
203         });
204     </script>
205 </body>
206 </html>
```

With the help of the JavaScript library jQuery, the following code snippet creates a hover effect for the "Log In" and "Sign Up" links on a webpage. The script begins to run once the webpage has finished loading completely. It uses the links' respective IDs to apply the hover() method to both of them. The text colour is changed to red when the mouse enters either link thanks to a JavaScript code that is called. The text colour is then returned to white when the mouse exits the link as a

result of another function that is configured to run. Users can see that these links are interactive when they hover over or move away from them thanks to this dynamic change in font colour.

In short, the script makes use of jQuery to change how the "Log In" and "Sign Up" links appear. By giving responsive visual feedback to link engagement, it makes it possible for the text colour to shift to red when hovering and return to white upon release.

MEN.HTML, WOMEN.HTML, KIDS.HTML, PERFUME.HTML :-

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>

function setupSlideShow(slider) {
    let currentSlide = 0;
    const slides = slider.find('.slide');

    function showSlide(index) {
        slides.removeClass('active');
        slides.eq(index).addClass('active');
    }

    slider.append('<div class="slide-indicators"></div>');
    const indicators = slider.find('.slide-indicators');

    slides.each(function(index) {
        indicators.append('<div class="indicator" $(index) == 0 ? "active" : ""> data-slide-index="' + index + '></div>');
    });

    indicators.on('click', '.indicator', function() {
        currentSlide = parseInt($(this).attr('data-slide-index'));
        showSlide(currentSlide);
        updateIndicators();
    });

    function updateIndicators() {
        indicators.find('.indicator').removeClass('active');
        indicators.find('[data-slide-index="' + currentSlide + '"]').addClass('active');
    }

    showSlide(currentSlide);
}

// Call the setupSlideShow function for each product's slider
$('.product-slider').each(function() {
    setupSlideShow(this);
});

$(document).ready(function() {
    const cart = [];

    function updateCart() {
        $('.cart-items').empty();
        let total = 0;

        cart.forEach((item, index) => {
            $('.cart-items').append('<li>$(item.product) - $(item.size) - $(item.price) <button class="remove-from-cart-button" data-index="' + index + '>Remove</button></li>');
            total += item.price;
        });

        $('.total-price').text('Total Price: $' + total.toFixed(2));
        $('.cart-count').text(cart.length);
    }

    $('.add-to-cart-button').click(function() {
        const product = $(this).siblings('h3').text();
        const selectedSize = $(this).siblings('size-options').find('size-select').val();
        const price = parseFloat($(this).siblings('price').text().replace('$', ''));

        if (selectedSize) {
            cart.push({ product, size: selectedSize, price });
            updateCart();
            alertWithCancel('Added ' + product + ' (Size: ' + selectedSize + ') to cart.', function() {
                // Handle cancel option
                cart.pop(); // Remove the last added item from cart
                updateCart();
            });
        } else {
            alert('Please select a size before adding to cart.');
```

This script is designed to enhance the functionality of a webpage, likely for an online shopping platform. It utilizes jQuery to create two main features: a slide show for product images and a shopping cart functionality.

The first part of the script defines a function named **setupSlideShow**, which takes a slider element as an argument. This function sets up a slide show within the provided slider element. It cycles through the slides within the slider and displays them one by one, highlighting the active slide. Additionally, it adds slide indicators below the slides to allow users to navigate between them. The script associates a click event with these indicators, allowing users to switch between slides easily. This functionality makes the presentation of product images more interactive and engaging for users.

The second part of the script focuses on managing the shopping cart. It initializes an empty array named **cart** to store items that users add to their cart. Whenever a user clicks the "Add to Cart" button on a product, the script extracts relevant information like product name, selected size, and price. It then adds this information to the **cart** array and updates the display of the cart contents. Users are also given the option to cancel the addition through a custom alert box. The script enables removal of items from the cart by clicking on the "Remove" button associated with each item. Furthermore, it allows users to filter products based on categories, hiding and showing the products accordingly when a category filter is selected.

