



Practical guides

Shopify Advanced Guide

Design, integrate and improve your store: catalogue, markets, automations, integrations and data-driven decisions.

Francesco Guiducci

Edition October 2026 · English

ifgdigital.com

For personal or internal business use; unauthorized distribution is prohibited.

Contents

How to use this guide	3
1 · Define the project and the expected result	3
2 · Separate environments, data and publishing	4
3 · Build a usable taxonomy	4
4 · Model metafields and metaobjects	5
5 · Make the product page verifiable	5
6 · Build accessibility into the journey	5
7 · Measure performance and fix the cause	6
8 · Organise SEO and content around real intent	6
9 · Open a market with a complete matrix	7
10 · Tie privacy and commercial rules to the configuration	8
11 · Build automations with checks and exceptions	8
12 · Design marketing and measurable funnels	9
13 · Run orders and support as one process	9
14 · Define the measurement model	9
15 · Run experiments and read cohorts	10
16 · Integrate apps, ERP and APIs with clear ownership	11
17 · Protect access, backups and recovery	11
18 · Evaluate dynamic pricing and promotions	12
19 · Choose Plus, Functions and checkout customisation	12
20 · Use AI, retention and community with concrete checks	13
Final lab: three scenarios with acceptance criteria	13
Acceptance log	14

How to use this guide

This guide starts from a store that already works and makes sense. It helps you organise larger catalogues, markets, automations, integrations and data decisions. "Advanced" describes the level of the work, not the Shopify plan every procedure needs: check availability by feature, account and country.

I'm Francesco Guiducci, founder of IFG Digital and a freelance Shopify consultant and developer based in Rome, with a background in mechanical engineering. On a project I tie requirements, implementation and testing together: a setting saved in the admin has to be checked on the path your customer or operator actually uses.

The scenarios are teaching examples, not results from clients. Legal and tax questions belong to the qualified professionals. Here you'll find how to turn an approved specification into technical requirements and test cases.

The twenty chapters cover three areas: configuration (chapters 1–10), operations (11–16) and control (17–20). The final lab and the acceptance log close the work.

1 • Define the project and the expected result

Start from an observed problem. "Increase conversions" is too broad to guide a change. "In the spare-parts catalogue, customers can't confirm compatibility and open tickets before buying" names the content, the path and the measure.

For each initiative, fill in:

Field	Teaching example
Problem	Spare-part compatibility is hard to find
Initial evidence	Classified tickets and a path test
Change	Structured data and a filter by model
Expected result	More correct choices, fewer repeat questions
Primary metric	Compatibility questions per 100 orders
Harm check	Returns for incompatible products
Owner and review	A named person and a date

Record baseline, period, source and limits. A target is a decision, not a guaranteed forecast. Rank work by how severe the problem is, how many people it affects and what the fix costs. A blocked checkout or a wrong price comes before a new animation.

Keep a decision log with scope, rationale and acceptance criterion. Whoever takes over the project should understand why a solution exists and when to revisit it.

2 · Separate environments, data and publishing

A theme copy lets you work on the front end. It doesn't duplicate the catalogue, payment accounts, apps, markets or orders. Define the real scope before you call an environment "staging".

For each change, identify the objects involved, the shared data, how you can test, and how you restore. A new line of theme text and an SKU migration need different protection. When you duplicate a store, Shopify doesn't copy orders, discount codes, issued gift cards, saved reports, apps, or tax, shipping, checkout and payment settings. You rebuild those by hand or with dedicated tools.

Release procedure:

1. Record the store, theme or integration and its starting version.
2. Keep a copy of the relevant data and code.
3. Apply the change within the agreed scope.
4. Check the result with positive and negative cases.
5. Collect the remaining issues and decide whether to publish.
6. Check the public result and the processes connected to it.
7. Roll back if the stop criterion is reached.

Rolling back a theme doesn't undo emails already sent, refunds or changes to external data. For those, prepare checks before execution and a realistic compensation plan, and don't promise full reversibility.

3 · Build a usable taxonomy

Keep standard category, product type, collection, variant option and filterable attribute apart. Don't give every tag the same job. Start from a representative sample that includes exceptions and incomplete items.

Build a vocabulary for colours, materials, sizes and compatibility, and give it an owner who handles synonyms and new values. "Blu", "blue" and "navy" can mean different things: don't merge them just to tidy the spreadsheet.

