base.html template <script>:

- handles Bootstrap tooltips and toasts, MailChimp subscriptions

```
1   document.addEventListener('DOMContentLoaded', function () {
2               var tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-
3               var tooltipList = tooltipTriggerList.map(function (tooltipTriggerEl) {
4                   return new bootstrap.Tooltip(tooltipTriggerEl);
5               });
6
7               var toastElList = [].slice.call(document.querySelectorAll('.toast'));
8               var toastList = toastElList.map(function (toastEl) {
9                   return new bootstrap.Toast(toastEl);
10              });
11              toastList.forEach(toast => toast.show());
12          });
13
14  (function ($) {
15              window.fnames = new Array();
16              window.ftypes = new Array();
17              fnames[0] = 'EMAIL';
18              ftypes[0] = 'email';
19              fnames[1] = 'FNAME';
20              ftypes[1] = 'text';
21              fnames[2] = 'LNAME';
22              ftypes[2] = 'text';
23              fnames[3] = 'ADDRESS';
24              ftypes[3] = 'address';
25              fnames[4] = 'PHONE';
26              ftypes[4] = 'phone';
27              fnames[5] = 'BIRTHDAY';
28              ftypes[5] = 'birthday';
29              fnames[6] = 'COMPANY';
30              ftypes[6] = 'text';
31          }(jQuery));
32          var $mcj = jQuery.noConflict(true);
```

CONFIGURE

Metrics

There are 5 functions in this file.

Function with the largest signature take 1 arguments, while the median is 1.

Largest function has 16 statements in it, while the median is 1.

The most complex function has a cyclomatic complexity value of 1 while the median is 1.

Two warnings

15 The array literal notation [] is preferable.

16 The array literal notation [] is preferable.

Three undefined variables

4 bootstrap
9 bootstrap
17 fnames
19 fnames
21 fnames
23 fnames
25 fnames
27 fnames
29 fnames
18 ftypes
20 ftypes
22 ftypes
24 ftypes
26 ftypes
28 ftypes
30 ftypes

Two unused variables

3 tooltipList
32 $mcj

Notes. All warnings and ‘undefined variables’ are related to the 3d party vendors: Bootstrap and MailChimp. ‘Undefined variables’ are previously defined in the external JS files

index.html template <script>:

- handles Swiper function on the landing page. The vendor’s script

```
1   document.addEventListener('DOMContentLoaded', function () {
2           const swiperEl = document.querySelector('swiper-container');
3
4           swiperEl.addEventListener('swiperslidechange', function (event) {
5               const swiper = event.target.swiper;
6               const activeIndex = swiper.realIndex;
7               document.querySelectorAll('.caption').forEach((caption, index) => {
8                   caption.style.display = (index === activeIndex) ? 'block' : 'none';
9               });
10          });
11
12          // Initially show the caption for the first slide
13          document.querySelector('.caption[data-slide="0"]').style.display = 'block';
14      });
```

CONFIGURE

Metrics

There are 3 functions in this file.

Function with the largest signature take 2 arguments, while the median is 1.

Largest function has 3 statements in it, while the median is 3.

The most complex function has a cyclomatic complexity value of 2 while the median is 1.

products/js/top_down.js :

- handles ‘top’ and ‘down’ buttons for fast scrolling

```javascript
$(document).ready(function () {
    // Show/hide buttons based on scroll position
    $(window).on('scroll', function () {
        var scrollTop = $(this).scrollTop();
        var docHeight = $(document).height();
        var windowHeight = $(this).height();

        // Show "Top" button if scrolled down
        if (scrollTop > 100) {
            $('.btn-scroll[href="#top"]').addClass('show');
        } else {
            $('.btn-scroll[href="#top"]').removeClass('show');
        }

        // Show "Bottom" button if not at the bottom
        if (scrollTop + windowHeight < docHeight - 100) {
            $('.btn-scroll[href="#bottom"]').addClass('show');
        } else {
            $('.btn-scroll[href="#bottom"]').removeClass('show');
        }

        // Check if near the bottom of the page
        if (scrollTop + windowHeight >= docHeight - 100) {
            $('.btn-scroll').addClass('white-text');
        } else {
            $('.btn-scroll').removeClass('white-text');
        }

    });

    // Trigger scroll event on page load to set initial button visibility
    $(window).trigger('scroll');
});

```

CONFIGURE

Metrics

There are 2 functions in this file.

