```php
<?php
/**
 * WooCommerce Core Functions
 *
 * General core functions available on both the front-end and admin.
 *
 * @package WooCommerce\Functions
 * @version 3.3.0
 */

if ( ! defined( 'ABSPATH' ) ) {
	exit;
}

// Include core functions (available in both admin and frontend).
require WC_ABSPATH . 'includes/wc-conditional-functions.php';
require WC_ABSPATH . 'includes/wc-coupon-functions.php';
require WC_ABSPATH . 'includes/wc-user-functions.php';
require WC_ABSPATH . 'includes/wc-deprecated-functions.php';
require WC_ABSPATH . 'includes/wc-formatting-functions.php';
require WC_ABSPATH . 'includes/wc-order-functions.php';
require WC_ABSPATH . 'includes/wc-order-item-functions.php';
require WC_ABSPATH . 'includes/wc-page-functions.php';
require WC_ABSPATH . 'includes/wc-product-functions.php';
require WC_ABSPATH . 'includes/wc-stock-functions.php';
```

```
require WC_ABSPATH . 'includes/wc-account-functions.php';
require WC_ABSPATH . 'includes/wc-term-functions.php';
require WC_ABSPATH . 'includes/wc-attribute-functions.php';
require WC_ABSPATH . 'includes/wc-rest-functions.php';
require WC_ABSPATH . 'includes/wc-widget-functions.php';
require WC_ABSPATH . 'includes/wc-webhook-functions.php';

/**
 * Filters on data used in admin and frontend.
 */
add_filter( 'woocommerce_coupon_code', 'html_entity_decode' );
add_filter( 'woocommerce_coupon_code', 'sanitize_text_field' );
add_filter( 'woocommerce_coupon_code', 'wc_strtolower' );
add_filter( 'woocommerce_stock_amount', 'intval' ); // Stock amounts are integers by default.
add_filter( 'woocommerce_shipping_rate_label', 'sanitize_text_field' ); // Shipping rate label.
add_filter( 'woocommerce_attribute_label', 'wp_kses_post', 100 );

/**
 * Short Description (excerpt).
 */
add_filter( 'woocommerce_short_description', 'wptexturize' );
add_filter( 'woocommerce_short_description', 'convert_smilies' );
add_filter( 'woocommerce_short_description', 'convert_chars' );
add_filter( 'woocommerce_short_description', 'wpautop' );
add_filter( 'woocommerce_short_description', 'shortcode_unautop' );
```

```
add_filter( 'woocommerce_short_description', 'prepend_attachment' );

add_filter( 'woocommerce_short_description', 'do_shortcode', 11 ); // After wpautop().

add_filter( 'woocommerce_short_description', 'wc_format_product_short_description', 9999999 );

add_filter( 'woocommerce_short_description', 'wc_do_oembeds' );

add_filter( 'woocommerce_short_description', array( $GLOBALS['wp_embed'], 'run_shortcode' ), 8 ); // Before wpautop().


/**
 * Define a constant if it is not already defined.
 *
 * @since 3.0.0
 * @param string $name  Constant name.
 * @param mixed  $value Value.
 */
function wc_maybe_define_constant( $name, $value ) {
	if ( ! defined( $name ) ) {
		define( $name, $value );
	}
}


/**
 * Create a new order programmatically.
 *
 * Returns a new order object on success which can then be used to add additional data.
 *
 * @param  array $args Order arguments.
```

```
 * @return WC_Order|WP_Error
 */
function wc_create_order( $args = array() ) {
	$default_args = array(
		'status'        => null,
		'customer_id'   => null,
		'customer_note' => null,
		'parent'        => null,
		'created_via'   => null,
		'cart_hash'     => null,
		'order_id'      => 0,
	);

	try {
		$args  = wp_parse_args( $args, $default_args );
		$order = new WC_Order( $args['order_id'] );

		// Update props that were set (not null).
		if ( ! is_null( $args['parent'] ) ) {
			$order->set_parent_id( absint( $args['parent'] ) );
		}

		if ( ! is_null( $args['status'] ) ) {
			$order->set_status( $args['status'] );
		}
```

```
		if ( ! is_null( $args['customer_note'] ) ) {

			$order->set_customer_note( $args['customer_note'] );

		}


		if ( ! is_null( $args['customer_id'] ) ) {

			$order->set_customer_id( is_numeric( $args['customer_id'] ) ? absint( $args['customer_id'] ) : 0 );

		}


		if ( ! is_null( $args['created_via'] ) ) {

			$order->set_created_via( sanitize_text_field( $args['created_via'] ) );

		}


		if ( ! is_null( $args['cart_hash'] ) ) {

			$order->set_cart_hash( sanitize_text_field( $args['cart_hash'] ) );

		}


		// Set these fields when creating a new order but not when updating an existing order.

		if ( ! $args['order_id'] ) {

			$order->set_currency( get_woocommerce_currency() );

			$order->set_prices_include_tax( 'yes' === get_option( 'woocommerce_prices_include_tax' ) );

			$order->set_customer_ip_address( WC_Geolocation::get_ip_address() );

			$order->set_customer_user_agent( wc_get_user_agent() );

		}
```

```
                // Update other order props set automatically.
                $order->save();
        } catch ( Exception $e ) {
                return new WP_Error( 'error', $e->getMessage() );
        }

        return $order;
}

/**
 * Update an order. Uses wc_create_order.
 *
 * @param  array $args Order arguments.
 * @return WC_Order|WP_Error
 */
function wc_update_order( $args ) {
        if ( empty( $args['order_id'] ) ) {
                return new WP_Error( __( 'Invalid order ID.', 'woocommerce' ) );
        }
        return wc_create_order( $args );
}

/**
 * Get template part (for templates like the shop-loop).
```

```
 *
 * WC_TEMPLATE_DEBUG_MODE will prevent overrides in themes from taking priority.
 *
 * @param mixed  $slug Template slug.
 * @param string $name Template name (default: '').
 */
function wc_get_template_part( $slug, $name = '' ) {
	$template = '';

	// Look in yourtheme/slug-name.php and yourtheme/woocommerce/slug-name.php.
	if ( $name && ! WC_TEMPLATE_DEBUG_MODE ) {
		$template = locate_template( array( "{$slug}-{$name}.php", WC()->template_path() . "{$slug}-{$name}.php" ) );
	}

	// Get default slug-name.php.
	if ( ! $template && $name && file_exists( WC()->plugin_path() . "/templates/{$slug}-{$name}.php" ) ) {
		$template = WC()->plugin_path() . "/templates/{$slug}-{$name}.php";
	}

	// If template file doesn't exist, look in yourtheme/slug.php and yourtheme/woocommerce/slug.php.
	if ( ! $template && ! WC_TEMPLATE_DEBUG_MODE ) {
		$template = locate_template( array( "{$slug}.php", WC()->template_path() . "{$slug}.php" ) );
	}
```

```
	// Allow 3rd party plugins to filter template file from their plugin.
	$template = apply_filters( 'wc_get_template_part', $template, $slug, $name );

	if ( $template ) {
		load_template( $template, false );
	}
}

/**
 * Get other templates (e.g. product attributes) passing attributes and including the file.
 *
 * @param string $template_name Template name.
 * @param array  $args          Arguments. (default: array).
 * @param string $template_path Template path. (default: '').
 * @param string $default_path  Default path. (default: '').
 */
function wc_get_template( $template_name, $args = array(), $template_path = '', $default_path = '' ) {
	if ( ! empty( $args ) && is_array( $args ) ) {
		extract( $args ); // @codingStandardsIgnoreLine
	}

	$located = wc_locate_template( $template_name, $template_path, $default_path );

	if ( ! file_exists( $located ) ) {
```

```
                /* translators: %s template */
                wc_doing_it_wrong( __FUNCTION__, sprintf( __( '%s does not exist.', 'woocommerce' ), '<code>' . $located . '</code>' ), '2.1' );
                return;
        }

        // Allow 3rd party plugin filter template file from their plugin.
        $located = apply_filters( 'wc_get_template', $located, $template_name, $args, $template_path, $default_path );

        do_action( 'woocommerce_before_template_part', $template_name, $template_path, $located, $args );

        include $located;

        do_action( 'woocommerce_after_template_part', $template_name, $template_path, $located, $args );
}

/**
 * Like wc_get_template, but returns the HTML instead of outputting.
 *
 * @see wc_get_template
 * @since 2.5.0
 * @param string $template_name Template name.
 * @param array  $args          Arguments. (default: array).
```

```
 * @param string $template_path Template path. (default: '').
 * @param string $default_path  Default path. (default: '').
 *
 * @return string
 */
function wc_get_template_html( $template_name, $args = array(), $template_path = '', $default_path = '' ) {
	ob_start();
	wc_get_template( $template_name, $args, $template_path, $default_path );
	return ob_get_clean();
}
/**
 * Locate a template and return the path for inclusion.
 *
 * This is the load order:
 *
 * yourtheme/$template_path/$template_name
 * yourtheme/$template_name
 * $default_path/$template_name
 *
 * @param string $template_name Template name.
 * @param string $template_path Template path. (default: '').
 * @param string $default_path  Default path. (default: '').
 * @return string
 */
function wc_locate_template( $template_name, $template_path = '', $default_path = '' ) {
```

```
	if ( ! $template_path ) {
		$template_path = WC()->template_path();
	}

	if ( ! $default_path ) {
		$default_path = WC()->plugin_path() . '/templates/';
	}

	// Look within passed path within the theme - this is priority.
	$template = locate_template(
		array(
			trailingslashit( $template_path ) . $template_name,
			$template_name,
		)
	);

	// Get default template/.
	if ( ! $template || WC_TEMPLATE_DEBUG_MODE ) {
		$template = $default_path . $template_name;
	}

	// Return what we found.
	return apply_filters( 'woocommerce_locate_template', $template, $template_name, $template_path );
}
```

```
/**
 * Get Base Currency Code.
 *
 * @return string
 */
function get_woocommerce_currency() {
	return apply_filters( 'woocommerce_currency', get_option( 'woocommerce_currency' ) );
}


/**
 * Get full list of currency codes.
 *
 * @return array
 */
function get_woocommerce_currencies() {