Use manual collections for editorial choices and automated ones for stable rules. A menu can show navigation levels without creating a hierarchy between collection objects. Check what happens when a product changes tag, price or availability.

A product can have up to 2,048 variants and three options. Above 100 variants, some third-party themes, public apps and sales channels may not be compatible: the platform limit doesn't prove the whole solution works.

Catalogue test. First, middle and last variant; a combination that doesn't exist; option changes; search; combined filters; no results; sold-out product. Check SKU, media, price and inventory in connected systems too. When a filter returns nothing, give the shopper a clear way to remove conditions.

4 - Model metafields and metaobjects

Use structured data when the same data has to be validated, filtered, translated or reused. A metafield adds a value to a resource; a metaobject organises a reusable entity with several fields. Let the content decide, not what the feature allows.

Teaching example: a shared size guide holds a title, unit, table rows and notes. Products point to the relevant guide. Changing the guide affects many products, so record the dependencies and who owns them.

Before you create the model, define the technical name, type, allowed values, whether editors must fill it in, translation, and what happens when the data is missing. Don't change the type of a populated field without a planned migration.

Procedure: write a definition, fill in three different cases, connect the data to a compatible theme and check how it renders. Test a full value, an empty one, a long one and an unavailable reference. A field created in the admin doesn't show on the site by itself. An empty table shouldn't leave a heading with nothing under it.

For implementation, use current documentation and the features your theme supports. Keep a data dictionary with resource, field, example, source and where it's used.

5 - Make the product page verifiable

Collect the questions that stop people choosing: compatibility, size, what's included, timing, care and terms. Put each answer next to the decision it supports, not inside an image or on a remote page.

Tie every claimed benefit to proof. Reviews, counts, certifications and availability need a source. A countdown timer shows a real deadline; a scarcity message follows actual stock and conditions. Good graphics don't make a false statement true.

For media and descriptions, use the dedicated resources. The job here is to verify the system: variant selector, price, gallery, availability, quantity, cart and messages must update together.

Technical scenario: a customer picks an accessory that fits only model B. Check that the title, description and recommendation don't offer it for model A. If compatibility is unknown, don't turn it into "compatible".

An upsell offers a different or higher-spec version; a cross-sell offers a complement. Measure contribution and returns as well as AOV: a bigger order can be worse if discount, shipping or support cost grows faster than revenue.

6 - Build accessibility into the journey

Use WCAG as the technical reference and have the applicable legal scope defined for your store. The European Accessibility Act covers e-commerce services, but "mandatory for everyone" doesn't replace an assessment of who is in scope, which services, the exceptions and national transposition.

For WCAG 2.2 level AA, normal text needs a contrast ratio of at least 4.5:1. Large text needs 3:1; the standard defines it as at least 18 point, or 14 point bold, and sets out exceptions. Don't apply 4.5:1 to every element, and don't call 16 px a universal legal minimum.

Do a manual pass on top of the automated tools:

- ☐ Reach the menu, filters, gallery, variants and cart by keyboard.
- ☐ Check visible focus, focus order and where focus returns after a panel closes.
- ☐ Zoom the content and confirm the actions still work.
- ☐ Check labels, error messages and control states.
- ☐ Judge text alternatives in context, including decorative images.
- ☐ Confirm that banners and dialogs don't cover the action people need.

Log the URL, component, step, expected result, observed result and evidence. A high automated score doesn't prove overall conformance. For recurring problems, fix the shared component and test every surface that uses it.

7 - Measure performance and fix the cause

Choose representative pages and repeatable conditions: device, viewport, network, cache state and content. Keep lab measurements apart from real-user data. A test score isn't a direct forecast of sales.

Before you remove an app, find out what it loads and where. Cut unneeded resources, size images to the space they fill, and check scripts that delay interaction. Don't lazy-load the main image that's visible when the page opens.

Example: a gallery first shows an empty image and then shifts the content. Check reserved dimensions, loading and variant selection. JPEG compression alone can lower the weight without fixing the shift.

Compare before and after on the same pages in the same conditions, then check that zoom, video, filters and variants still work. Record the concrete change: "removed a script loaded on every page but used in one section" says more than "optimised speed".

Set a regression criterion before release. If a change improves the speed test but stops people choosing a variant, it isn't acceptable.

8 - Organise SEO and content around real intent

Map the question, the page that answers it and the next step. A search about a size may call for a guide; a search for a specific model, for the product page. Avoid several near-identical articles competing for the same intent.