Function with the largest signature take 0 arguments, while the median is 0.

Largest function has 12 statements in it, while the median is 7.

The most complex function has a cyclomatic complexity value of 4 while the median is 2.5.

products/includes/add_to_cart_script.html :

- handles adding product with selected quantity to the shopping cart

```javascript
$(document).ready(function () {
        // Listen for clicks on the decrease button
        $('#decrease_qtty').on('click', function () {
            let quantityInput = $('#display_qtty');
            let currentQuantity = parseInt(quantityInput.val());
            if (currentQuantity > 1) {
                quantityInput.val(currentQuantity - 1);
            }
        });

        // Listen for clicks on the increase button
        $('#increase_qtty').on('click', function () {
            let quantityInput = $('#display_qtty');
            let currentQuantity = parseInt(quantityInput.val());
            if (currentQuantity < 29) {
                quantityInput.val(currentQuantity + 1);
            }
        });

        // Listen for clicks on the 'Add to Cart' button
        $('#add_to_cart_submit').on('click', function () {
            let productId = $(this).val();

            let quantity = $('#display_qtty').val();

            $.ajax({
                type: 'POST',
                url: '/cart/add/',
                data: {
                    'product_id': productId,
                    'quantity': quantity,
                    'csrfmiddlewaretoken': '{{ csrf_token }}'
                },
                success: function (json) {
                    // Update the cart details in the navbar and offcanvas
                    window.location.reload();
                },
                error: function (xhr, errmsg, err) {
                    const errorMessage = 'Failed to add to the cart. Try again later
                    sessionStorage.setItem('toastMessage', errorMessage);
                    sessionStorage.setItem('toastType', 'danger');
                    alert('Failed to add product to cart.');

                    window.location.reload();
                }
            });
        });
    });
```

CONFIGURE

Metrics

There are 6 functions in this file.

Function with the largest signature take 3 arguments, while the median is 0.

Largest function has 5 statements in it, while the median is 3.5.

The most complex function has a cyclomatic complexity value of 2 while the median is 1.

products/js/sorting_select.js :

- Replace the URL with the current URL adding the sorting parameter

```
$(document).ready(function () {
    $('.sorting-select').change(function () {
        var selector = $(this);
        var currentUrl = new URL(window.location);

        var selectedVal = selector.val();
        if (selectedVal != "reset") {
            currentUrl.searchParams.set("sort", selectedVal);
        } else {
            currentUrl.searchParams.delete("sort");
        }
        window.location.replace(currentUrl);
    });
});
```

CONFIGURE

Metrics

There are 2 functions in this file.

Function with the largest signature take 0 arguments, while the median is 0.

Largest function has 7 statements in it, while the median is 4.

The most complex function has a cyclomatic complexity value of 2 while the median is 1.5.

cart/includes/change_quantity_and_progress_bar.html :

- Updates the width of progress bar to indicate how much left to free delivery
- Updates quantity of product and sends AJAX request to update the cart

```
$(document).ready(function () {

    // Update progress bar for free shipping threshold
    function updateProgressBar(newPercentage) {
        const progressBar = document.querySelector('.progress-bar');
        if (newPercentage < 100) {
            progressBar.style.width = newPercentage + '%';
        }
    }

    updateProgressBar('{{ percent_of_threshold }}');


    // Update quantity when user changes input

    // Decrement quantity
    $('.cart-btn-decrement').on('click', function () {
        var $input = $(this).siblings('.cart-qty-input');
        var currentQuantity = parseInt($input.val());
        var productId = $(this).closest('.count-input').data('product-id');

        if (currentQuantity > 1) {
            $input.val(currentQuantity - 1);
            updateCart(productId, $input.val());
        }
    });

    // Increment quantity
    $('.cart-btn-increment').on('click', function () {
        var $input = $(this).siblings('.cart-qty-input');
        var currentQuantity = parseInt($input.val());
        var productId = $(this).closest('.count-input').data('product-id');

        $input.val(currentQuantity + 1);
        updateCart(productId, $input.val());
    });

    // Function to send AJAX request
    function updateCart(productId, newQuantity) {
        $.ajax({
            url: '/cart/update/',
            type: 'POST',
            data: {
                'csrfmiddlewaretoken': '{{ csrf_token }}', // Ensure CSRF token
                'product_id': productId,
                'quantity': newQuantity
            },

            success: function (response) {
                // Handle success (e.g., update cart total, display message)
                window.location.reload();
            },
            error: function (xhr, status, error) {
                // Handle error
                console.error('Error updating cart:', error);
            }
        });
    }
});
```

CONFIGURE

Metrics

There are 7 functions in this file.