Overall, this script improves the user experience of an online shopping website by incorporating a dynamic image slide show and an interactive shopping cart system. These features enhance user engagement and simplify the shopping process, making the website more user-friendly and effective in facilitating purchases.

LOGIN.HTML:-

```
116 <script>
117
118 // script.js
119 document.querySelector('form').addEventListener('submit', function(event) {
120     event.preventDefault();
121     const username = event.target.querySelector('input[type="text"]').value;
122     const password = event.target.querySelector('input[type="password"]').value;
123
124
125     alert(`Logged in as ${username}`);
126 });
127
128 </script>
129 </html>
130
131
132
```

This script handles the process when a user tries to log in using a form on a website. When the user submits the form (by pressing a button), the script takes the username and password entered in the form's text fields. It doesn't perform the actual checking of these credentials; it's like a placeholder for that part. Instead, it just shows a message saying "Logged in as [username]" to simulate a successful login. The script reminds developers to put the real login checks in place, like sending the data to a server for verification, so users can actually log in securely.

SIGNUP.HTML:-

```
76 <script>
77   $(document).ready(function() {
78     $('#signup-form').submit(function(e) {
79       e.preventDefault();
80       const username = $('#username').val();
81       const email = $('#email').val();
82       const password = $('#password').val();
83       const confirmPassword = $('#confirm-password').val();
84
85       if (password !== confirmPassword) {
86         alert('Passwords do not match. Please try again.');
```

When a user submits the sign-up form on the website, this script takes over. It first checks if the password entered matches the confirmed password. If they don't match, it displays an alert saying, "The passwords you entered are different. Please check and try again."

If the passwords match, the script displays a confirmation box asking if the user wants to proceed. If the user clicks "OK" in the confirmation box, it means they want to continue. In that case, the script redirects them to the website's home page (index.html).

In short, this script helps users sign up by making sure their passwords match. If they do, it asks for confirmation to proceed, and if confirmed, it takes the user to the home page of the website.

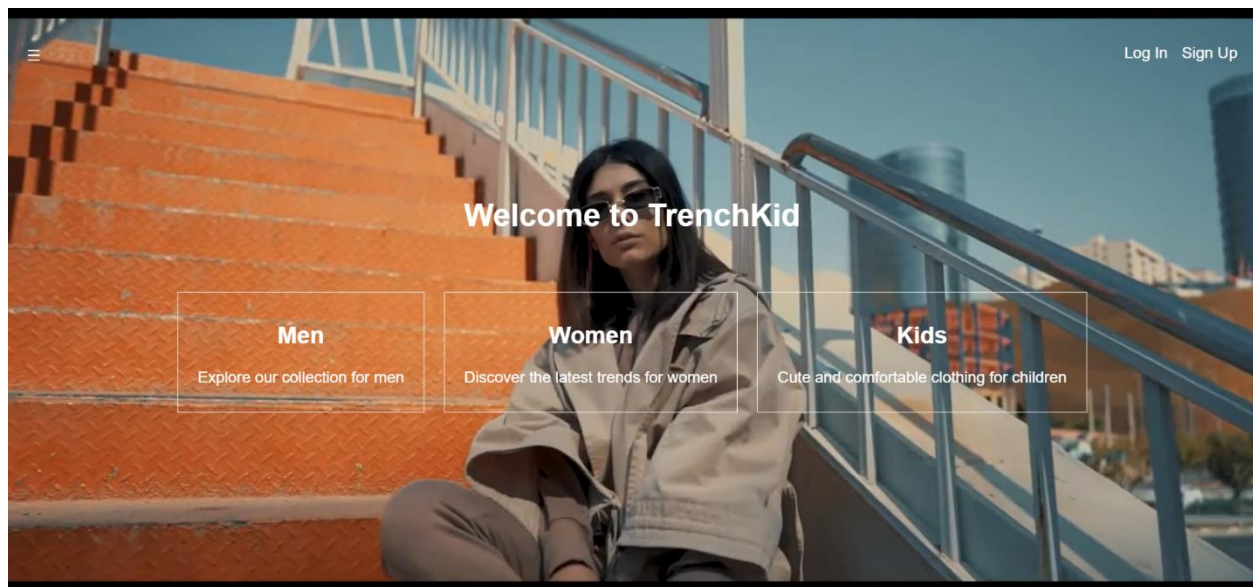
ABOUT US: -

```
198 <script>
199   const stars = document.querySelectorAll('.star');
200   const reviewComment = document.getElementById('review-comment');
201   const submitReviewBtn = document.getElementById('submit-review');
202   const reviewMessage = document.getElementById('review-message');
203
204   let selectedStars = 0;
205
206   stars.forEach(star => {
207     star.addEventListener('click', () => {
208       selectedStars = parseInt(star.getAttribute('data-star'));
209       updateStars(selectedStars);
210     });
211   });
212
213   submitReviewBtn.addEventListener('click', () => {
214     if (selectedStars > 0 && reviewComment.value.trim() !== '') {
215       reviewMessage.classList.remove('hidden');
216       reviewMessage.textContent = 'Review posted!';
217
218       clearReviewForm();
219     }
220   });
221
222   function updateStars(numStars) {
223     stars.forEach((star, index) => {
224       star.classList.remove('selected');
225       if (index < numStars) {
226         star.classList.add('selected');
227       }
228     });
229   }
230
231   function clearReviewForm() {
232     selectedStars = 0;
233     updateStars(selectedStars);
234     reviewComment.value = '';
235   }
236 </script>
```

