How to incorporate gift wrapping seamlessly using checkout extensibility, utilizing the Checkout UI component and Cart Transform API.

TABLE OF CONTENTS:

Shopify places a strong emphasis on the significance of a smooth and efficient checkout process for converting potential customers. The platform has invested substantial efforts to ensure that the Shopify checkout is not only exceptionally fast but also user-friendly.

By understanding the critical role that the checkout experience plays in shaping customer perceptions and adhering to regulatory requirements, Shopify recognizes the need for flexibility. Brands are empowered to create unique and compelling checkout experiences that align with marketing strategies, all while ensuring order accuracy and compliance with merchant policies and regulations.

The integration of checkout extensibility serves as a key solution to achieving this delicate balance. Businesses utilizing Shopify Plus gain access to tools, functionalities, and deployment models that enable the creation of customized checkouts tailored to the unique needs of each brand. Exploring various technologies, this approach allows for the development of bespoke checkout solutions, unlocking opportunities for enhanced conversion rates, improved lifetime customer value, and increased average order value. The article delves into specific technologies and explores key use cases to showcase the full potential of these customizations.

A brief overview of the latest Shopify Winter ’23 Edition

Discover the versatility of checkout extensibility, a comprehensive suite of robust capabilities empowering developers to tailor the checkout experience. This suite encompasses Checkout UI extensions, Web pixel extensions, Shopify functions, and the Checkout branding API.

In the latest Shopify Winter ’23 Edition, exciting announcements were made regarding enhanced checkout UI extension capabilities and expanded surface areas, bringing substantial advantages for both developers and merchants.

For developers, the opportunity to directly sell checkout apps to Shopify Plus merchants through the Shopify App Store has been introduced. This facilitates broad distribution, enabling easy installation and configuration. These app-powered checkout extensions not only ensure upgradability and integration with Shop Pay but also boast higher conversion rates.

In-house developers and agencies gain equal access to these platform capabilities, empowering them to craft custom apps tailored to specific merchant needs.

For Shopify Plus admins, the installation and configuration process is streamlined through the new checkout editor. This app-centric model enables admins to swiftly customize their checkout interface without delving into code, offering a seamless and efficient solution.

Understanding Shopify Checkout:

At present, Shopify empowers merchants to establish their checkout processes, offering a high degree of customization to align with their specific requirements. Merchants can tailor various aspects, such as providing diverse payment options, incorporating discount and promo codes, creating custom fields for customer information, and even showcasing product recommendations to bolster sales. However, seasoned users of Shopify are well aware of the limitations imposed on the extent of customization within the checkout process. These constraints often become apparent in the course of navigating the platform and can present challenges as users discover the boundaries of what can and cannot be modified.

Customizing Checkout in Shopify:

Recognizing the pivotal role of delivering an exceptional shopping experience, customizing the checkout process becomes a crucial aspect of enhancing user satisfaction. Whether streamlining fields for quicker checkout, introducing new fields to elevate customer service, adjusting payment options, or implementing various modifications to foster trust and loyalty, there are a myriad of avenues to tailor the checkout process to suit your business's unique requirements. This exploration delves into diverse customization options, offering insights into how you can fine-tune your Shopify checkout page to achieve optimal perfection.

Traditional Approach to Customizing Shopify Checkout:

Historically, for brands operating on Shopify Plus, the avenue to customize Shopify's checkout process was through the checkout.liquid theme file. This approach provided direct access to the code, enabling brands to meticulously tailor the checkout experience to align with their unique branding.

This high level of customization empowered merchants to modify everything from page layout to payment and shipping options, offering endless possibilities. Businesses could craft a distinctive and memorable checkout experience that resonated with their customers, fostering brand identity.

Regrettably, this method of customization was exclusive to Shopify Plus brands, leaving smaller businesses and regular Shopify plan merchants feeling constrained and dissatisfied with their inability to create a truly personalized checkout experience.

Current Challenges in Checkout Customization on Shopify:

The current approach to checkout customization on Shopify presents challenges for merchants, impacting their ability to optimize the checkout process.

1. Limited Customization Options: Shopify's current customization options for the checkout process are relatively basic. While color and font changes are feasible, the layout of the page remains fixed. This limitation provides structure but hinders extensive design modifications.