	static $currencies;

	if ( ! isset( $currencies ) ) {
		$currencies = array_unique(
			apply_filters(
				'woocommerce_currencies',
				array(
					'AED' => __( 'United Arab Emirates dirham', 'woocommerce' ),
					'AFN' => __( 'Afghan afghani', 'woocommerce' ),
					'ALL' => __( 'Albanian lek', 'woocommerce' ),
```

```
'AMD' => __( 'Armenian dram', 'woocommerce' ),
'ANG' => __( 'Netherlands Antillean guilder', 'woocommerce' ),
'AOA' => __( 'Angolan kwanza', 'woocommerce' ),
'ARS' => __( 'Argentine peso', 'woocommerce' ),
'AUD' => __( 'Australian dollar', 'woocommerce' ),
'AWG' => __( 'Aruban florin', 'woocommerce' ),
'AZN' => __( 'Azerbaijani manat', 'woocommerce' ),
'BAM' => __( 'Bosnia and Herzegovina convertible mark',
'woocommerce' ),
'BBD' => __( 'Barbadian dollar', 'woocommerce' ),
'BDT' => __( 'Bangladeshi taka', 'woocommerce' ),
'BGN' => __( 'Bulgarian lev', 'woocommerce' ),
'BHD' => __( 'Bahraini dinar', 'woocommerce' ),
'BIF' => __( 'Burundian franc', 'woocommerce' ),
'BMD' => __( 'Bermudian dollar', 'woocommerce' ),
'BND' => __( 'Brunei dollar', 'woocommerce' ),
'BOB' => __( 'Bolivian boliviano', 'woocommerce' ),
'BRL' => __( 'Brazilian real', 'woocommerce' ),
'BSD' => __( 'Bahamian dollar', 'woocommerce' ),
'BTC' => __( 'Bitcoin', 'woocommerce' ),
'BTN' => __( 'Bhutanese ngultrum', 'woocommerce' ),
'BWP' => __( 'Botswana pula', 'woocommerce' ),
'BYR' => __( 'Belarusian ruble (old)', 'woocommerce' ),
'BYN' => __( 'Belarusian ruble', 'woocommerce' ),
'BZD' => __( 'Belize dollar', 'woocommerce' ),
'CAD' => __( 'Canadian dollar', 'woocommerce' ),
```

```
'CDF' => __( 'Congolese franc', 'woocommerce' ),

'CHF' => __( 'Swiss franc', 'woocommerce' ),

'CLP' => __( 'Chilean peso', 'woocommerce' ),

'CNY' => __( 'Chinese yuan', 'woocommerce' ),

'COP' => __( 'Colombian peso', 'woocommerce' ),

'CRC' => __( 'Costa Rican col&oacute;n', 'woocommerce' ),

'CUC' => __( 'Cuban convertible peso', 'woocommerce' ),

'CUP' => __( 'Cuban peso', 'woocommerce' ),

'CVE' => __( 'Cape Verdean escudo', 'woocommerce' ),

'CZK' => __( 'Czech koruna', 'woocommerce' ),

'DJF' => __( 'Djiboutian franc', 'woocommerce' ),

'DKK' => __( 'Danish krone', 'woocommerce' ),

'DOP' => __( 'Dominican peso', 'woocommerce' ),

'DZD' => __( 'Algerian dinar', 'woocommerce' ),

'EGP' => __( 'Egyptian pound', 'woocommerce' ),

'ERN' => __( 'Eritrean nakfa', 'woocommerce' ),

'ETB' => __( 'Ethiopian birr', 'woocommerce' ),

'EUR' => __( 'Euro', 'woocommerce' ),

'FJD' => __( 'Fijian dollar', 'woocommerce' ),

'FKP' => __( 'Falkland Islands pound', 'woocommerce' ),

'GBP' => __( 'Pound sterling', 'woocommerce' ),

'GEL' => __( 'Georgian lari', 'woocommerce' ),

'GGP' => __( 'Guernsey pound', 'woocommerce' ),

'GHS' => __( 'Ghana cedi', 'woocommerce' ),

'GIP' => __( 'Gibraltar pound', 'woocommerce' ),
```

```
'GMD' => __( 'Gambian dalasi', 'woocommerce' ),
'GNF' => __( 'Guinean franc', 'woocommerce' ),
'GTQ' => __( 'Guatemalan quetzal', 'woocommerce' ),
'GYD' => __( 'Guyanese dollar', 'woocommerce' ),
'HKD' => __( 'Hong Kong dollar', 'woocommerce' ),
'HNL' => __( 'Honduran lempira', 'woocommerce' ),
'HRK' => __( 'Croatian kuna', 'woocommerce' ),
'HTG' => __( 'Haitian gourde', 'woocommerce' ),
'HUF' => __( 'Hungarian forint', 'woocommerce' ),
'IDR' => __( 'Indonesian rupiah', 'woocommerce' ),
'ILS' => __( 'Israeli new shekel', 'woocommerce' ),
'IMP' => __( 'Manx pound', 'woocommerce' ),
'INR' => __( 'Indian rupee', 'woocommerce' ),
'IQD' => __( 'Iraqi dinar', 'woocommerce' ),
'IRR' => __( 'Iranian rial', 'woocommerce' ),
'IRT' => __( 'Iranian toman', 'woocommerce' ),
'ISK' => __( 'Icelandic kr&oacute;na', 'woocommerce' ),
'JEP' => __( 'Jersey pound', 'woocommerce' ),
'JMD' => __( 'Jamaican dollar', 'woocommerce' ),
'JOD' => __( 'Jordanian dinar', 'woocommerce' ),
'JPY' => __( 'Japanese yen', 'woocommerce' ),
'KES' => __( 'Kenyan shilling', 'woocommerce' ),
'KGS' => __( 'Kyrgyzstani som', 'woocommerce' ),
'KHR' => __( 'Cambodian riel', 'woocommerce' ),
'KMF' => __( 'Comorian franc', 'woocommerce' ),
```

```
'KPW' => __( 'North Korean won', 'woocommerce' ),

'KRW' => __( 'South Korean won', 'woocommerce' ),

'KWD' => __( 'Kuwaiti dinar', 'woocommerce' ),

'KYD' => __( 'Cayman Islands dollar', 'woocommerce' ),

'KZT' => __( 'Kazakhstani tenge', 'woocommerce' ),

'LAK' => __( 'Lao kip', 'woocommerce' ),

'LBP' => __( 'Lebanese pound', 'woocommerce' ),

'LKR' => __( 'Sri Lankan rupee', 'woocommerce' ),

'LRD' => __( 'Liberian dollar', 'woocommerce' ),

'LSL' => __( 'Lesotho loti', 'woocommerce' ),

'LYD' => __( 'Libyan dinar', 'woocommerce' ),

'MAD' => __( 'Moroccan dirham', 'woocommerce' ),

'MDL' => __( 'Moldovan leu', 'woocommerce' ),

'MGA' => __( 'Malagasy ariary', 'woocommerce' ),

'MKD' => __( 'Macedonian denar', 'woocommerce' ),

'MMK' => __( 'Burmese kyat', 'woocommerce' ),

'MNT' => __( 'Mongolian t&ouml;gr&ouml;g', 'woocommerce' ),

'MOP' => __( 'Macanese pataca', 'woocommerce' ),

'MRO' => __( 'Mauritanian ouguiya', 'woocommerce' ),

'MUR' => __( 'Mauritian rupee', 'woocommerce' ),

'MVR' => __( 'Maldivian rufiyaa', 'woocommerce' ),

'MWK' => __( 'Malawian kwacha', 'woocommerce' ),

'MXN' => __( 'Mexican peso', 'woocommerce' ),

'MYR' => __( 'Malaysian ringgit', 'woocommerce' ),

'MZN' => __( 'Mozambican metical', 'woocommerce' ),
```

```
'NAD' => __( 'Namibian dollar', 'woocommerce' ),

'NGN' => __( 'Nigerian naira', 'woocommerce' ),

'NIO' => __( 'Nicaraguan c&oacute;rdoba', 'woocommerce' ),

'NOK' => __( 'Norwegian krone', 'woocommerce' ),

'NPR' => __( 'Nepalese rupee', 'woocommerce' ),

'NZD' => __( 'New Zealand dollar', 'woocommerce' ),

'OMR' => __( 'Omani rial', 'woocommerce' ),

'PAB' => __( 'Panamanian balboa', 'woocommerce' ),

'PEN' => __( 'Peruvian nuevo sol', 'woocommerce' ),

'PGK' => __( 'Papua New Guinean kina', 'woocommerce' ),

'PHP' => __( 'Philippine peso', 'woocommerce' ),

'PKR' => __( 'Pakistani rupee', 'woocommerce' ),

'PLN' => __( 'Polish z&#x142;oty', 'woocommerce' ),

'PRB' => __( 'Transnistrian ruble', 'woocommerce' ),

'PYG' => __( 'Paraguayan guaran&iacute;', 'woocommerce' ),

'QAR' => __( 'Qatari riyal', 'woocommerce' ),

'RON' => __( 'Romanian leu', 'woocommerce' ),

'RSD' => __( 'Serbian dinar', 'woocommerce' ),

'RUB' => __( 'Russian ruble', 'woocommerce' ),

'RWF' => __( 'Rwandan franc', 'woocommerce' ),

'SAR' => __( 'Saudi riyal', 'woocommerce' ),

'SBD' => __( 'Solomon Islands dollar', 'woocommerce' ),

'SCR' => __( 'Seychellois rupee', 'woocommerce' ),

'SDG' => __( 'Sudanese pound', 'woocommerce' ),

'SEK' => __( 'Swedish krona', 'woocommerce' ),
```

```
'SGD' => __( 'Singapore dollar', 'woocommerce' ),
'SHP' => __( 'Saint Helena pound', 'woocommerce' ),
'SLL' => __( 'Sierra Leonean leone', 'woocommerce' ),
'SOS' => __( 'Somali shilling', 'woocommerce' ),
'SRD' => __( 'Surinamese dollar', 'woocommerce' ),
'SSP' => __( 'South Sudanese pound', 'woocommerce' ),
'STD' => __( 'S&atilde;o Tom&eacute; and Pr&iacute;ncipe dobra', 'woocommerce' ),
'SYP' => __( 'Syrian pound', 'woocommerce' ),
'SZL' => __( 'Swazi lilangeni', 'woocommerce' ),
'THB' => __( 'Thai baht', 'woocommerce' ),
'TJS' => __( 'Tajikistani somoni', 'woocommerce' ),
'TMT' => __( 'Turkmenistan manat', 'woocommerce' ),
'TND' => __( 'Tunisian dinar', 'woocommerce' ),
'TOP' => __( 'Tongan pa&#x2bb;anga', 'woocommerce' ),
'TRY' => __( 'Turkish lira', 'woocommerce' ),
'TTD' => __( 'Trinidad and Tobago dollar', 'woocommerce' ),
'TWD' => __( 'New Taiwan dollar', 'woocommerce' ),
'TZS' => __( 'Tanzanian shilling', 'woocommerce' ),
'UAH' => __( 'Ukrainian hryvnia', 'woocommerce' ),
'UGX' => __( 'Ugandan shilling', 'woocommerce' ),
'USD' => __( 'United States (US) dollar', 'woocommerce' ),
'UYU' => __( 'Uruguayan peso', 'woocommerce' ),
'UZS' => __( 'Uzbekistani som', 'woocommerce' ),
'VEF' => __( 'Venezuelan bol&iacute;var', 'woocommerce' ),
```

```
                                        'VND' => __( 'Vietnamese &#x111;&#x1ed3;ng', 'woocommerce'
),
                                        'VUV' => __( 'Vanuatu vatu', 'woocommerce' ),
                                        'WST' => __( 'Samoan t&#x101;l&#x101;', 'woocommerce' ),
                                        'XAF' => __( 'Central African CFA franc', 'woocommerce' ),
                                        'XCD' => __( 'East Caribbean dollar', 'woocommerce' ),
                                        'XOF' => __( 'West African CFA franc', 'woocommerce' ),
                                        'XPF' => __( 'CFP franc', 'woocommerce' ),
                                        'YER' => __( 'Yemeni rial', 'woocommerce' ),
                                        'ZAR' => __( 'South African rand', 'woocommerce' ),
                                        'ZMW' => __( 'Zambian kwacha', 'woocommerce' ),
                                    )
                                )
                        );
                }

                return $currencies;
}



/**
 * Get Currency symbol.
 *
 * @param string $currency Currency. (default: '').
 * @return string
 */
```

```
function get_woocommerce_currency_symbol( $currency = '' ) {
	if ( ! $currency ) {
		$currency = get_woocommerce_currency();
	}

	$symbols 	= apply_filters(
		'woocommerce_currency_symbols', array(
			'AED' => '&#x62f;.&#x625;',
			'AFN' => '&#x60b;',
			'ALL' => 'L',
			'AMD' => 'AMD',
			'ANG' => '&fnof;',
			'AOA' => 'Kz',
			'ARS' => '&#36;',
			'AUD' => '&#36;',
			'AWG' => 'Afl.',
			'AZN' => 'AZN',
			'BAM' => 'KM',
			'BBD' => '&#36;',
			'BDT' => '&#2547; ',
			'BGN' => '&#1083;&#1074;.',
			'BHD' => '.&#x62f;.&#x628;',
			'BIF' => 'Fr',
			'BMD' => '&#36;',
			'BND' => '&#36;',
```