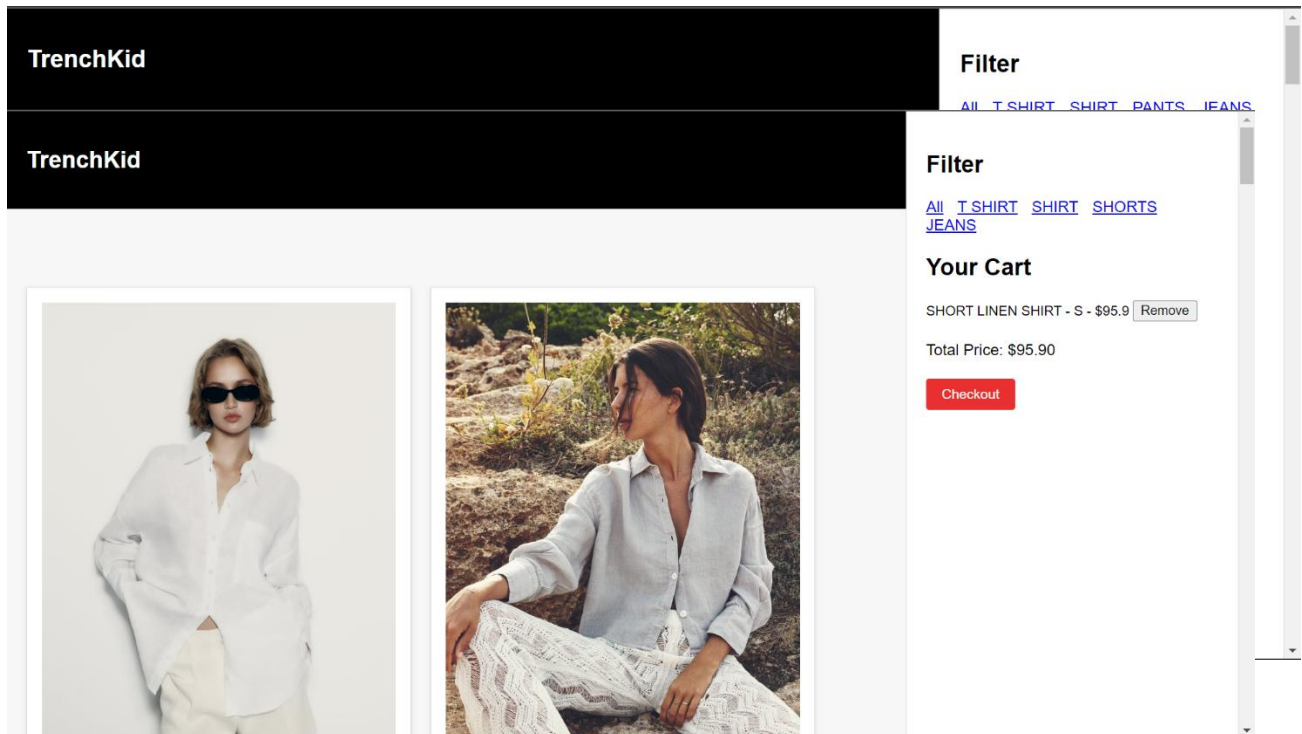
This JavaScript code snippet performs vital functions for a customer review section on a webpage. It starts by identifying specific HTML elements using class and ID identifiers. A variable called `selectedStars` is initialized to keep track of the number of stars a user selects. Event listeners are then attached to each star element to respond to user clicks. When a star is clicked, the `selectedStars` variable is updated, and the `updateStars` function is triggered. Additionally, a listener is placed on the "Submit Review" button. When clicked, it verifies if the user has