Function with the largest signature take 3 arguments, while the median is 1.

Largest function has 6 statements in it, while the median is 3.

The most complex function has a cyclomatic complexity value of 2 while the median is 1.

checkout/js/stripe_elements.js :

- Mounts Stripe payment element with Card and PayPal payment methods
- Sends data to server to modify PaymentIntent with Metadata
- Submits the form to Stripe with billing and shipping details

```
$(document).ready(function () {
    const stripe_public_key = $('#id_stripe_public_key').text().slice(1, -1);
    const clientSecret = $('#id_client_secret').text().slice(1, -1);

    const stripe = Stripe(stripe_public_key);

    const appearance = {
        theme: 'flat',
        variables: { colorPrimaryText: '#262626' }
    };
    const options = {
        layout: {
            type: 'tabs',
            defaultCollapsed: false,
        }
    };
    const elements = stripe.elements({ clientSecret, appearance });
    const paymentElement = elements.create('payment', options);
    paymentElement.mount('#payment-element');

    // Handle realtime validation errors on the payment element
    paymentElement.addEventListener('change', function (event) {
        var errorDiv = document.getElementById('card-errors');
        if (event.error) {
            var html = `
            <span class="icon" role="alert">
                <i class="fa-regular fa-triangle-exclamation"></i>
            </span>
            <span>${event.error.message}</span>
        `;
            $(errorDiv).html(html);
        } else {
            errorDiv.textContent = '';
        }
    });

    // Handle form submit
    var form = document.getElementById('payment-form');

    var paymentID = clientSecret.split('_secret')[0];

    form.addEventListener('submit', function (ev) {
        ev.preventDefault();
        paymentElement.update({ 'disabled': true });
        $('.complete-order').attr('disabled', true);
        $('#payment-form').fadeToggle(100);
        $('#loading-overlay').fadeToggle(100);

        var csrfToken = $('input[name="csrfmiddlewaretoken"]').val();
        var postData = {
            'csrfmiddlewaretoken': csrfToken,
            'client_secret': clientSecret,
        };
        var url = '/checkout/cache_checkout_data/';

        $.post(url, postData).done(function () {
            stripe.confirmPayment({
                elements,
                confirmParams: {
                    return_url: `${window.location.origin}/checkout/order_pending/
                    payment_method_data: {
                        billing_details: {
                            email: billingEmail,
                            name: billingName,
                            phone: billingPhone,
                        },
                    },
                    shipping: {
                        name: fullName,
                        phone: phoneNumber,
                        address: {
                            line1: streetAddress,
```

CONFIGURE

Metrics

There are 6 functions in this file.

Function with the largest signature take 1 arguments, while the median is 0.5.

Largest function has 12 statements in it, while the median is 7.

The most complex function has a cyclomatic complexity value of 3 while the median is 1.

One warning

102 Missing semicolon.

11 undefined variables

5 Stripe
63 billingEmail
64 billingName
65 billingPhone
69 fullName
70 phoneNumber
72 streetAddress
73 townCity
74 county
75 postcode
76 country

Notes. All ‘undefined variables’ are defined in the template prior to executing this script:

```
<script>
streetAddress = "{{ shipping_details.street_address }}";
townCity = "{{ shipping_details.town_city }}";
county = "{{ shipping_details.county }}";
postcode = "{{ shipping_details.postcode }}";
country = "{{ shipping_details.country }}";
phoneNumber = "{{ shipping_details.phone_number }}";
fullName = "{{ shipping_details.first_name }} {{ shipping_details.last_name }}";
billingName = "{{ billing_details.billing_name }}";
billingEmail = "{{ billing_details.billing_email }}";
billingPhone = "{{ billing_details.billing_phone_number }}";
</script>
```

checkout/order-pending.html:

- the script lives in the order-pending page – transitional page a customer is redirected to after Payment Form has been submitted
- checks if appropriate order tracked by PaymentIntent id appears in the database
- if payment proved successful the webhook handler creates the order
- when the order appears in db the script redirects a customer to the order confirmation page

```
// This script checks the status of the order every 5 seconds
    // until the order is found or the maximum number of attempts is reached.
    // The order is being tracked using the PID passed from Django.
    // When it is found, the user is redirected to the order confirmation page.
    $('#loading-overlay').fadeToggle(100);
    // Maximum number of attempts to check order status
    const maxAttempts = 10;
    let attemptCount = 0;

    function checkOrderStatus() {
        const pid = '{{ pid }}'; // the PID passed from Django

        fetch(`/checkout/order_pending/${pid}/status/`)
            .then(response => response.json())
            .then(data => {
                if (data.order_exists) {
                    // Redirect to the order confirmation page
                    window.location.href = `/checkout/confirmation/${data.order_numb
                } else {
                    attemptCount++;
                    if (attemptCount < maxAttempts) {
                        setTimeout(checkOrderStatus, 5000); // Retry after 5 second
                    } else {
                        // If order is not found after max attempts, show an error m
                        document.getElementById('loading-message').innerText =
                            "We're having trouble confirming your order. Please con

                        $('#loading-overlay').fadeToggle(100);
                    }
                }
            })
            .catch(error => {
                console.error('Error fetching order status:', error);
                // Handle fetch error, possibly show an error message to the user
                document.getElementById('loading-message').innerText =
                    "An error occurred while checking your order status. Please try
            });
    }

    // Start checking the order status after an initial delay
    setTimeout(checkOrderStatus, 2000); // Start after 2 seconds
```

CONFIGURE

Metrics

There are 4 functions in this file.

Function with the largest signature take 1 arguments, while the median is 1.

Largest function has 7 statements in it, while the median is 2.

The most complex function has a cyclomatic complexity value of 3 while the median is 1.

user_profile/js/toggle_select_wishlist_items.js:

- toggles ‘select all’ and ‘unselect all’ checkboxes for product items in the wishlist view

```
// Get master checkbox
    const masterCheckbox = document.getElementById('wishlist-master');
    // Select all product checkboxes
    const checkboxes = document.querySelectorAll('.select-card-check');
    // Get the label for the master checkbox
    const masterLabel = document.querySelector('label[for="wishlist-master"]');

    // Add event listener to the master checkbox
    masterCheckbox.addEventListener('change', function() {
        const isChecked = masterCheckbox.checked;
        // Select or deselect all checkboxes
        checkboxes.forEach(checkbox => {
            checkbox.checked = isChecked;
        });
        // Change label text based on selection state
        masterLabel.textContent = isChecked ? 'Unselect all' : 'Select all';
    });
```

CONFIGURE

Metrics

There are 2 functions in this file.

Function with the largest signature take 1 arguments, while the median is 0.5.

Largest function has 3 statements in it, while the median is 2.

The most complex function has a cyclomatic complexity value of 2 while the median is 1.5.

user_profile/js/personal_info.js:

- serves the view-profile.html page and handles submission of all forms therein, except <password form>:
    - Personal info form
    - Contact phone form
    - Email form

- maps and wraps json responses into Bootstrap Toast messages

```
document.addEventListener('DOMContentLoaded', function () {
    const formBasic = document.querySelector('#basic_info_form');
    const formPhone = document.querySelector('#user_phone_form');

    // User Basic Info Form submission
    formBasic.addEventListener('submit', function (event) {
        event.preventDefault(); // Prevent the default form submission
        const formData = new FormData(formBasic);

        fetch(formBasic.action, {
            method: 'POST',
            body: formData,
            headers: { 'X-CSRFToken': formBasic.querySelector('[name=csrfmiddlewaret
        })
            .then(response => response.json())
            .then(data => {
                if (data.success) {
                    // Update the DOM with the new data
                    const titleNameElement = document.querySelector('#title_name');
                    const dateOfBirthElement = document.querySelector('#date_of_birt

                    // Check if the title is "Prefer not to use" and set it to an em
                    const title = data.updated_basic.title === 'Prefer not to use' ?

                    // Construct the new title name
                    const newTitleName = `${title} ${data.updated_basic.first_name}
                    titleNameElement.textContent = newTitleName;

                    // Update the date of birth:

                    const dateOfBirthString = data.updated_basic.date_of_birth;
                    // Parse the date string into a Date object
                    const dateOfBirth = new Date(dateOfBirthString);
                    // Format the date using Intl.DateTimeFormat
                    const options = { year: 'numeric', month: 'short', day: 'numeric
                    let formattedDate = new Intl.DateTimeFormat('en-US', options).fo
                    // Insert dot (`.`) after the short for a month
                    formattedDate = formattedDate.replace(/(\w{3}) /, '$1. ');

                    dateOfBirthElement.textContent = formattedDate;

                } else {
                    console.log('Form submission failed:', data.errors || data.messa
                }
                showToast(data);
            })
            .catch(error => console.log('Error:', error));
    });

    // Contact form submission
    formPhone.addEventListener('submit', function (event) {
        event.preventDefault(); // Prevent the default form submission
        const formData = new FormData(formPhone);

        fetch(formPhone.action, {
            method: 'POST',
            body: formData,
            headers: { 'X-CSRFToken': formPhone.querySelector('[name=csrfmiddlewaret
        })
            .then(response => response.json())
            .then(data => {
                if (data.success) {
                    const phoneElement = document.querySelector('#phone_profile');

                    // Update phone number on the page
                    phoneElement.textContent = data.updated_phone;

                } else {
                    console.log('Form submission failed:', data.errors || data.messa
                }
                showToast(data);
            })
```