'BOB' => 'Bs.',

'BRL' => 'R$',

'BSD' => '$',

'BTC' => '฿',

'BTN' => 'Nu.',

'BWP' => 'P',

'BYR' => 'Br',

'BYN' => 'Br',

'BZD' => '$',

'CAD' => '$',

'CDF' => 'Fr',

'CHF' => 'CHF',

'CLP' => '$',

'CNY' => '¥',

'COP' => '$',

'CRC' => '₡',

'CUC' => '$',

'CUP' => '$',

'CVE' => '$',

'CZK' => 'Kč',

'DJF' => 'Fr',

'DKK' => 'DKK',

'DOP' => 'RD$',

'DZD' => 'د.ج',

'EGP' => 'EGP',

```
'ERN' => 'Nfk',

'ETB' => 'Br',

'EUR' => '&euro;',

'FJD' => '&#36;',

'FKP' => '&pound;',

'GBP' => '&pound;',

'GEL' => '&#x20be;',

'GGP' => '&pound;',

'GHS' => '&#x20b5;',

'GIP' => '&pound;',

'GMD' => 'D',

'GNF' => 'Fr',

'GTQ' => 'Q',

'GYD' => '&#36;',

'HKD' => '&#36;',

'HNL' => 'L',

'HRK' => 'kn',

'HTG' => 'G',

'HUF' => '&#70;&#116;',

'IDR' => 'Rp',

'ILS' => '&#8362;',

'IMP' => '&pound;',

'INR' => '&#8377;',

'IQD' => '&#x639;.&#x62f;',

'IRR' => '&#xfdfc;',
```