2. Inability to Add New Functionality: Offering custom payment or shipping options becomes challenging without resorting to intricate workarounds or hiring a Shopify developer. This limitation proves to be a significant obstacle for businesses requiring more than standard payment and shipping choices.

3. Lack of Flexibility: Despite Shopify providing templates and layouts for the checkout page, merchants face constraints in extensive modifications. The limited options make it challenging to ensure a seamless checkout experience aligned with the overall website's branding and design.

4. Inconsistency with the Site: The checkout page's limited customization options can result in inconsistency with the rest of the site. This lack of flexibility in branding and design customization may confuse and frustrate customers, potentially leading to cart abandonment.

Revolutionizing Shopify Checkout with Checkout Extensibility

Shopify merchants now have cause for celebration, thanks to the introduction of Checkout Extensibility APIs. This marks a departure from the traditional method of utilizing Checkout Liquid, which is set to phase out for in-checkout pages after August 13, 2024.

So, what makes Checkout Extensibility so remarkable? For business owners valuing flexibility and control, Checkout Extensibility is a game-changer. This robust API empowers merchants to shape their checkout process to suit their needs, ushering in a new era of customization.

How Checkout Extensions Transform the Checkout Experience

Imagine your customers navigating your online store, adding items to their cart, and reaching the pivotal moment of checkout. This moment, the tipping point in the customer journey, is where Checkout Extensions come into play, giving you the ability to create an exceptional checkout experience.

Payment extensions enable the integration of third-party payment providers, expanding payment method options for customers. Meanwhile, shipping extensions allow the addition of new shipping providers, offering a broader range of shipping choices.

Yet, the capabilities extend further. With checkout UI extensions, you can completely revamp the visual identity of your checkout page, incorporating custom branding elements to create a distinctive and standout checkout experience.

Benefits of Embracing Checkout Extensions

The advantages of adopting checkout extensions are manifold, influencing the customer experience and conversion rates significantly.

1. Increased Flexibility: Merchants gain the ability to swiftly adjust their checkout process, fostering greater control and flexibility. This adaptability contributes to customer loyalty and drives sales.

2. Improved Customer Experience: Streamlining and enhancing the checkout process reduces cart abandonment rates, increasing the likelihood of conversions. An optimized checkout experience builds a strong brand reputation and a loyal customer base.

3. Increased Conversions and Sales: Custom payment and shipping options tailored to customer needs can be offered, enhancing the convenience of the checkout experience and positively impacting conversions and revenue.

4. Enhanced Functionality: Checkout Extensions provide a range of enhanced functionalities, allowing merchants to create a more personalized and differentiated checkout process. These APIs enable the addition of custom options and integration with third-party services.

5. Integration with Shopify Features: Checkout Extensions seamlessly integrate with other Shopify features, including Shopify Payments and Shopify Shipping, streamlining the management of the entire checkout process from a centralized location.