CONFIGURE

Metrics

There are 13 functions in this file.

Function with the largest signature take 3 arguments, while the median is 1.

Largest function has 14 statements in it, while the median is 2.

The most complex function has a cyclomatic complexity value of 4 while the median is 1.

One undefined variable

117 bootstrap

Notes. 'Undefined variable' bootstrap is defined in the external Bootstrap files

user_profile/order-history.html :

- toggling arrow icons up and down when drop-down element is triggerred

```
// Toggle icon on click up or down
   $(document).ready(function () {
       $('.btn-icon').on('click', function () {
           var icon = $(this).find('.toggle-icon');
           icon.toggleClass('fa-chevrons-down fa-chevrons-up');
       });
   });
```

CONFIGURE

Metrics

There are 2 functions in this file.

Function with the largest signature take 0 arguments, while the median is 0.

Largest function has 2 statements in it, while the median is 1.5.

The most complex function has a cyclomatic complexity value of 1 while the median is 1.

user_profile/includes/add_shipping_address.html :

- handles ajax post request to submit the shipping address form

```
$(document).ready(function () {
      // Listen for the modal show event
      $('#shippingBackdrop').on('show.bs.modal', function (event) {
          var button = $(event.relatedTarget); // Button that triggered the modal
          var addressId = button.data('address-id'); // Extract info from data-* a

          if (addressId) {
              // Fetch the existing data using AJAX
              $.ajax({
                  url: '/profile/get_address_data/', // URL to fetch data
                  data: {
                      'address_id': addressId
                  },
                  success: function (data) {
                      // Pre-fill the form fields with the data
                      $('#id_delivery_first_name').val(data.delivery_first_name);
                      $('#id_delivery_last_name').val(data.delivery_last_name);
                      $('#id_delivery_email').val(data.delivery_email);
                      $('#id_delivery_phone_number').val(data.delivery_phone_numb
                      $('#id_shipping_town_city').val(data.shipping_town_city);
                      $('#id_shipping_county').val(data.shipping_county);
                      $('#id_shipping_street_address').val(data.shipping_street_a
                      $('#id_shipping_postcode').val(data.shipping_postcode);
                      $('#id_shipping_country').val(data.shipping_country);
                      // Update the form action for editing
                      var editActionUrl = '/profile/shipping_addresses/edit/' + a
                      $('#profile-shipping-details').attr('action', editActionUrl
                  }
              });
          } else {
              // Reset form for adding a new address
              $('#profile-shipping-details')[0].reset();
              $('#profile-shipping-details').attr('action', "{% url 'manage_shipp
          }
      });
  });
```

CONFIGURE

Metrics

There are 3 functions in this file.

Function with the largest signature take 1 arguments, while the median is 1.

Largest function has 11 statements in it, while the median is 6.

The most complex function has a cyclomatic complexity value of 2 while the median is 1.

vendor/add_product.html and edit_product.html :

- inserts text notification about the product image name to be uploaded

```
$('#id_image').change(function () {
      var file = $('#id_image')[0].files[0];
      $('#filename').text(`Image will be set to: ${file.name}`);
  });

```

CONFIGURE

Metrics

There is only one function in this file.

It takes no arguments.

This function contains 2 statements.

Cyclomatic complexity number for this function is 1.