'IRT' => 'تومان',

'ISK' => 'kr.',

'JEP' => '£',

'JMD' => '$',

'JOD' => 'د.ا',

'JPY' => '¥',

'KES' => 'KSh',

'KGS' => 'сом',

'KHR' => '៛',

'KMF' => 'Fr',

'KPW' => '₩',

'KRW' => '₩',

'KWD' => 'د.ك',

'KYD' => '$',

'KZT' => 'KZT',

'LAK' => '₭',

'LBP' => 'ل.ل',

'LKR' => 'රු',

'LRD' => '$',

'LSL' => 'L',

'LYD' => 'ل.د',

'MAD' => 'د.م.',

'MDL' => 'MDL',

'MGA' => 'Ar',

'MKD' => 'ден',

'MMK' => 'Ks',

'MNT' => '₮',

'MOP' => 'P',

'MRO' => 'UM',

'MUR' => '₨',

'MVR' => '.ރ',

'MWK' => 'MK',

'MXN' => '$',

'MYR' => 'RM',

'MZN' => 'MT',

'NAD' => '$',

'NGN' => '₦',

'NIO' => 'C$',

'NOK' => 'kr',

'NPR' => '₨',

'NZD' => '$',

'OMR' => 'ر.ع.',

'PAB' => 'B/.',

'PEN' => 'S/.',

'PGK' => 'K',

'PHP' => '₱',

'PKR' => '₨',

'PLN' => 'zł',

'PRB' => 'р.',

'PYG' => '₲',

'QAR' => 'ر.ق',

'RMB' => '¥',

'RON' => 'lei',

'RSD' => 'дин.',

'RUB' => '₽',

'RWF' => 'Fr',

'SAR' => 'ر.س',

'SBD' => '$',

'SCR' => '₨',

'SDG' => 'ج.س.',

'SEK' => 'kr',

'SGD' => 'SG$',

'SHP' => '£',

'SLL' => 'Le',

'SOS' => 'Sh',

'SRD' => '$',

'SSP' => '£',

'STD' => 'Db',

'SYP' => 'ل.س',

'SZL' => 'L',

'THB' => '฿',

'TJS' => 'ЅМ',

'TMT' => 'm',

'TND' => 'د.ت',

'TOP' => 'T$',

```
                'TRY' => '&#8378;',
                'TTD' => '&#36;',
                'TWD' => '&#78;&#84;&#36;',
                'TZS' => 'Sh',
                'UAH' => '&#8372;',
                'UGX' => 'UGX',
                'USD' => '&#36;',
                'UYU' => '&#36;',
                'UZS' => 'UZS',
                'VEF' => 'Bs F',
                'VND' => '&#8363;',
                'VUV' => 'Vt',
                'WST' => 'T',
                'XAF' => 'CFA',
                'XCD' => '&#36;',
                'XOF' => 'CFA',
                'XPF' => 'Fr',
                'YER' => '&#xfdfc;',
                'ZAR' => '&#82;',
                'ZMW' => 'ZK',
            )
);
$currency_symbol = isset( $symbols[ $currency ] ) ? $symbols[ $currency ] : '';

return apply_filters( 'woocommerce_currency_symbol', $currency_symbol, $currency );
```

```
}


/**
 * Send HTML emails from WooCommerce.
 *
 * @param mixed  $to         Receiver.
 * @param mixed  $subject    Subject.
 * @param mixed  $message    Message.
 * @param string $headers    Headers. (default: "Content-Type: text/html\r\n").
 * @param string $attachments Attachments. (default: "").
 */
function wc_mail( $to, $subject, $message, $headers = "Content-Type: text/html\r\n", $attachments = '' ) {
	$mailer = WC()->mailer();

	$mailer->send( $to, $subject, $message, $headers, $attachments );
}


/**
 * Return "theme support" values from the current theme, if set.
 *
 * @since  3.3.0
 * @param  string $prop Name of prop (or key::subkey for arrays of props) if you want a specific value. Leave blank to get all props as an array.
 * @param  mixed  $default Optional value to return if the theme does not declare support for a prop.
 * @return mixed  Value of prop(s).
```

```
 */
function wc_get_theme_support( $prop = '', $default = null ) {
	$theme_support = get_theme_support( 'woocommerce' );
	$theme_support = is_array( $theme_support ) ? $theme_support[0] : false;

	if ( ! $theme_support ) {
		return $default;
	}

	if ( $prop ) {
		$prop_stack = explode( '::', $prop );
		$prop_key   = array_shift( $prop_stack );

		if ( isset( $theme_support[ $prop_key ] ) ) {
			$value = $theme_support[ $prop_key ];

			if ( count( $prop_stack ) ) {
				foreach ( $prop_stack as $prop_key ) {
					if ( is_array( $value ) && isset( $value[ $prop_key ] ) ) {
						$value = $value[ $prop_key ];
					} else {
						$value = $default;
						break;
					}
				}
```

```
                        }
                } else {
                        $value = $default;
                }

                return $value;
        }

        return $theme_support;
}

/**
 * Get an image size by name or defined dimensions.
 *
 * The returned variable is filtered by woocommerce_get_image_size_{image_size} filter to
 * allow 3rd party customisation.
 *
 * Sizes defined by the theme take priority over settings. Settings are hidden when a theme
 * defines sizes.
 *
 * @param array|string $image_size Name of the image size to get, or an array of dimensions.
 * @return array Array of dimensions including width, height, and cropping mode. Cropping mode is 0 for no crop, and 1 for hard crop.
 */
function wc_get_image_size( $image_size ) {
        $size = array(
```

```
		'width'  => 600,
		'height' => 600,
		'crop'   => 1,
);

if ( is_array( $image_size ) ) {
		$size      = array(
			'width'  => isset( $image_size[0] ) ? absint( $image_size[0] ) : 600,
			'height' => isset( $image_size[1] ) ? absint( $image_size[1] ) : 600,
			'crop'   => isset( $image_size[2] ) ? absint( $image_size[2] ) : 1,
		);
		$image_size = $size['width'] . '_' . $size['height'];
} else {
		$image_size = str_replace( 'woocommerce_', '', $image_size );

		// Legacy size mapping.
		if ( 'shop_single' === $image_size ) {
			$image_size = 'single';
		} elseif ( 'shop_catalog' === $image_size ) {
			$image_size = 'thumbnail';
		} elseif ( 'shop_thumbnail' === $image_size ) {
			$image_size = 'gallery_thumbnail';
		}

		if ( 'single' === $image_size ) {
```

```
                $size['width']  = absint( wc_get_theme_support( 'single_image_width', get_option( 'woocommerce_single_image_width', 600 ) ) );

                $size['height'] = '';

                $size['crop']   = 0;


            } elseif ( 'gallery_thumbnail' === $image_size ) {

                $size['width']  = absint( wc_get_theme_support( 'gallery_thumbnail_image_width', 100 ) );

                $size['height'] = $size['width'];

                $size['crop']   = 1;


            } elseif ( 'thumbnail' === $image_size ) {

                $size['width'] = absint( wc_get_theme_support( 'thumbnail_image_width', get_option( 'woocommerce_thumbnail_image_width', 300 ) ) );

                $cropping      = get_option( 'woocommerce_thumbnail_cropping', '1:1' );


                if ( 'uncropped' === $cropping ) {

                    $size['height'] = '';

                    $size['crop']   = 0;

                } elseif ( 'custom' === $cropping ) {

                    $width         = max( 1, get_option( 'woocommerce_thumbnail_cropping_custom_width', '4' ) );

                    $height        = max( 1, get_option( 'woocommerce_thumbnail_cropping_custom_height', '3' ) );

                    $size['height'] = absint( round( ( $size['width'] / $width ) * $height ) );

                    $size['crop']   = 1;

                } else {
```

```
                                $cropping_split = explode( ':', $cropping );
                                $width          = max( 1, current( $cropping_split ) );
                                $height         = max( 1, end( $cropping_split ) );
                                $size['height'] = absint( round( ( $size['width'] / $width ) * $height ) );
                                $size['crop']   = 1;
                        }
                }
        }

        return apply_filters( 'woocommerce_get_image_size_' . $image_size, $size );
}

/**
 * Queue some JavaScript code to be output in the footer.
 *
 * @param string $code Code.
 */
function wc_enqueue_js( $code ) {
        global $wc_queued_js;

        if ( empty( $wc_queued_js ) ) {
                $wc_queued_js = '';
        }

        $wc_queued_js .= "\n" . $code . "\n";
```

```
}

/**
 * Output any queued javascript code in the footer.
 */
function wc_print_js() {
	global $wc_queued_js;

	if ( ! empty( $wc_queued_js ) ) {
		// Sanitize.
		$wc_queued_js = wp_check_invalid_utf8( $wc_queued_js );
		$wc_queued_js = preg_replace( '/&#(x)?0*(?(1)27|39);?/i', "'", $wc_queued_js );
		$wc_queued_js = str_replace( "\r", '', $wc_queued_js );

		$js = "<!-- WooCommerce JavaScript -->\n<script type=\"text/javascript\">\njQuery(function($) { $wc_queued_js });\n</script>\n";

		/**
		 * Queued jsfilter.
		 *
		 * @since 2.6.0
		 * @param string $js JavaScript code.
		 */
		echo apply_filters( 'woocommerce_queued_js', $js ); // WPCS: XSS ok.

		unset( $wc_queued_js );
```

```
	}
}

/**
 * Set a cookie - wrapper for setcookie using WP constants.
 *
 * @param  string  $name   Name of the cookie being set.
 * @param  string  $value  Value of the cookie.
 * @param  integer $expire Expiry of the cookie.
 * @param  bool    $secure Whether the cookie should be served only over https.
 */
function wc_setcookie( $name, $value, $expire = 0, $secure = false ) {
	if ( ! headers_sent() ) {
		setcookie( $name, $value, $expire, COOKIEPATH ? COOKIEPATH : '/', COOKIE_DOMAIN, $secure, apply_filters( 'woocommerce_cookie_httponly', false, $name, $value, $expire, $secure ) );
	} elseif ( defined( 'WP_DEBUG' ) && WP_DEBUG ) {
		headers_sent( $file, $line );
		trigger_error( "{$name} cookie cannot be set - headers already sent by {$file} on line {$line}", E_USER_NOTICE ); // @codingStandardsIgnoreLine
	}
}

/**
 * Get the URL to the WooCommerce REST API.
 *
 * @since 2.1
```

```
 * @param string $path an endpoint to include in the URL.
 * @return string the URL.
 */
function get_woocommerce_api_url( $path ) {
	$version = defined( 'WC_API_REQUEST_VERSION' ) ? WC_API_REQUEST_VERSION : substr( WC_API::VERSION, 0, 1 );

	$url = get_home_url( null, "wc-api/v{$version}/", is_ssl() ? 'https' : 'http' );

	if ( ! empty( $path ) && is_string( $path ) ) {
		$url .= ltrim( $path, '/' );
	}

	return $url;
}

/**
 * Get a log file path.
 *
 * @since 2.2
 *
 * @param string $handle name.
 * @return string the log file path.
 */
function wc_get_log_file_path( $handle ) {
	return WC_Log_Handler_File::get_log_file_path( $handle );
```

```
}

/**
 * Get a log file name.
 *
 * @since 3.3
 *
 * @param string $handle Name.
 * @return string The log file name.
 */
function wc_get_log_file_name( $handle ) {
	return WC_Log_Handler_File::get_log_file_name( $handle );
}

/**
 * Recursively get page children.
 *
 * @param  int $page_id Page ID.
 * @return int[]
 */
function wc_get_page_children( $page_id ) {
	$page_ids = get_posts(
		array(
