Migrating from Shopify's Old Checkout to Checkout Extensibility: A Step-by-Step Guide

This document provides a comprehensive guide to migrating from Shopify's legacy checkout to the new Checkout Extensibility framework. It includes technical steps, code changes, app extension usage, and custom events.

Table of Contents

1. Introduction

Shopify's new Checkout Extensibility is a secure, performant, and upgrade-safe way to customize the checkout experience using extensions, Shopify Functions, and pixels.

2. Prerequisites

- Shopify Plus plan
- GitHub account connected to Shopify admin
- Shopify CLI installed
- Familiarity with React, GraphQL, and Liquid

3. Overview of Checkout Extensibility

Checkout Extensibility includes:

- **Checkout UI Extensions**: For UI changes
- **Shopify Functions**: For backend logic (discounts, delivery options, etc.)
- **Pixels**: For customer behavior tracking
- **Post-purchase Extensions**: For upsells and surveys

4. Migration Steps

Step 1: Understand Your Existing Customizations

- Review custom scripts and checkout.liquid file
- Identify functionality needing migration

Step 2: Enable Checkout Extensibility

- In Shopify admin, navigate to **Settings > Checkout**
- Click "Upgrade to Checkout Extensibility"

Step 3: Rebuild Customizations using Checkout UI Extensions

- Use Shopify CLI: shopify extension create
- Choose **Checkout UI Extension**
- Use provided APIs: useApi, useExtensionInput, etc.

Example:

```
import {
  reactExtension,
  useApi,
} from '@shopify/ui-extensions-react/checkout';

export default reactExtension('Checkout::Dynamic::Render', () => {
  const {buyerJourney} = useApi();
  return <Text>Hello, custom checkout!</Text>; });
```

Step 4: Use Shopify Functions

- Use for logic like delivery date customization or automatic discounts
- Example setup for a Function (discount):

shopify generate function discount

Step 5: Use Pixels for Tracking

- Go to **Settings > Customer events**
- Add pixel code or use Shopify Pixels API

Example Pixel:

```
analytics.subscribe("checkout_completed", (event) => {
  console.log("Checkout completed:", event);
});
```

Step 6: QA and Testing

- Use the Checkout Preview tool
- Test on staging environment
- Validate extension behavior

Step 7: Launch

- Push extension to GitHub repo connected to Shopify admin
- Publish the extension

5. Code Changes

- Replace checkout.liquid customizations with React-based UI extensions
- Remove legacy script tags from theme files
- Migrate tracking codes to Pixels

6. Using App Extensions

- Build app extensions to deploy custom UI components
- Example: Post-purchase offer extension

7. Creating and Using Customer Events

- Use Shopify Pixels for tracking events like **checkout_started**, **checkout_completed**
- Use **analytics.subscribe()** API in custom pixels

8. Best Practices

- Keep extensions modular
- Use GitHub for version control
- Regularly test using checkout preview tool
- Monitor extension performance

9. Resources

- Shopify Checkout Extensibility Docs
- Shopify CLI
- Shopify Functions
- Shopify Pixels

For technical support and troubleshooting, refer to the Shopify Dev Community.

Author: Talal Haider