assigned a star rating and provided a comment. If both conditions are met, a message proclaiming "Review posted!" is displayed. The `clearReviewForm` function is called to reset the form by resetting the star count, adjusting star highlights, and clearing the review comment field. This cohesive system empowers users to submit reviews with ratings and comments, while also offering instant feedback and interaction through star selection.

HOME PAGE OUTPUT: -



MEN PAGE OUTPUT: -



WOMEN PAGE OUTPUT: -

KIDS PAGE OUTPUT: -

TrenchKid



Filter

[All](#) [BOYS](#) [GIRLS](#)

Your Cart

TEXT T-SHIRT - S - \$75.9 [Remove](#)

Total Price: \$75.90

[Checkout](#)

PERFUME PAGE OUTPUT: -

TrenchKid



Filter

[All](#) [MEN](#) [WOMEN](#)

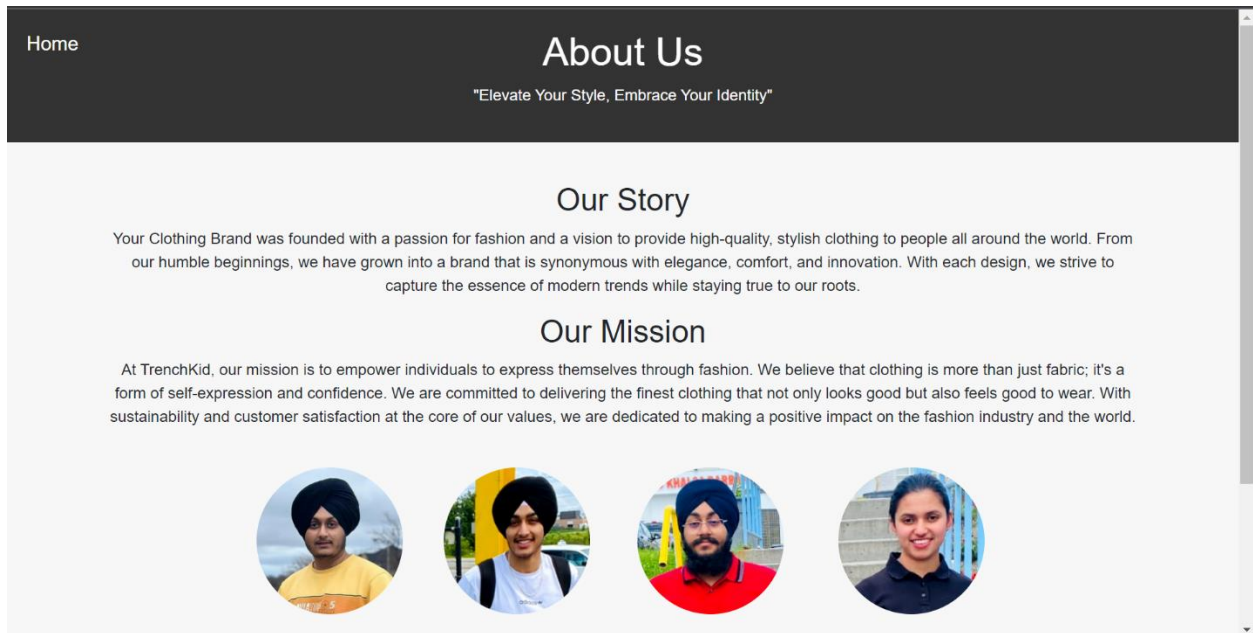
Your Cart

NIGHT POUR HOMME II SPORT 100 ML -
\$275.9 [Remove](#)

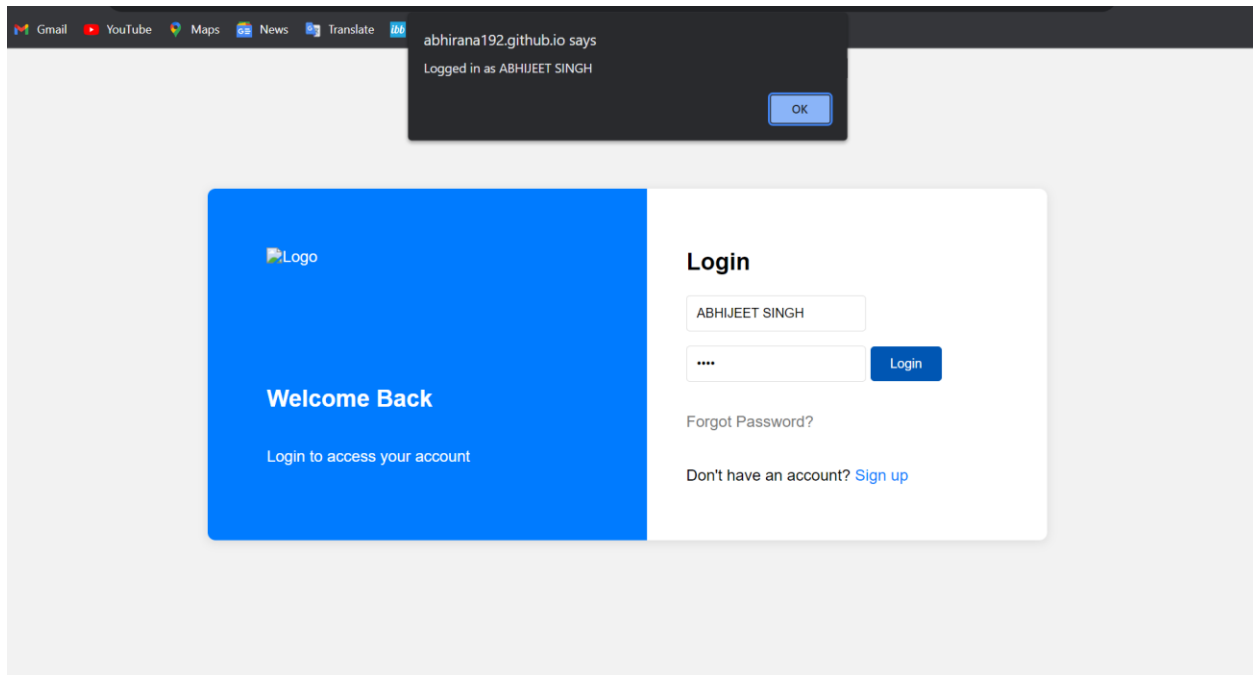
Total Price: \$275.90

[Checkout](#)