			'post_parent' => $page_id,
			'post_type'   => 'page',
```

```
                        'numberposts' => -1, // @codingStandardsIgnoreLine
                        'post_status' => 'any',
                        'fields'      => 'ids',
                )
        );

        if ( ! empty( $page_ids ) ) {
                foreach ( $page_ids as $page_id ) {
                        $page_ids = array_merge( $page_ids, wc_get_page_children( $page_id ) );
                }
        }

        return $page_ids;
}

/**
 * Flushes rewrite rules when the shop page (or it's children) gets saved.
 */
function flush_rewrite_rules_on_shop_page_save() {
        $screen    = get_current_screen();
        $screen_id = $screen ? $screen->id : '';

        // Check if this is the edit page.
        if ( 'page' !== $screen_id ) {
                return;
```

```
	}


	// Check if page is edited.
	if ( empty( $_GET['post'] ) || empty( $_GET['action'] ) || ( isset( $_GET['action'] ) && 'edit' !== $_GET['action'] ) ) { // WPCS: input var ok, CSRF ok.
		return;
	}

	$post_id      = intval( $_GET['post'] ); // WPCS: input var ok, CSRF ok.
	$shop_page_id = wc_get_page_id( 'shop' );

	if ( $shop_page_id === $post_id || in_array( $post_id, wc_get_page_children( $shop_page_id ), true ) ) {
		do_action( 'woocommerce_flush_rewrite_rules' );
	}
}
add_action( 'admin_footer', 'flush_rewrite_rules_on_shop_page_save' );

/**
 * Various rewrite rule fixes.
 *
 * @since 2.2
 * @param array $rules Rules.
 * @return array
 */
function wc_fix_rewrite_rules( $rules ) {
```

```
	global $wp_rewrite;


	$permalinks = wc_get_permalink_structure();


	// Fix the rewrite rules when the product permalink have %product_cat% flag.
	if ( preg_match( '`/(.+)(/%product_cat%)`', $permalinks['product_rewrite_slug'], $matches ) ) {
		foreach ( $rules as $rule => $rewrite ) {
			if ( preg_match( '`^' . preg_quote( $matches[1], '`' ) . '/\(`', $rule ) &&
preg_match( '/^(index\.php\?product_cat)(?!(.*product))/', $rewrite ) ) {
				unset( $rules[ $rule ] );
			}
		}
	}


	// If the shop page is used as the base, we need to handle shop page subpages to avoid 404s.
	if ( ! $permalinks['use_verbose_page_rules'] ) {
		return $rules;
	}


	$shop_page_id = wc_get_page_id( 'shop' );
	if ( $shop_page_id ) {
		$page_rewrite_rules = array();
		$subpages           = wc_get_page_children( $shop_page_id );


		// Subpage rules.
		foreach ( $subpages as $subpage ) {
```

```
                        $uri                = get_page_uri( $subpage );
                        $page_rewrite_rules[ $uri . '/?$' ] = 'index.php?pagename=' . $uri;
                        $wp_generated_rewrite_rules = $wp_rewrite->generate_rewrite_rules( $uri, EP_PAGES, true, true, false, false );
                        foreach ( $wp_generated_rewrite_rules as $key => $value ) {
                                $wp_generated_rewrite_rules[ $key ] = $value . '&pagename=' . $uri;
                        }
                        $page_rewrite_rules = array_merge( $page_rewrite_rules, $wp_generated_rewrite_rules );
                }

                // Merge with rules.
                $rules = array_merge( $page_rewrite_rules, $rules );
        }

        return $rules;
}
add_filter( 'rewrite_rules_array', 'wc_fix_rewrite_rules' );

/**
 * Prevent product attachment links from breaking when using complex rewrite structures.
 *
 * @param  string $link    Link.
 * @param  int    $post_id Post ID.
 * @return string
 */
```

```
function wc_fix_product_attachment_link( $link, $post_id ) {
	$parent_type = get_post_type( wp_get_post_parent_id( $post_id ) );
	if ( 'product' === $parent_type || 'product_variation' === $parent_type ) {
		$link = home_url( '/?attachment_id=' . $post_id );
	}
	return $link;
}
add_filter( 'attachment_link', 'wc_fix_product_attachment_link', 10, 2 );

/**
 * Protect downloads from ms-files.php in multisite.
 *
 * @param string $rewrite rewrite rules.
 * @return string
 */
function wc_ms_protect_download_rewite_rules( $rewrite ) {
	if ( ! is_multisite() || 'redirect' === get_option( 'woocommerce_file_download_method' ) ) {
		return $rewrite;
	}

	$rule  = "\n# WooCommerce Rules - Protect Files from ms-files.php\n\n";
	$rule .= "<IfModule mod_rewrite.c>\n";
	$rule .= "RewriteEngine On\n";
	$rule .= "RewriteCond %{QUERY_STRING} file=woocommerce_uploads/ [NC]\n";
	$rule .= "RewriteRule /ms-files.php$ - [F]\n";
```

```
		$rule .= "</IfModule>\n\n";


		return $rule . $rewrite;
}
add_filter( 'mod_rewrite_rules', 'wc_ms_protect_download_rewite_rules' );


/**
 * Formats a string in the format COUNTRY:STATE into an array.
 *
 * @since 2.3.0
 * @param  string $country_string Country string.
 * @return array
 */
function wc_format_country_state_string( $country_string ) {
		if ( strstr( $country_string, ':' ) ) {
			list( $country, $state ) = explode( ':', $country_string );
		} else {
			$country = $country_string;
			$state   = '';
		}
		return array(
			'country' => $country,
			'state'   => $state,
		);
}
```

```
/**
 * Get the store's base location.
 *
 * @since 2.3.0
 * @return array
 */
function wc_get_base_location() {
	$default = apply_filters( 'woocommerce_get_base_location', get_option( 'woocommerce_default_country' ) );

	return wc_format_country_state_string( $default );
}

/**
 * Get the customer's default location.
 *
 * Filtered, and set to base location or left blank. If cache-busting,
 * this should only be used when 'location' is set in the querystring.
 *
 * @since 2.3.0
 * @return array
 */
function wc_get_customer_default_location() {
	$location = array();
```

```
	switch ( get_option( 'woocommerce_default_customer_address' ) ) {
		case 'geolocation_ajax':
		case 'geolocation':
			// Exclude common bots from geolocation by user agent.
			$ua = strtolower( wc_get_user_agent() );

			if ( ! strstr( $ua, 'bot' ) && ! strstr( $ua, 'spider' ) && ! strstr( $ua, 'crawl' ) ) {
				$location = WC_Geolocation::geolocate_ip( '', true, false );
			}

			// Base fallback.
			if ( empty( $location['country'] ) ) {
				$location = wc_format_country_state_string( apply_filters( 'woocommerce_customer_default_location', get_option( 'woocommerce_default_country' ) ) );
			}
			break;
		case 'base':
			$location = wc_format_country_state_string( apply_filters( 'woocommerce_customer_default_location', get_option( 'woocommerce_default_country' ) ) );
			break;
		default:
			$countries = WC()->countries->get_allowed_countries();
			$location  = wc_format_country_state_string( apply_filters( 'woocommerce_customer_default_location', 1 === count( $countries ) ? key( $countries ) : '' ) );
			break;
	}
```

```
	return apply_filters( 'woocommerce_customer_default_location_array', $location );
}

/**
 * Get user agent string.
 *
 * @since  3.0.0
 * @return string
 */
function wc_get_user_agent() {
	return isset( $_SERVER['HTTP_USER_AGENT'] ) ? wc_clean( wp_unslash( $_SERVER['HTTP_USER_AGENT'] ) ) : ''; // @codingStandardsIgnoreLine
}

// This function can be removed when WP 3.9.2 or greater is required.
if ( ! function_exists( 'hash_equals' ) ) :
	/**
	 * Compare two strings in constant time.
	 *
	 * This function was added in PHP 5.6.
	 * It can leak the length of a string.
	 *
	 * @since 3.9.2
	 *
	 * @param string $a Expected string.
	 * @param string $b Actual string.
```

```
		 * @return bool Whether strings are equal.
		 */
		function hash_equals( $a, $b ) {
			$a_length = strlen( $a );
			if ( strlen( $b ) !== $a_length ) {
				return false;
			}
			$result = 0;

			// Do not attempt to "optimize" this.
			for ( $i = 0; $i < $a_length; $i++ ) {
				$result |= ord( $a[ $i ] ) ^ ord( $b[ $i ] );
			}

			return 0 === $result;
		}
	endif;

	/**
	 * Generate a rand hash.
	 *
	 * @since  2.4.0
	 * @return string
	 */
	function wc_rand_hash() {
```

```
	if ( ! function_exists( 'openssl_random_pseudo_bytes' ) ) {
		return sha1( wp_rand() );
	}

	return bin2hex( openssl_random_pseudo_bytes( 20 ) ); // @codingStandardsIgnoreLine
}

/**
 * WC API - Hash.
 *
 * @since  2.4.0
 * @param  string $data Message to be hashed.
 * @return string
 */
function wc_api_hash( $data ) {
	return hash_hmac( 'sha256', $data, 'wc-api' );
}

/**
 * Find all possible combinations of values from the input array and return in a logical order.
 *
 * @since 2.5.0
 * @param array $input Input.
 * @return array
 */
```

```
function wc_array_cartesian( $input ) {
	$input   = array_filter( $input );
	$results = array();
	$indexes = array();
	$index   = 0;

	// Generate indexes from keys and values so we have a logical sort order.
	foreach ( $input as $key => $values ) {
		foreach ( $values as $value ) {
			$indexes[ $key ][ $value ] = $index++;
		}
	}

	// Loop over the 2D array of indexes and generate all combinations.
	foreach ( $indexes as $key => $values ) {
		// When result is empty, fill with the values of the first looped array.
		if ( empty( $results ) ) {
			foreach ( $values as $value ) {
				$results[] = array( $key => $value );
			}
		} else {
			// Second and subsequent input sub-array merging.
			foreach ( $results as $result_key => $result ) {
				foreach ( $values as $value ) {
					// If the key is not set, we can set it.
```

```
                                        if ( ! isset( $results[ $result_key ][ $key ] ) ) {
                                                $results[ $result_key ][ $key ] = $value;
                                        } else {
                                                // If the key is set, we can add a new combination to the
results array.
                                                $new_combination        = $results[ $result_key ];
                                                $new_combination[ $key ] = $value;
                                                $results[]              = $new_combination;
                                        }
                                }
                        }
                }
        }

        // Sort the indexes.
        arsort( $results );

        // Convert indexes back to values.
        foreach ( $results as $result_key => $result ) {