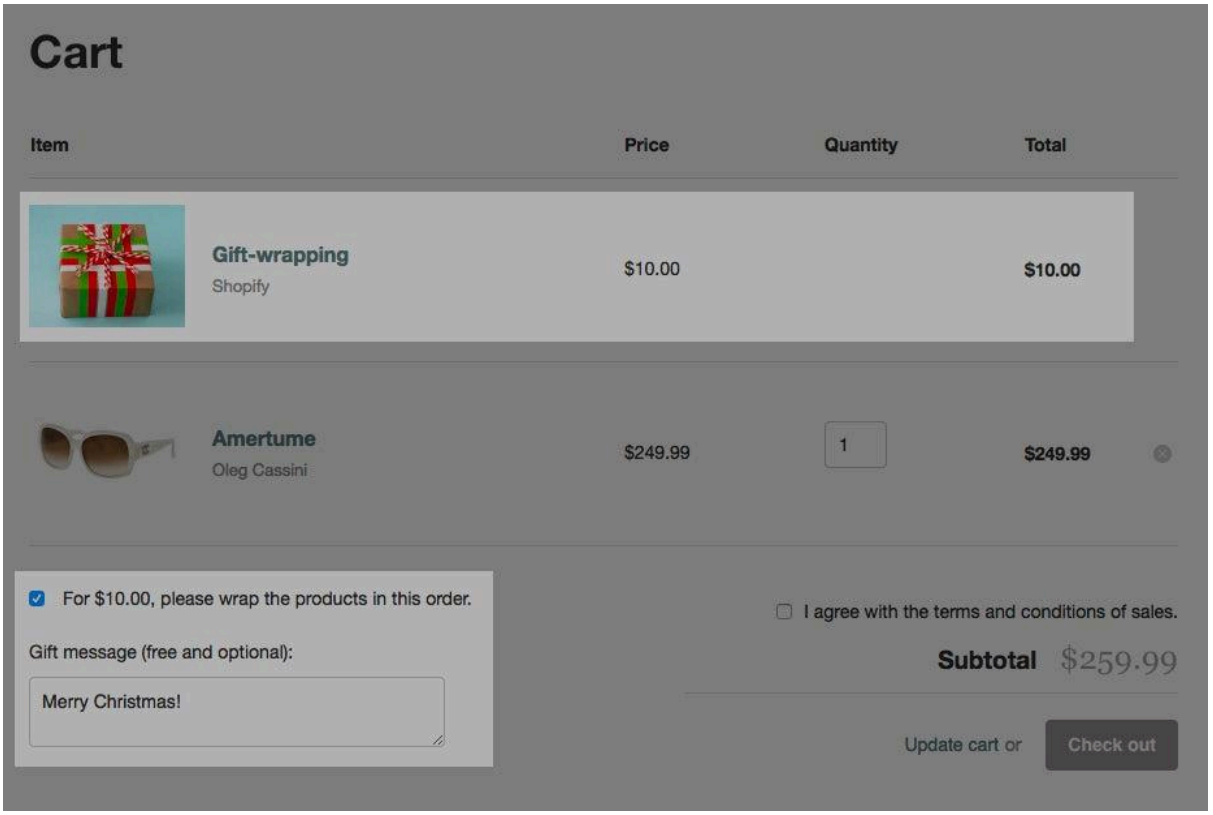
Examples of Checkout Extensions in Action

Merchants can leverage checkout extensions to:

- Add custom payment methods or shipping options aligned with specific business needs and brand values.
- Introduce specific shipping options, such as same-day delivery, not available by default in Shopify.
- Enhance the checkout process by adding custom payment methods or shipping options, potentially boosting conversions.
- Incorporate order notes or gift-wrapping options for customers to personalize their orders.
- Include additional fields on the checkout page to collect essential information, enhancing the overall efficiency of order fulfillment.
- Integrate with third-party services, like fraud detection or tax calculation tools, to bolster security and accuracy in the checkout process.

How to add a gift wrapping option to your cart page.

Provide your customers with the option of availing of a gift-wrapping service directly on the cart page of your online store. For those desiring their orders to be wrapped, you have the flexibility to either apply a flat rate or charge on a per-product basis.



Creating a Gift-Wrap Option: Step-by-Step Guide

Follow these steps to add a gift-wrap option to your cart page:

1. Create a Gift-Wrap Product:

- Navigate to Products in your Shopify admin.
- Click on "Add product."

Now, create a gift-wrap product as you would for any other item:

- Provide a detailed product description, outlining the materials used for gift wrapping.
- Assign a price for the gift-wrap service; set it to 0 if you want to offer free gift wrapping.
- Optionally, upload an image displaying how a gift-wrapped order will appear.
- Ensure the gift-wrap product has inventory, or adjust settings if inventory tracking isn't necessary, especially if your store has multiple locations.
- Click Save."

2. Create a menu:

Proceed to create a menu pointing to your gift-wrap product:

- Go to Online Store > Navigation in your Shopify admin.
- Click "Add menu."
- Name the menu "Gift Wrapping" to assign the handle "gift wrapping."
- Add the gift-wrap product to the menu:
- Click "Add menu item" and enter a name for the link to the gift-wrap product.
- In the Link field, select "Products," then choose the gift-wrap product from the drop-down menu.
- Click "Add."
- Click "Save menu."

3. Create a code snippet:

- Navigate to Online Store > Themes in your Shopify admin.
- Locate the theme you want to edit, click the three dots (...) to open the actions menu, and select "Edit code."
- In the Snippets directory, click "Add a new snippet."
- Name your snippet "gift-wrapping" and click "Create snippet." Your snippet file will open in the code editor.

