EcomLabs

MIGRATION GUIDE

Migrate a discount from Shopify Scripts to Functions

A volume promotion that lived in a Ruby Script, rewritten as a JavaScript discount Function: input query, function, shopify.extension.toml, five test carts with their real output, the mutation that creates the discount, and how to stack it with another one on the same line.

FOR	Shopify Plus stores that ran promotions in Shopify Scripts, and the apps migrating them
TESTED WITH	Node 22.22.3 · graphql 16.15.0 (Discount Function schemas 2026-04 and 2026-07) · Windows 11
UPDATED	November 11, 2026
DOCS	shopify.dev/docs/api/functions/latest/discount
ARTICLE	ecomlabs.dev/g/functions-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you are migrating
Step 01	Generate the extension
Step 02	The input query
Step 03	The function
Step 04	Test in Node with five carts
Step 05	Create the discount on a dev store
When you finish	Checklist

BEFORE YOU START

What you are migrating

No Shopify Script has run since June 30, 2026 (Shopify changelog, April 9, 2026). This guide rebuilds a promotion as a JavaScript discount Function, using the Shopify CLI template and Discount Function API 2026-10: products tagged `bulk-tier` get 10% off with 3 to 5 units on the line and 15% with 6 or more, never gift cards, and it combines with a loyalty discount on the same line. The original Script, which rewrote the price with `change_line_price`, is in the article.

STEP 01

Generate the extension

From your app folder:

```
shopify app generate extension --template discount --flavor vanilla-js --name volume-discount
```

The template ships two targets. This promotion only uses the cart lines one: delete the `cart.delivery-options.discounts.generate.run` target from the TOML, its `src/cart_delivery_options*` files, and its line in `src/index.js`.

In the TOML the CLI generated, set `api_version = "2026-10"`, keep `handle = "volume-discount"`, and leave a single `[[extensions.targeting]]` block: `target = "cart.lines.discounts.generate.run"`, `input_query = "src/cart_lines_discounts_generate_run.graphql"` and `export = "cart-lines-discounts-generate-run"`. `[extensions.build]` stays as generated (`path = "dist/function.wasm"`). The `handle` is the `functionHandle` in step 05.

STEP 02

The input query

Replace `src/cart_lines_discounts_generate_run.graphql`:

```
query CartInput {
  cart {
    lines {
      id
      quantity
      merchandise { __typename ... on ProductVariant { product { isGiftCard hasAnyTag(tags: ["bulk-tier"]) } } }
}
    }
  }
  discount { discountClasses metafield(namespace: "$app", key: "volume-tiers") { jsonValue } }
}
```

`hasAnyTag` returns a boolean: tags are not readable as a list. The discount's metafield holds the tiers.

STEP 03

The function

Replace `src/cart_lines_discounts_generate_run.js`:

```
// The tiers the Script had as constants. A "volume-tiers" metafield on the discount overrides them.
const DEFAULT_TIERS = [
  { minQuantity: 6, percentage: 15 },
  { minQuantity: 3, percentage: 10 },
];

export function cartLinesDiscountsGenerateRun(input) {
  if (!input.discount.discountClasses.includes("PRODUCT")) {
    return { operations: [] };
  }

  const tiers = [...(input.discount.metafield?.jsonValue?.tiers ?? DEFAULT_TIERS)]
    .sort((a, b) => b.minQuantity - a.minQuantity);

  const candidates = [];
  for (const line of input.cart.lines) {
    if (line.merchandise.__typename !== "ProductVariant") continue;
    const { product } = line.merchandise;
    if (product.isGiftCard || !product.hasAnyTag) continue;

    const tier = tiers.find((t) => line.quantity >= t.minQuantity);
    if (!tier) continue;

    candidates.push({
      message: `${tier.percentage}% off for ${tier.minQuantity}+`,
      targets: [{ cartLine: { id: line.id } }],
      value: { percentage: { value: tier.percentage.toFixed(1) } },
    });
  }

  if (candidates.length === 0) {
    return { operations: [] };
  }

  return {
    operations: [{ productDiscountsAdd: { candidates, selectionStrategy: "ALL" } }],
  };
}
```

The percentage is a string (`"15.0"`) and `ALL` applies every candidate (with `FIRST`, only the first). The template imports these values from `../generated/api`; here they are plain strings so it runs in Node.