                $converted_values = array();

                // Sort the values.
                arsort( $results[ $result_key ] );

                // Convert the values.
                foreach ( $results[ $result_key ] as $key => $value ) {
```

```
                        $converted_values[ $key ] = array_search( $value, $indexes[ $key ], true );
                }

                $results[ $result_key ] = $converted_values;
        }

        return $results;
}

/**
 * Run a MySQL transaction query, if supported.
 *
 * @since 2.5.0
 * @param string $type Types: start (default), commit, rollback.
 * @param bool   $force use of transactions.
 */
function wc_transaction_query( $type = 'start', $force = false ) {
        global $wpdb;

        $wpdb->hide_errors();

        wc_maybe_define_constant( 'WC_USE_TRANSACTIONS', true );

        if ( WC_USE_TRANSACTIONS || $force ) {
                switch ( $type ) {
```

```
                        case 'commit':
                                $wpdb->query( 'COMMIT' );
                                break;
                        case 'rollback':
                                $wpdb->query( 'ROLLBACK' );
                                break;
                        default:
                                $wpdb->query( 'START TRANSACTION' );
                                break;
                }
        }
}

/**
 * Gets the url to the cart page.
 *
 * @since  2.5.0
 *
 * @return string Url to cart page
 */
function wc_get_cart_url() {
        return apply_filters( 'woocommerce_get_cart_url', wc_get_page_permalink( 'cart' ) );
}

/**
```

```
 * Gets the url to the checkout page.
 *
 * @since  2.5.0
 *
 * @return string Url to checkout page
 */
function wc_get_checkout_url() {
	$checkout_url = wc_get_page_permalink( 'checkout' );
	if ( $checkout_url ) {
		// Force SSL if needed.
		if ( is_ssl() || 'yes' === get_option( 'woocommerce_force_ssl_checkout' ) ) {
			$checkout_url = str_replace( 'http:', 'https:', $checkout_url );
		}
	}

	return apply_filters( 'woocommerce_get_checkout_url', $checkout_url );
}

/**
 * Register a shipping method.
 *
 * @since 1.5.7
 * @param string|object $shipping_method class name (string) or a class object.
 */
function woocommerce_register_shipping_method( $shipping_method ) {
```

```
		WC()->shipping->register_shipping_method( $shipping_method );
}

if ( ! function_exists( 'wc_get_shipping_zone' ) ) {
		/**
		 * Get the shipping zone matching a given package from the cart.
		 *
		 * @since  2.6.0
		 * @uses   WC_Shipping_Zones::get_zone_matching_package
		 * @param  array $package Shipping package.
		 * @return WC_Shipping_Zone
		 */
		function wc_get_shipping_zone( $package ) {
			return WC_Shipping_Zones::get_zone_matching_package( $package );
		}
}

/**
 * Get a nice name for credit card providers.
 *
 * @since  2.6.0
 * @param  string $type Provider Slug/Type.
 * @return string
 */
function wc_get_credit_card_type_label( $type ) {
```

```
	// Normalize.
	$type = strtolower( $type );
	$type = str_replace( '-', ' ', $type );
	$type = str_replace( '_', ' ', $type );

	$labels = apply_filters(
		'woocommerce_credit_card_type_labels', array(
			'mastercard'      => __( 'MasterCard', 'woocommerce' ),
			'visa'            => __( 'Visa', 'woocommerce' ),
			'discover'        => __( 'Discover', 'woocommerce' ),
			'american express' => __( 'American Express', 'woocommerce' ),
			'diners'          => __( 'Diners', 'woocommerce' ),
			'jcb'             => __( 'JCB', 'woocommerce' ),
		)
	);

	return apply_filters( 'woocommerce_get_credit_card_type_label', ( array_key_exists( $type, $labels ) ? $labels[ $type ] : ucfirst( $type ) ) );
}

/**
 * Outputs a "back" link so admin screens can easily jump back a page.
 *
 * @param string $label Title of the page to return to.
 * @param string $url   URL of the page to return to.
 */
```

```
function wc_back_link( $label, $url ) {

	echo '<small class="wc-admin-breadcrumb"><a href="' . esc_url( $url ) . '" aria-label="' . esc_attr( $label ) . '">&#x2934;</a></small>';

}


/**
 * Display a WooCommerce help tip.
 *
 * @since  2.5.0
 *
 * @param  string $tip       Help tip text.
 * @param  bool   $allow_html Allow sanitized HTML if true or escape.
 * @return string
 */
function wc_help_tip( $tip, $allow_html = false ) {
	if ( $allow_html ) {
		$tip = wc_sanitize_tooltip( $tip );
	} else {
		$tip = esc_attr( $tip );
	}

	return '<span class="woocommerce-help-tip" data-tip="' . $tip . '"></span>';
}


/**
 * Return a list of potential postcodes for wildcard searching.
```

```
 *
 * @since 2.6.0
 * @param  string $postcode Postcode.
 * @param  string $country  Country to format postcode for matching.
 * @return string[]
 */
function wc_get_wildcard_postcodes( $postcode, $country = '' ) {
	$formatted_postcode = wc_format_postcode( $postcode, $country );
	$length             = function_exists( 'mb_strlen' ) ? mb_strlen( $formatted_postcode ) : strlen( $formatted_postcode );
	$postcodes          = array(
		$postcode,
		$formatted_postcode,
		$formatted_postcode . '*',
	);

	for ( $i = 0; $i < $length; $i ++ ) {
		$postcodes[] = ( function_exists( 'mb_substr' ) ? mb_substr( $formatted_postcode, 0, ( $i + 1 ) * -1 ) : substr( $formatted_postcode, 0, ( $i + 1 ) * -1 ) ) . '*';
	}

	return $postcodes;
}

/**
 * Used by shipping zones and taxes to compare a given $postcode to stored
```

```
 * postcodes to find matches for numerical ranges, and wildcards.
 *
 * @since 2.6.0
 * @param string $postcode          Postcode you want to match against stored postcodes.
 * @param array  $objects           Array of postcode objects from Database.
 * @param string $object_id_key     DB column name for the ID.
 * @param string $object_compare_key DB column name for the value.
 * @param string $country           Country from which this postcode belongs. Allows for formatting.
 * @return array Array of matching object ID and matching values.
 */
function wc_postcode_location_matcher( $postcode, $objects, $object_id_key, $object_compare_key, $country = '' ) {
	$postcode         = wc_normalize_postcode( $postcode );
	$wildcard_postcodes = array_map( 'wc_clean', wc_get_wildcard_postcodes( $postcode, $country ) );
	$matches          = array();

	foreach ( $objects as $object ) {
		$object_id      = $object->$object_id_key;
		$compare_against = $object->$object_compare_key;

		// Handle postcodes containing ranges.
		if ( strstr( $compare_against, '...' ) ) {
			$range = array_map( 'trim', explode( '...', $compare_against ) );

			if ( 2 !== count( $range ) ) {
```

```
                continue;
            }

            list( $min, $max ) = $range;

            // If the postcode is non-numeric, make it numeric.
            if ( ! is_numeric( $min ) || ! is_numeric( $max ) ) {
                $compare = wc_make_numeric_postcode( $postcode );
                $min     = str_pad( wc_make_numeric_postcode( $min ), strlen( $compare ), '0' );
                $max     = str_pad( wc_make_numeric_postcode( $max ), strlen( $compare ), '0' );
            } else {
                $compare = $postcode;
            }

            if ( $compare >= $min && $compare <= $max ) {
                $matches[ $object_id ]   = isset( $matches[ $object_id ] ) ? $matches[ $object_id ] : array();
                $matches[ $object_id ][] = $compare_against;
            }
        } elseif ( in_array( $compare_against, $wildcard_postcodes, true ) ) {
            // Wildcard and standard comparison.
            $matches[ $object_id ]   = isset( $matches[ $object_id ] ) ? $matches[ $object_id ] : array();
            $matches[ $object_id ][] = $compare_against;
        }
```

```
		}

		return $matches;
}

/**
 * Gets number of shipping methods currently enabled. Used to identify if
 * shipping is configured.
 *
 * @since  2.6.0
 * @param  bool $include_legacy Count legacy shipping methods too.
 * @return int
 */
function wc_get_shipping_method_count( $include_legacy = false ) {
		global $wpdb;

		$transient_name = 'wc_shipping_method_count_' . ( $include_legacy ? 1 : 0 ) . '_' .
WC_Cache_Helper::get_transient_version( 'shipping' );
		$method_count   = get_transient( $transient_name );

		if ( false === $method_count ) {
			$method_count = absint( $wpdb->get_var( "SELECT COUNT(*) FROM {$wpdb-
>prefix}woocommerce_shipping_zone_methods" ) );

			if ( $include_legacy ) {
				// Count activated methods that don't support shipping zones.
```

```
                        $methods = WC()->shipping->get_shipping_methods();


                        foreach ( $methods as $method ) {
                                if ( isset( $method->enabled ) && 'yes' === $method->enabled && !
$method->supports( 'shipping-zones' ) ) {
                                        $method_count++;
                                }
                        }
                }

                set_transient( $transient_name, $method_count, DAY_IN_SECONDS * 30 );
        }

        return absint( $method_count );
}

/**
 * Wrapper for set_time_limit to see if it is enabled.
 *
 * @since 2.6.0
 * @param int $limit Time limit.
 */
function wc_set_time_limit( $limit = 0 ) {
        if ( function_exists( 'set_time_limit' ) && false === strpos( ini_get( 'disable_functions' ),
'set_time_limit' ) && ! ini_get( 'safe_mode' ) ) { // phpcs:ignore
PHPCompatibility.PHP.DeprecatedIniDirectives.safe_modeDeprecatedRemoved
                @set_time_limit( $limit ); // @codingStandardsIgnoreLine
```

```
		}
}

/**
 * Wrapper for nocache_headers which also disables page caching.
 *
 * @since 3.2.4
 */
function wc_nocache_headers() {
		WC_Cache_Helper::set_nocache_constants();
		nocache_headers();
}

/**
 * Used to sort products attributes with uasort.
 *
 * @since 2.6.0
 * @param array $a First attribute to compare.
 * @param array $b Second attribute to compare.
 * @return int
 */
function wc_product_attribute_uasort_comparison( $a, $b ) {
		return wc_uasort_comparison( $a['position'], $b['position'] );
}
```

```
/**
 * Used to sort shipping zone methods with uasort.
 *
 * @since 3.0.0