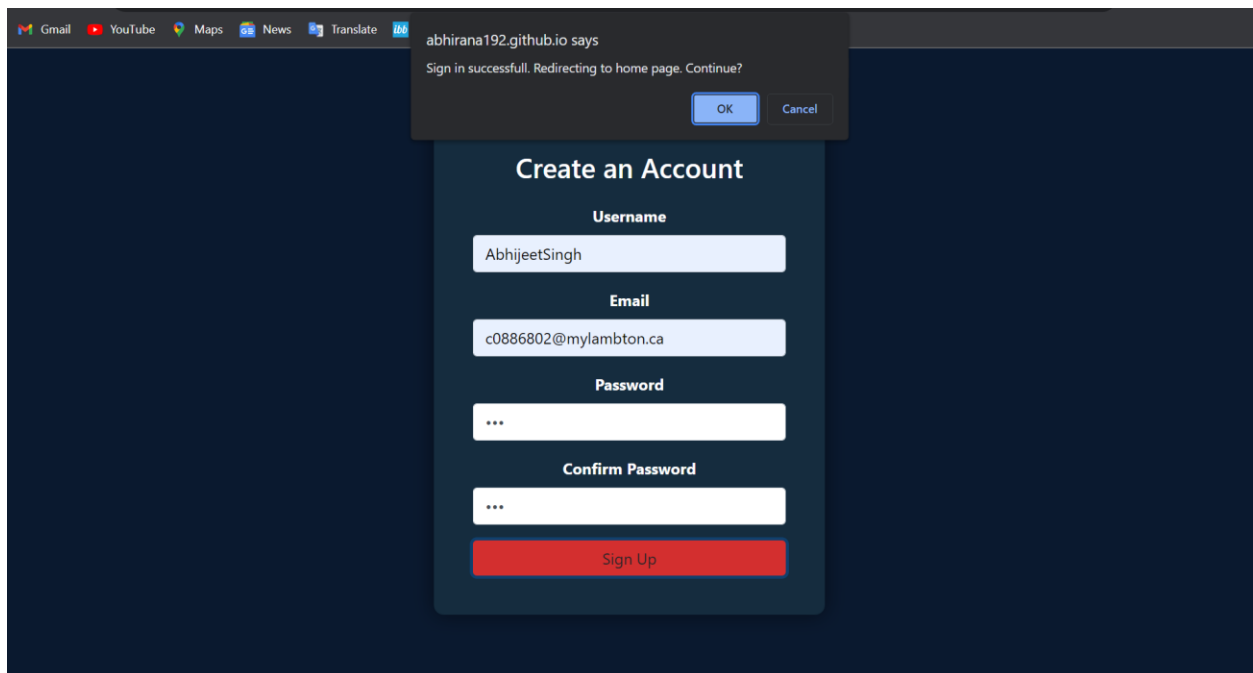
ABOUT US PAGE OUTPUT: -



LOG IN PAGE OUTPUT: -



SIGN UP PAGE OUTPUT: -



The screenshot displays a web browser window with a dark blue background. At the top, a notification bar from 'abhirana192.github.io' states 'Sign in successfull. Redirecting to home page. Continue?' with 'OK' and 'Cancel' buttons. Below this, a 'Create an Account' form is centered. The form includes input fields for 'Username' (containing 'AbhijeetSingh'), 'Email' (containing 'c0886802@mylambton.ca'), 'Password' (masked with '***'), and 'Confirm Password' (masked with '***'). A red 'Sign Up' button is positioned at the bottom of the form.

abhirana192.github.io says
Sign in successfull. Redirecting to home page. Continue?
OK Cancel

Create an Account

Username
AbhijeetSingh

Email
c0886802@mylambton.ca

Password

Confirm Password

Sign Up