Now, paste the relevant code into your new gift-wrapping snippet file based on how you plan to charge customers for the gift-wrapping service.

Note: The specific code will depend on your chosen method of charging for the gift-wrapping service.

Repeat these steps to implement the desired gift-wrap functionality seamlessly into your Shopify cart page.

Adding a Flat Rate Charge for Gift Wrapping:

To implement a flat-rate charge for gift wrapping, follow these steps and paste the provided code:

Paste the following code and save:

```
{% if link lists.gift-wrapping.links.size > 0 and
linklists.gift-wrapping.links.first.type == 'product_link' %}

<div
  id="is-a-gift"
```

```
  style="clear: left; margin: 30px 0"
  class="clearfix rte"
>
  <p>
        <input
        id="gift-wrapping"
        type="checkbox"
        name="attributes[gift-wrapping]"
        value="yes"
        {% if cart.attributes.gift-wrapping %}
        checked="checked"
        {% endif %}
        style="float: none"
        />
        <label
        for="gift-wrapping"
        style="display:inline; padding-left: 5px; float: none;"
        >
        For {{ linklists.gift-wrapping.links.first.object.price | money }}
        please wrap the products in this order.
        </label>
  </p>
  <p>
        <label style="display:block" for="gift-note"
        >Gift message (free and optional):</label
        >
        <textarea name="attributes[gift-note]" id="gift-note">
{{ cart.attributes.gift-note }}</textarea
        >
  </p>
</div>

{% assign id = linklists.gift-wrapping.links.first.object.variants.first.id
%} {% assign gift_wraps_in_cart = 0 %} {% for an item in the cart.items %} {% of
item.id == id %} {% assign gift_wraps_in_cart = item.quantity %} {% endif %}
{% endfor %}

<style>
  #updates_{{ id }} { display: none; }
</style>

<script>

  Shopify.Cart = Shopify.Cart || {};
```

```
Shopify.Cart.GiftWrap = {};

Shopify.Cart.GiftWrap.set = function() {
        var headers = new Headers({ 'Content-Type': 'application/json' });

        var request = {
        method: 'POST',
        headers: headers,
        body: JSON.stringify({ updates: { {{ id }}: 1 }, attributes: { 'gift-wrapping': true } })
        };
        fetch('/cart/update.js', request)
        .then(function() {
        location.href = '/cart';
        });
}

Shopify.Cart.GiftWrap.remove = function() {
        var headers = new Headers({ 'Content-Type': 'application/json' });

        var request = {
        method: 'POST',
        headers: headers,
        body: JSON.stringify({ updates: { {{ id }}: 0 }, attributes: { 'gift-wrapping': '',
'gift-note': '' } })
        };
        fetch('/cart/update.js', request)
        .then(function() {
        location.href = '/cart';
        });
}

// If we have nothing but gift-wrap items in the cart,.
{% if cart.items.size == 1 and gift_wraps_in_cart > 0 %}
document.addEventListener("DOMContentLoaded", function(){
        Shopify.Cart.GiftWrap.remove();
});
// If we have more than one gift-wrap item in the cart,.
{% elsif gift_wraps_in_cart > 1 %}
document.addEventListener("DOMContentLoaded", function(){
        Shopify.Cart.GiftWrap.set();
});
// If we have a gift-wrapped item in the cart but our gift-wrapping cart attribute has not been
set.
```

```
  {% elsif gift_wraps_in_cart > 0 and cart.attributes.gift-wrapping == blank  %}
  document.addEventListener("DOMContentLoaded", function(){
        Shopify.Cart.GiftWrap.set();
  });
  // If we have no gift-wrap item in the cart but our gift-wrapping cart attribute has been set.
  {% elsif gift_wraps_in_cart == 0 and cart.attributes.gift-wrapping != blank  %}
  document.addEventListener("DOMContentLoaded", function(){
        Shopify.Cart.GiftWrap.set();
  });
  {% endif %}

  // When the gift-wrapping checkbox is checked or unchecked.
  document.addEventListener("DOMContentLoaded", function(){

document.querySelector('[name="attributes[gift-wrapping]"]').addEventListener("change",
function(event) {
        if (event.target.checked) {
        Shopify.Cart.GiftWrap.set();
        } else {
        Shopify.Cart.GiftWrap.remove();
        }

        });

        document.querySelector('#gift-note').addEventListener("change", function(evt) {
        var note = evt.target.value;
        var headers = new Headers({ 'Content-Type': 'application/json' });

        var request = {
        method: 'POST',
        headers: headers,
        body: JSON.stringify({ attributes: { 'gift-note': note } })
        };

        fetch('/cart/update.js', request);
        });
  });
</script>

{% else %}

<p style="clear: left; margin: 30px 0" class="rte">
  You attempted to add a gift-wrapping script to your shopping cart, but it
  won't work because you don't have a link list with the handle
```

```
  <code>gift-wrapping</code> which, in turn, contains a link to your
  gift-wrapping product. Please review the steps outlined
  <a

href="https://help.shopify.com/manual/online-store/themes/os/customize/add-gift-wrap-optio
n"
        target="_blank"
        rel="noopener no-referrer nofollow"
        >here</a
  >.
</p>

{% endif %}
```