 * @param array $a First shipping zone method to compare.
 * @param array $b Second shipping zone method to compare.
 * @return int
 */
function wc_shipping_zone_method_order_uasort_comparison( $a, $b ) {
	return wc_uasort_comparison( $a->method_order, $b->method_order );
}

/**
 * User to sort checkout fields based on priority with uasort.
 *
 * @since 3.5.1
 * @param array $a First field to compare.
 * @param array $b Second field to compare.
 * @return int
 */
function wc_checkout_fields_uasort_comparison( $a, $b ) {
	/*
	 * We are not guaranteed to get a priority
	 * setting. So don't compare if they don't
	 * exist.
```

```
		 */
		if ( ! isset( $a['priority'], $b['priority'] ) ) {
			return 0;
		}

		return wc_uasort_comparison( $a['priority'], $b['priority'] );
}

/**
 * User to sort two values with ausort.
 *
 * @since 3.5.1
 * @param int $a First value to compare.
 * @param int $b Second value to compare.
 * @return int
 */
function wc_uasort_comparison( $a, $b ) {
		if ( $a === $b ) {
			return 0;
		}
		return ( $a < $b ) ? -1 : 1;
}

/**
 * Get rounding mode for internal tax calculations.
```

```
 *
 * @since 3.2.4
 * @return int
 */
function wc_get_tax_rounding_mode() {
	$constant = WC_TAX_ROUNDING_MODE;

	if ( 'auto' === $constant ) {
		return 'yes' === get_option( 'woocommerce_prices_include_tax', 'no' ) ? 2 : 1;
	}

	return intval( $constant );
}

/**
 * Get rounding precision for internal WC calculations.
 * Will increase the precision of wc_get_price_decimals by 2 decimals, unless WC_ROUNDING_PRECISION is set to a higher number.
 *
 * @since 2.6.3
 * @return int
 */
function wc_get_rounding_precision() {
	$precision = wc_get_price_decimals() + 2;
	if ( absint( WC_ROUNDING_PRECISION ) > $precision ) {
		$precision = absint( WC_ROUNDING_PRECISION );
```

```
		}

		return $precision;

}


/**
 * Add precision to a number and return a number.
 *
 * @since  3.2.0
 * @param  float $value Number to add precision to.
 * @param  bool  $round If should round after adding precision.
 * @return int|float
 */
function wc_add_number_precision( $value, $round = true ) {
		$cent_precision = pow( 10, wc_get_price_decimals() );
		$value          = $value * $cent_precision;
		return $round ? round( $value, wc_get_rounding_precision() - wc_get_price_decimals() ) :
$value;
}


/**
 * Remove precision from a number and return a float.
 *
 * @since  3.2.0
 * @param  float $value Number to add precision to.
 * @return float
 */
```

```
function wc_remove_number_precision( $value ) {
	$cent_precision = pow( 10, wc_get_price_decimals() );
	return $value / $cent_precision;
}

/**
 * Add precision to an array of number and return an array of int.
 *
 * @since  3.2.0
 * @param  array $value Number to add precision to.
 * @param  bool  $round Should we round after adding precision?.
 * @return int|array
 */
function wc_add_number_precision_deep( $value, $round = true ) {
	if ( ! is_array( $value ) ) {
		return wc_add_number_precision( $value, $round );
	}

	foreach ( $value as $key => $sub_value ) {
		$value[ $key ] = wc_add_number_precision_deep( $sub_value, $round );
	}

	return $value;
}
```

```
/**
 * Remove precision from an array of number and return an array of int.
 *
 * @since  3.2.0
 * @param  array $value Number to add precision to.
 * @return int|array
 */
function wc_remove_number_precision_deep( $value ) {
	if ( ! is_array( $value ) ) {
		return wc_remove_number_precision( $value );
	}

	foreach ( $value as $key => $sub_value ) {
		$value[ $key ] = wc_remove_number_precision_deep( $sub_value );
	}

	return $value;
}

/**
 * Get a shared logger instance.
 *
 * Use the woocommerce_logging_class filter to change the logging class. You may provide one of the following:
 *     - a class name which will be instantiated as `new $class` with no arguments
 *     - an instance which will be used directly as the logger
```

```
 * In either case, the class or instance *must* implement WC_Logger_Interface.
 *
 * @see WC_Logger_Interface
 *
 * @return WC_Logger
 */
function wc_get_logger() {
	static $logger = null;

	$class = apply_filters( 'woocommerce_logging_class', 'WC_Logger' );

	if ( null !== $logger && is_string( $class ) && is_a( $logger, $class ) ) {
		return $logger;
	}

	$implements = class_implements( $class );

	if ( is_array( $implements ) && in_array( 'WC_Logger_Interface', $implements, true ) ) {
		$logger = is_object( $class ) ? $class : new $class();
	} else {
		wc_doing_it_wrong(
			__FUNCTION__,
			sprintf(
				/* translators: 1: class name 2: woocommerce_logging_class 3:
WC_Logger_Interface */
```

```
                                __( 'The class %1$s provided by %2$s filter must implement %3$s.',
'woocommerce' ),
                                '<code>' . esc_html( is_object( $class ) ? get_class( $class ) : $class ) .
'</code>',
                                '<code>woocommerce_logging_class</code>',
                                '<code>WC_Logger_Interface</code>'
                        ),
                        '3.0'
                );

                $logger = is_a( $logger, 'WC_Logger' ) ? $logger : new WC_Logger();
        }

        return $logger;
}

/**
 * Trigger logging cleanup using the logging class.
 *
 * @since 3.4.0
 */
function wc_cleanup_logs() {
        $logger = wc_get_logger();

        if ( is_callable( array( $logger, 'clear_expired_logs' ) ) ) {
                $logger->clear_expired_logs();
```

```
		}
}
add_action( 'woocommerce_cleanup_logs', 'wc_cleanup_logs' );


/**
 * Prints human-readable information about a variable.
 *
 * Some server environments blacklist some debugging functions. This function provides a safe way to
 * turn an expression into a printable, readable form without calling blacklisted functions.
 *
 * @since 3.0
 *
 * @param mixed $expression The expression to be printed.
 * @param bool  $return     Optional. Default false. Set to true to return the human-readable string.
 * @return string|bool False if expression could not be printed. True if the expression was printed.
 *     If $return is true, a string representation will be returned.
 */
function wc_print_r( $expression, $return = false ) {
		$alternatives = array(
			array(
				'func' => 'print_r',
				'args' => array( $expression, true ),
			),
			array(
				'func' => 'var_export',
```

```
                'args' => array( $expression, true ),
        ),
        array(
                'func' => 'json_encode',
                'args' => array( $expression ),
        ),
        array(
                'func' => 'serialize',
                'args' => array( $expression ),
        ),
);

$alternatives = apply_filters( 'woocommerce_print_r_alternatives', $alternatives, $expression );

foreach ( $alternatives as $alternative ) {
        if ( function_exists( $alternative['func'] ) ) {
                $res = call_user_func_array( $alternative['func'], $alternative['args'] );
                if ( $return ) {
                        return $res;
                }

                echo $res; // WPCS: XSS ok.
                return true;
        }
}
```

```
	return false;
}

/**
 * Registers the default log handler.
 *
 * @since 3.0
 * @param array $handlers Handlers.
 * @return array
 */
function wc_register_default_log_handler( $handlers ) {
	if ( defined( 'WC_LOG_HANDLER' ) && class_exists( WC_LOG_HANDLER ) ) {
		$handler_class   = WC_LOG_HANDLER;
		$default_handler = new $handler_class();
	} else {
		$default_handler = new WC_Log_Handler_File();
	}

	array_push( $handlers, $default_handler );

	return $handlers;
}
add_filter( 'woocommerce_register_log_handlers', 'wc_register_default_log_handler' );
```

```
/**
 * Based on wp_list_pluck, this calls a method instead of returning a property.
 *
 * @since 3.0.0
 * @param array      $list             List of objects or arrays.
 * @param int|string $callback_or_field Callback method from the object to place instead of the entire object.
 * @param int|string $index_key        Optional. Field from the object to use as keys for the new array.
 *                                     Default null.
 * @return array Array of values.
 */
function wc_list_pluck( $list, $callback_or_field, $index_key = null ) {
	// Use wp_list_pluck if this isn't a callback.
	$first_el = current( $list );
	if ( ! is_object( $first_el ) || ! is_callable( array( $first_el, $callback_or_field ) ) ) {
		return wp_list_pluck( $list, $callback_or_field, $index_key );
	}
	if ( ! $index_key ) {
		/*
		 * This is simple. Could at some point wrap array_column()
		 * if we knew we had an array of arrays.
		 */
		foreach ( $list as $key => $value ) {
			$list[ $key ] = $value->{$callback_or_field}();
		}
		return $list;
```

```
		}

		/*
		 * When index_key is not set for a particular item, push the value
		 * to the end of the stack. This is how array_column() behaves.
		 */
		$newlist = array();
		foreach ( $list as $value ) {
			// Get index. @since 3.2.0 this supports a callback.
			if ( is_callable( array( $value, $index_key ) ) ) {
				$newlist[ $value->{$index_key}() ] = $value->{$callback_or_field}();
			} elseif ( isset( $value->$index_key ) ) {
				$newlist[ $value->$index_key ] = $value->{$callback_or_field}();
			} else {
				$newlist[] = $value->{$callback_or_field}();
			}
		}
		return $newlist;
	}

	/**
	 * Get permalink settings for things like products and taxonomies.
	 *
	 * As of 3.3.0, the permalink settings are stored to the option instead of
	 * being blank and inheritting from the locale. This speeds up page loading
```

```
 * times by negating the need to switch locales on each page load.
 *
 * This is more inline with WP core behavior which does not localize slugs.
 *
 * @since  3.0.0
 * @return array
 */
function wc_get_permalink_structure() {
	$saved_permalinks = (array) get_option( 'woocommerce_permalinks', array() );
	$permalinks       = wp_parse_args(
		array_filter( $saved_permalinks ), array(
			'product_base'          => _x( 'product', 'slug', 'woocommerce' ),
			'category_base'         => _x( 'product-category', 'slug', 'woocommerce' ),
