Woocommerce

- Woocommerce Prevent Duplicate Orders
- Change Rendered Text for (Add to Cart, Price (Prefix/Suffix), Update Total)

Woocommerce Prevent Duplicate Orders

The purpose of this code is to create a 3-minute window after an order is placed, during which the same email address cannot be used to place another order. This helps prevent accidental duplicate orders that might occur due to double-clicking or refreshing the page during checkout.

If a user tries to place another order within this 3-minute window, they'll see an error message directing them to check their existing orders or wait before placing a new one.

This is a common practice in e-commerce to prevent unintentional duplicate transactions, improving the user experience and reducing potential order fulfillment issues.

```
 /** PREVENT DUPLICATE ORDERS */
add_filter('woocommerce_defer_transactional_emails', '__return_true' );
add_action( 'woocommerce_after_checkout_validation', 'forbid2orders', 10, 2 );
function forbid2orders( $fields, $errors ){
	$c_user_email = $fields['billing_email'];
	$c_transient_key = 'create_order_'.$c_user_email;
	$create_order = true;
	if(false === ($c_transient_results = get_transient($c_transient_key))){
		$c_transient_results = $fields['billing_email'];
		set_transient($c_transient_key, $c_transient_results, 3 * MINUTE_IN_SECONDS );
	}else{
		if($c_transient_results == $fields['billing_email']){
			$errors->add( 'validation', 'An order has already been created.  Please check your <a href="/my-account/orders/">Orders</a> or wait a few minutes to place a new one.' );
		}
	}
}
```

This code is implementing a mechanism to prevent duplicate orders in a WooCommerce-based WordPress site. Here's a breakdown of what the code is doing:

1. Deferring transactional emails:

```
add_filter('woocommerce_defer_transactional_emails', '__return_true' );
```

This line defers the sending of transactional emails in WooCommerce, which can help prevent premature email notifications.

2. Adding a custom validation hook:

```
add_action( 'woocommerce_after_checkout_validation', 'forbid2orders', 10, 2 );
```

This adds a custom function `forbid2orders` to the WooCommerce checkout validation process.

3. The `forbid2orders` function:
   - It extracts the customer's email from the checkout fields.
   - Creates a unique transient key based on the email.
   - Checks if a transient with this key already exists:
     - If it doesn't exist, it creates a new transient with the email, lasting for 3 minutes.
     - If it does exist and matches the current email, it adds an error message to prevent a duplicate order.

The purpose of this code is to create a 3-minute window after an order is placed, during which the same email address cannot be used to place another order. This helps prevent accidental duplicate orders that might occur due to double-clicking or refreshing the page during checkout.

If a user tries to place another order within this 3-minute window, they'll see an error message directing them to check their existing orders or wait before placing a new one.

This is a common practice in e-commerce to prevent unintentional duplicate transactions, improving the user experience and reducing potential order fulfillment issues.

Change Rendered Text for (Add to Cart, Price (Prefix/Suffix), Update Total)

```
// CUSTOM ADD TO CART BUTTON
add_filter( 'woocommerce_product_add_to_cart_text', 'custom_add_to_cart_price', 20, 2 ); // Shop and other archives pages
add_filter( 'woocommerce_product_single_add_to_cart_text', 'custom_add_to_cart_price', 20, 2 ); // Single product pages
function custom_add_to_cart_price( $button_text, $product ) {
   // Variable products
   if( $product->is_type('variable') ) {
      // shop and archives
      if( ! is_product() ){
         $product_price = wc_price( wc_get_price_to_display( $product, array( 'price' => $product->get_variation_price() ) ) );
         return $button_text . ' - From ' . strip_tags( $product_price );
      }
      // Single product pages
      else {
         return $button_text;
      }
   }
   // All other product types
   else {
      $product_price = wc_price( wc_get_price_to_display( $product ) );
      return 'BUY THIS ITEM FOR ' . strip_tags( $product_price );
   }
}
```

```
#region ADD SUFFIX/PREFIX TO PRICES
// https://www.businessbloomer.com/woocommerce-add-prefix-suffix-product-prices/
// Adds Suffix to WooCommerce prices
```

```
/**
 * @snippet       Adds suffix to WooCommerce prices
 * @how-to        Get CustomizeWoo.com FREE
 * @author        Rodolfo Melogli
 * @compatible    WooCommerce 3.8
 * @donate $9     https://businessbloomer.com/bloomer-armada/
 */
add_filter('woocommerce_get_price_suffix', 'bbloomer_add_price_suffix', 99, 4);
function bbloomer_add_price_suffix($html, $product, $price, $qty)
{
   $html .= ' suffix here';
   return $html;
}
// Adds prefix to WooCommerce prices
/**
 * @snippet       Adds prefix to WooCommerce prices
 * @how-to        Get CustomizeWoo.com FREE
 * @author        Rodolfo Melogli
 * @compatible    WooCommerce 3.8
 * @donate $9     https://businessbloomer.com/bloomer-armada/
 */
add_filter('woocommerce_get_price_html', 'bbloomer_add_price_prefix', 99, 2);
function bbloomer_add_price_prefix($price, $product)
{
   $price = 'Prefix here ' . $price;
   return $price;
}
#endregion
```

```
#region CHANGE TEXT 'Update totals' text on cart page to Recalculate Shipping
add_filter('gettext', 'woocomerce_text_strings', 20, 3);
function woocomerce_text_strings($translated_text, $text, $domain)
{
   switch ($translated_text) {
      case 'Update totals':
         $translated_text = __('Recalculate Shipping', 'woocommerce');
         break;
   }
   return $translated_text;
}
#endregion
```