Feel free to paste this code into your Shopify theme to seamlessly integrate a flat-rate charge for gift wrapping into your cart page.

Implementing Gift Wrap Charge Multiplied by Product Quantity:

To add a charge multiplied by the number of products in the order, follow these steps and paste the provided code:

Paste the following code and save:

```
{% if linklists.gift-wrapping.links.size > 0 and
linklists.gift-wrapping.links.first.type == 'product_link' %}

<div
  id="is-a-gift"
  style="clear: left; margin: 30px 0"
  class="clearfix rte"
>
  <p>
        <input
        id="gift-wrapping"
        type="checkbox"
        name="attributes[gift-wrapping]"
        value="yes"
        {% if cart.attributes.gift-wrapping %}
        checked="checked"
        {% endif %}
        style="float: none"
        />
        <label
        for="gift-wrapping"
```

```
        style="display:inline; padding-left: 5px; float: none;"
        >
        For {{ linklists.gift-wrapping.links.first.object.price | money }} per
        item, please wrap the products in this order.
        </label>
  </p>
  <p>
        <label style="display:block" for="gift-note"
        >Gift message (free and optional):</label
        >
        <textarea name="attributes[gift-note]" id="gift-note">
{{ cart.attributes.gift-note }}</textarea
        >
  </p>
</div>

{% assign id = linklists.gift-wrapping.links.first.object.variants.first.id
%} {% assign gift_wraps_in_cart = 0 %} {% for an item in cart.items %} {% of
item.id == id %} {% assign gift_wraps_in_cart = item.quantity %} {% endif %}
{% endfor %} {% assign items_in_cart = cart.item_count | minus:
gift_wraps_in_cart %}

<style>
  #updates_{{ id }} { display: none; }
</style>

<script>

  Shopify.Cart = Shopify.Cart || {};

  Shopify.Cart.GiftWrap = {};

  Shopify.Cart.GiftWrap.set = function() {
        var headers = new Headers({ 'Content-Type': 'application/json' });

        var request = {
        method: 'POST',
        headers: headers,
        body: JSON.stringify({ updates: { {{ id }}: {{ items_in_cart }} }, attributes: {
'gift-wrapping': true } })
        };
        fetch('/cart/update.js', request)
        .then(function() {
        location.href = '/cart';
```

```
        });
  }

  Shopify.Cart.GiftWrap.remove = function() {
        var headers = new Headers({ 'Content-Type': 'application/json' });

        var request = {
        method: 'POST',
        headers: headers,
        body: JSON.stringify({ updates: { {{ id }}: 0 }, attributes: { 'gift-wrapping': '',
'gift-note': '' } })
        };
        fetch('/cart/update.js', request)
        .then(function() {
        location.href = '/cart';
        });
  }

  // If we have nothing but gift-wrap items in the cart.
  {% if cart.items.size == 1 and gift_wraps_in_cart > 0 %}
  document.addEventListener("DOMContentLoaded", function(){
        Shopify.Cart.GiftWrap.remove();
  });
  // If we don't have the right amount of gift-wrap items in the cart.
  {% elsif gift_wraps_in_cart > 0 and gift_wraps_in_cart != items_in_cart %}
  document.addEventListener("DOMContentLoaded", function(){
        Shopify.Cart.GiftWrap.set();
  });
  // If we have a gift-wrapped item in the cart but our gift-wrapping cart attribute has not been
set.
  {% elsif gift_wraps_in_cart > 0 and cart.attributes.gift-wrapping == blank  %}
  document.addEventListener("DOMContentLoaded", function(){
        Shopify.Cart.GiftWrap.set();
  });
  // If we have no gift-wrap item in the cart but our gift-wrapping cart attribute has been set.
  {% elsif gift_wraps_in_cart == 0 and cart.attributes.gift-wrapping != blank  %}
  document.addEventListener("DOMContentLoaded", function(){
        Shopify.Cart.GiftWrap.set();
  });
  {% endif %}

  // When the gift-wrapping checkbox is checked or unchecked.
  document.addEventListener("DOMContentLoaded", function(){
```

```
document.querySelector('[name="attributes[gift-wrapping]"]').addEventListener("change",
function(event) {
        if (event.target.checked) {
        Shopify.Cart.GiftWrap.set();
        } else {
        Shopify.Cart.GiftWrap.remove();
        }

        });

        document.querySelector('#gift-note').addEventListener("change", function(evt) {
        var note = evt.target.value;
        var headers = new Headers({ 'Content-Type': 'application/json' });

        var request = {
        method: 'POST',
        headers: headers,
        body: JSON.stringify({ attributes: { 'gift-note': note } })
        };

        fetch('/cart/update.js', request);
        });
  });
</script>

{% else %}

<p style="clear: left; margin: 30px 0" class="rte">
  You attempted to add a gift-wrapping script to your shopping cart, but it
  won't work because you don't have a link list with a handle
  <code>gift-wrapping</code> which, in turn, contains a link to your
  gift-wrapping product. Please review the steps outlined
  <a

href="https://help.shopify.com/manual/online-store/themes/os/customize/add-gift-wrap-optio
n"
        target="_blank"
        rel="noopener no-referrer nofollow"
        >here</a
  >.
</p>

{% endif %}
```