			'tag_base'              => _x( 'product-tag', 'slug', 'woocommerce' ),
			'attribute_base'        => '',
			'use_verbose_page_rules' => false,
		)
	);

	if ( $saved_permalinks !== $permalinks ) {
		update_option( 'woocommerce_permalinks', $permalinks );
	}

	$permalinks['product_rewrite_slug']   = untrailingslashit( $permalinks['product_base'] );
	$permalinks['category_rewrite_slug']  = untrailingslashit( $permalinks['category_base'] );
```

```
	$permalinks['tag_rewrite_slug']       = untrailingslashit( $permalinks['tag_base'] );
	$permalinks['attribute_rewrite_slug'] = untrailingslashit( $permalinks['attribute_base'] );

	return $permalinks;
}

/**
 * Switch WooCommerce to site language.
 *
 * @since 3.1.0
 */
function wc_switch_to_site_locale() {
	if ( function_exists( 'switch_to_locale' ) ) {
		switch_to_locale( get_locale() );

		// Filter on plugin_locale so load_plugin_textdomain loads the correct locale.
		add_filter( 'plugin_locale', 'get_locale' );

		// Init WC locale.
		WC()->load_plugin_textdomain();
	}
}

/**
 * Switch WooCommerce language to original.
```

```
 *
 * @since 3.1.0
 */
function wc_restore_locale() {
	if ( function_exists( 'restore_previous_locale' ) ) {
		restore_previous_locale();

		// Remove filter.
		remove_filter( 'plugin_locale', 'get_locale' );

		// Init WC locale.
		WC()->load_plugin_textdomain();
	}
}

/**
 * Convert plaintext phone number to clickable phone number.
 *
 * Remove formatting and allow "+".
 * Example and specs:
https://developer.mozilla.org/en/docs/Web/HTML/Element/a#Creating_a_phone_link
 *
 * @since 3.1.0
 *
 * @param string $phone Content to convert phone number.
 * @return string Content with converted phone number.
```

```
 */
function wc_make_phone_clickable( $phone ) {
	$number = trim( preg_replace( '/[^\d|\+]/', '', $phone ) );

	return $number ? '<a href="tel:' . esc_attr( $number ) . '">' . esc_html( $phone ) . '</a>' : '';
}

/**
 * Get an item of post data if set, otherwise return a default value.
 *
 * @since  3.0.9
 * @param  string $key     Meta key.
 * @param  string $default Default value.
 * @return mixed Value sanitized by wc_clean.
 */
function wc_get_post_data_by_key( $key, $default = '' ) {
	return wc_clean( wp_unslash( wc_get_var( $_POST[ $key ], $default ) ) ); // @codingStandardsIgnoreLine
}

/**
 * Get data if set, otherwise return a default value or null. Prevents notices when data is not set.
 *
 * @since  3.2.0
 * @param  mixed  $var     Variable.
 * @param  string $default Default value.
```

```
 * @return mixed
 */
function wc_get_var( &$var, $default = null ) {
	return isset( $var ) ? $var : $default;
}

/**
 * Read in WooCommerce headers when reading plugin headers.
 *
 * @since 3.2.0
 * @param array $headers Headers.
 * @return array
 */
function wc_enable_wc_plugin_headers( $headers ) {
	if ( ! class_exists( 'WC_Plugin_Updates' ) ) {
		include_once dirname( __FILE__ ) . '/admin/plugin-updates/class-wc-plugin-updates.php';
	}

	$headers['WCRequires'] = WC_Plugin_Updates::VERSION_REQUIRED_HEADER;
	$headers['WCTested']   = WC_Plugin_Updates::VERSION_TESTED_HEADER;
	return $headers;
}
add_filter( 'extra_plugin_headers', 'wc_enable_wc_plugin_headers' );

/**
```

```
 * Prevent auto-updating the WooCommerce plugin on major releases if there are untested extensions active.
 *
 * @since 3.2.0
 * @param  bool   $should_update If should update.
 * @param  object $plugin        Plugin data.
 * @return bool
 */
function wc_prevent_dangerous_auto_updates( $should_update, $plugin ) {
	if ( ! isset( $plugin->plugin, $plugin->new_version ) ) {
		return $should_update;
	}

	if ( 'woocommerce/woocommerce.php' !== $plugin->plugin ) {
		return $should_update;
	}

	if ( ! class_exists( 'WC_Plugin_Updates' ) ) {
		include_once dirname( __FILE__ ) . '/admin/plugin-updates/class-wc-plugin-updates.php';
	}

	$new_version      = wc_clean( $plugin->new_version );
	$plugin_updates   = new WC_Plugin_Updates();
	$untested_plugins = $plugin_updates->get_untested_plugins( $new_version, 'major' );
	if ( ! empty( $untested_plugins ) ) {
```

```
                return false;
        }

        return $should_update;
}
add_filter( 'auto_update_plugin', 'wc_prevent_dangerous_auto_updates', 99, 2 );

/**
 * Delete expired transients.
 *
 * Deletes all expired transients. The multi-table delete syntax is used.
 * to delete the transient record from table a, and the corresponding.
 * transient_timeout record from table b.
 *
 * Based on code inside core's upgrade_network() function.
 *
 * @since 3.2.0
 * @return int Number of transients that were cleared.
 */
function wc_delete_expired_transients() {
        global $wpdb;

        $sql  = "DELETE a, b FROM $wpdb->options a, $wpdb->options b
                WHERE a.option_name LIKE %s
                AND a.option_name NOT LIKE %s
```

```
                AND b.option_name = CONCAT( '_transient_timeout_', SUBSTRING( a.option_name, 12 )
)

                AND b.option_value < %d";

        $rows = $wpdb->query( $wpdb->prepare( $sql, $wpdb->esc_like( '_transient_' ) . '%', $wpdb-
>esc_like( '_transient_timeout_' ) . '%', time() ) ); // WPCS: unprepared SQL ok.



        $sql   = "DELETE a, b FROM $wpdb->options a, $wpdb->options b

                WHERE a.option_name LIKE %s

                AND a.option_name NOT LIKE %s

                AND b.option_name = CONCAT( '_site_transient_timeout_', SUBSTRING(
a.option_name, 17 ) )

                AND b.option_value < %d";

        $rows2 = $wpdb->query( $wpdb->prepare( $sql, $wpdb->esc_like( '_site_transient_' ) . '%',
$wpdb->esc_like( '_site_transient_timeout_' ) . '%', time() ) ); // WPCS: unprepared SQL ok.



        return absint( $rows + $rows2 );

}

add_action( 'woocommerce_installed', 'wc_delete_expired_transients' );



/**

 * Make a URL relative, if possible.

 *

 * @since 3.2.0

 * @param string $url URL to make relative.

 * @return string

 */

function wc_get_relative_url( $url ) {
```

```
	return wc_is_external_resource( $url ) ? $url : str_replace( array( 'http://', 'https://' ), '//', $url );
}

/**
 * See if a resource is remote.
 *
 * @since 3.2.0
 * @param string $url URL to check.
 * @return bool
 */
function wc_is_external_resource( $url ) {
	$wp_base = str_replace( array( 'http://', 'https://' ), '//', get_home_url( null, '/', 'http' ) );

	return strstr( $url, '://' ) && ! strstr( $url, $wp_base );
}

/**
 * See if theme/s is activate or not.
 *
 * @since 3.3.0
 * @param string|array $theme Theme name or array of theme names to check.
 * @return boolean
 */
function wc_is_active_theme( $theme ) {
	return is_array( $theme ) ? in_array( get_template(), $theme, true ) : get_template() === $theme;
```

```
}

/**
 * Cleans up session data - cron callback.
 *
 * @since 3.3.0
 */
function wc_cleanup_session_data() {
	$session_class = apply_filters( 'woocommerce_session_handler', 'WC_Session_Handler' );
	$session       = new $session_class();

	if ( is_callable( array( $session, 'cleanup_sessions' ) ) ) {
		$session->cleanup_sessions();
	}
}
add_action( 'woocommerce_cleanup_sessions', 'wc_cleanup_session_data' );

/**
 * Convert a decimal (e.g. 3.5) to a fraction (e.g. 7/2).
 * From: https://www.designedbyaturtle.co.uk/2015/converting-a-decimal-to-a-fraction-in-php/
 *
 * @param float $decimal the decimal number.
 * @return array|bool a 1/2 would be [1, 2] array (this can be imploded with '/' to form a string).
 */
function wc_decimal_to_fraction( $decimal ) {
```

```
	if ( 0 > $decimal || ! is_numeric( $decimal ) ) {
		// Negative digits need to be passed in as positive numbers and prefixed as negative once the response is imploded.
		return false;
	}

	if ( 0 === $decimal ) {
		return array( 0, 1 );
	}

	$tolerance   = 1.e-4;
	$numerator   = 1;
	$h2          = 0;
	$denominator = 0;
	$k2          = 1;
	$b           = 1 / $decimal;

	do {
		$b           = 1 / $b;
		$a           = floor( $b );
		$aux         = $numerator;
		$numerator   = $a * $numerator + $h2;
		$h2          = $aux;
		$aux         = $denominator;
		$denominator = $a * $denominator + $k2;
		$k2          = $aux;
```

```
                $b          = $b - $a;
        } while ( abs( $decimal - $numerator / $denominator ) > $decimal * $tolerance );

        return array( $numerator, $denominator );
}

/**
 * Round discount.
 *
 * @param  double $value Amount to round.
 * @param  int    $precision DP to round.
 * @return float
 */
function wc_round_discount( $value, $precision ) {
        if ( version_compare( PHP_VERSION, '5.3.0', '>=' ) ) {
                return round( $value, $precision, WC_DISCOUNT_ROUNDING_MODE ); // phpcs:ignore PHPCompatibility.PHP.NewFunctionParameters.round_modeFound
        }

        if ( 2 === WC_DISCOUNT_ROUNDING_MODE ) {
                return wc_legacy_round_half_down( $value, $precision );
        }

        return round( $value, $precision );
}
```

```
/**
 * Return the html selected attribute if stringified $value is found in array of stringified $options
 * or if stringified $value is the same as scalar stringified $options.
 *
 * @param string|int       $value   Value to find within options.
 * @param string|int|array $options Options to go through when looking for value.
 * @return string
 */
function wc_selected( $value, $options ) {
	if ( is_array( $options ) ) {
		$options = array_map( 'strval', $options );
		return selected( in_array( (string) $value, $options, true ), true, false );
	}

	return selected( $value, $options, false );
}

/**
 * Retrieves the MySQL server version. Based on $wpdb.
 *
 * @since 3.4.1
 * @return array Vesion information.
 */
function wc_get_server_database_version() {
	global $wpdb;
```

```
	if ( empty( $wpdb->is_mysql ) ) {
		return array(
			'string' => '',
			'number' => '',
		);
	}

	if ( $wpdb->use_mysqli ) {
		$server_info = mysqli_get_server_info( $wpdb->dbh ); // @codingStandardsIgnoreLine.
	} else {
		$server_info = mysql_get_server_info( $wpdb->dbh ); // @codingStandardsIgnoreLine.
	}

	return array(
		'string' => $server_info,
		'number' => preg_replace( '/([^\d.]+).*/', '', $server_info ),
	);
}
```