STEP 04

Test in Node with five carts

In the extension folder, create `inputs/cart-mixed.json`, in exactly the shape the query returns:

```
{ "cart": { "lines": [
  { "id": "gid://shopify/CartLine/1", "quantity": 6, "merchandise": { "__typename": "ProductVariant",
    "product": { "isGiftCard": false, "hasAnyTag": true } } },
  { "id": "gid://shopify/CartLine/2", "quantity": 3, "merchandise": { "__typename": "ProductVariant",
    "product": { "isGiftCard": false, "hasAnyTag": true } } },
  { "id": "gid://shopify/CartLine/3", "quantity": 10, "merchandise": { "__typename": "ProductVariant",
    "product": { "isGiftCard": true, "hasAnyTag": true } } },
  { "id": "gid://shopify/CartLine/4", "quantity": 1, "merchandise": { "__typename": "ProductVariant",
    "product": { "isGiftCard": false, "hasAnyTag": true } } },
  { "id": "gid://shopify/CartLine/5", "quantity": 8, "merchandise": { "__typename": "CustomProduct" } }
] },
  "discount": { "discountClasses": ["PRODUCT"], "metafield": null } }
```

The other four are copies of that file with changes:

- `cart-qualifies.json`: line 1 only, with `"quantity": 4`.
- `cart-no-match.json`: line 1 with `"quantity": 2`, and line 2 with `"quantity": 5` and `"hasAnyTag": false`.
- `cart-config.json`: line 1 only, with `"quantity": 2` and `"metafield": { "jsonValue": { "tiers": [{ "minQuantity": 2, "percentage": 5 }] } }`.
- `cart-order-only.json`: line 1 only, and `"discountClasses": ["ORDER"]`.

Save `run-carts.mjs` next to `package.json` (the template already sets `"type": "module"`) and run it:

```
import { readdirSync, readFileSync } from "node:fs";
import { cartLinesDiscountsGenerateRun } from "./src/cart_lines_discounts_generate_run.js";

for (const file of readdirSync("inputs")) {
  const input = JSON.parse(readFileSync(`inputs/${file}`, "utf8"));
  const { operations } = cartLinesDiscountsGenerateRun(input);
  const candidates = operations[0]?.productDiscountsAdd.candidates ?? [];
  console.log(file, candidates.map((c) => `${c.targets[0].cartLine.id} ${c.value.percentage.value}%`));
}

$ node run-carts.mjs
cart-config.json [ 'gid://shopify/CartLine/1 5.0%' ]
cart-mixed.json [ 'gid://shopify/CartLine/1 15.0%', 'gid://shopify/CartLine/2 10.0%' ]
cart-no-match.json []
cart-order-only.json []
cart-qualifies.json [ 'gid://shopify/CartLine/1 10.0%' ]
```

That is the real output on Node 22.22.3. We also validated the query and every output against the 2026-04 and 2026-07 schemas. Inside your app, the next step is the compiled build: `shopify app function run --input inputs/cart-mixed.json --export cart-lines-discounts-generate-run` (we did not run it: no app was created).

STEP 05

Create the discount on a dev store

Create a dev store on the Plus plan from the Dev Dashboard, run `shopify app dev`, press `g` to open GraphiQL, pick the latest stable version, and run:

```
mutation CreateVolumeDiscount {
  discountAutomaticAppCreate(automaticAppDiscount: {
    title: "Volume discount"
    functionHandle: "volume-discount"
    discountClasses: [PRODUCT]
    startsAt: "2026-11-11T00:00:00Z"
    tags: ["volume"]
    combinesWith: { productDiscounts: true, productDiscountsWithTagsOnSameCartLine: { add: ["loyalty"] } }
    metafields: [{ namespace: "$app", key: "volume-tiers", type: "json",
      value: "{\"tiers\":[{\"minQuantity\":6,\"percentage\":15},{\"minQuantity\":3,\"percentage\":10}]}" }]
  }) {
    automaticAppDiscount { discountId status combinesWith { productDiscounts
productDiscountsWithTagsOnSameCartLine } }
    userErrors { field message }
  }
}
```

`tags` are the discount's own tags; `add` lists the tags it accepts on the same line. Updating `tags` replaces the old ones (to append, use `tagsAdd`). If `userErrors` says "Could not find Function", check that `app dev` is still running, the handle, and the `write_discounts` scope.

> **CAUTION**
>
> Same-line stacking is Shopify Plus only (Admin API 2026-04). We validated this mutation against the schema, but did not run it on any store.

Stack the loyalty discount and test

1. In the admin, create a product discount (say, $20 off t-shirts) tagged `loyalty`.
2. Under **Combinations**: **Product discounts**, **Multiple per product**, and `volume` under **Combine on same product with discounts tagged**.
3. Deactivate other discounts, try 2 units (nothing), 4 (10%), 6 (15%), and 6 with the loyalty discount (both; the percentage applies before the fixed amount), and replay each run with `shopify app function replay`.

WHEN YOU FINISH

Checklist

- [ ] Every promotion that lived in Scripts is written out in plain words, with its combinations.
- [ ] All five carts give the expected output in Node.
- [ ] Both discounts have reciprocal tags and the store is on Plus.