Paste this code into your Shopify theme to enable a charge multiplied by the number of products in the order for the gift-wrapping option.

Include the Gift-Wrapping Snippet in Your Cart Template:

To include the gift-wrapping snippet in your cart template, follow these steps:

1. Navigate to the Cart Template:

- In the Shopify admin, go to Online Store > Themes.
- Find the theme you want to edit, click the... button to open the actions menu, and then click Edit Code.
- In the Templates directory, look for cart-template.liquid. If your theme doesn't have a cart-template.liquid, then click cart.liquid.

2. Locate the closing </form> Tag:

- Within the cart-template.liquid or cart.liquid file, find the closing </form> tag in the code.

3. Insert the Snippet Code:

- On a new line above the closing </form> tag, paste the following code:
  liquid

{% render 'gift-wrapping' %}

4. Save Your Changes:

- Click Save to apply the changes to your cart template.

By following these steps, you've successfully included the gift-wrapping snippet in your cart template. This ensures that the gift-wrapping functionality is integrated into your Shopify store's cart page.

Bottom Line

As the proprietor of an e-commerce venture, you recognize the paramount importance of ensuring a seamless and enjoyable checkout process for your customers. Harnessing the power of Shopify's APIs, diverse tools, and robust community support allows you to tailor your checkout journey, aligning it precisely with your unique business requirements. Picture the capability to integrate custom fields, fashion bespoke checkout pages, and genuinely sculpt the checkout process to reflect your brand—it's a transformative endeavor.

Why settle for a one-size-fits-all checkout routine when you have the opportunity to customize it extensively, infusing your brand's identity into every transaction? With Shopify, you're not confined to a cookie-cutter approach; instead, you can unlock the full potential of your business by creating a checkout experience that resonates with your customers on a personal level.

Contact Us

Should queries arise regarding the intricacies of checkout extensibility, don't hesitate to engage with our software development company. Feel free to seek guidance or pose inquiries, as we are here to assist you on your journey toward an optimized and uniquely tailored checkout process.