Write titles, descriptions and subheadings to explain the content. There is no general rule that the keyword must appear in the first 150 words. Keep URLs stable; if you change them, map the redirects and internal links.

Check the public page, indexability, canonical, language and links. Use Search Console to separate crawling, indexing and performance. A submitted sitemap doesn't guarantee every page is indexed, and impressions aren't a ranking position.

Before adding a structured-data app, check what the theme already generates: two systems can produce duplicate or conflicting data. Price, availability and reviews in the markup must match the real content.

Don't promise FAQ rich results. Google stopped showing them from 7 May 2026 and removed the documentation on 15 June 2026. FAQs are still useful when they answer your customers' questions.

For external links, look for relevance and editorial value. Don't buy link schemes, and don't promise a SERP position. Track pages and queries over time, noting catalogue and campaign changes.

9 · Open a market with a complete matrix

Pick a pilot market you can sustain and define language, products, prices, currency, payment, delivery, returns and support. Translation alone isn't localisation.

Element	Proof required
Catalogue	Products available and permitted in the market
Language	Terminology, units and terms reviewed
Price and currency	Consistent across page, cart and checkout
Payment	Method available in the customer's context
Logistics	Rate, timing, tracking and returns process
Tax and customs	Approved specification and a test total
Support	Channel and reply in the intended language

Keep the local selling currency apart from the payout currency. Shopify shows prices in local currencies only with Shopify Payments or Adyen; with other providers, every market pays in the store's default currency. When a customer pays in a currency other than your payout currency, a conversion fee applies. Provider and plan affect availability, so don't credit every international feature to Advanced or Plus alone.

For domains, subfolders and hreflang, check what is actually published, and read the existing implementation before adding a second one. Test the language switch, links and the way back to the starting market. Terms for managed services and local payment methods need checking against current sources, not an old list.

Before extending to another country, reconcile at least one full operating cycle of the pilot: order, delivery, any return, and cost. An order the checkout accepted doesn't show that the whole service works.

10 · Tie privacy and commercial rules to the configuration

Keep a requirements register: approved source, affected configuration, owner and proof. For discounts, keep price history and conditions; for tax, specify country, customer and product; for personal data, state purpose, access, retention and request handling.

OSS, IOSS, reverse charge and data erasure aren't universal automatisms. A professional assesses the case and the retention obligations; the technical team implements what is approved and tests the scenarios.

Google Consent Mode governs how tags behave according to consent state. It isn't a banner and it doesn't certify compliance: you still collect consent through a banner or a consent management platform and pass the state to Google. Capturing the choice, passing the state and tag behaviour are three separate steps.

Technical matrix. First visit, rejection, acceptance, partial choice, withdrawal and a return visit. For each case, observe what the tags and integrations in the specification actually do. Don't write "no data sent" without measuring it: implementation modes can differ.

Track an erasure request across the relevant systems, with identity checks and the agreed handling. The button in Shopify doesn't show that every external copy has been dealt with.

11 · Build automations with checks and exceptions

Shopify Flow is included on the Basic, Grow, Advanced and Plus plans; some actions and limits depend on the plan. The Send HTTP Request action is available on Grow, Advanced and Plus.

Before building the flow, write the trigger, conditions, actions, exclusions and expected result. Decide who receives errors, how to switch it off and how to prevent duplicate effects.

Teaching scenario, stock reorder:

1. Event: inventory change on the variant.
2. Condition: relevant location and product, quantity below the agreed threshold.
3. Check: is there already an open request for that variant and location?
4. Action: create or update an internal request in the chosen system.
5. Close-out: the owner confirms the reorder or records why it was excluded.

The flow shouldn't create ten identical requests for ten updates. Test threshold reached, back above threshold, repeated event, excluded location and missing data. Log every run and the decision taken.

For marketing, check Shopify Messaging, Shopify's current system for email and automations, or the provider you chose. Tell abandoned cart, checkout started and purchase completed apart. Plan for exclusion after purchase, unsubscribes and frequency caps. Having an email address doesn't make an audience authorised.

Save first without activating. Check content and recipients on an authorised test, then re-read the status and the runs. "Active" doesn't prove delivery or that the final message is correct.

12 · Design marketing and measurable funnels

Tie each channel to a stage: discovery, evaluation, purchase or repurchase. Specify audience, content, destination and measure. A complete project doesn't need every social channel or every pixel.

For a campaign, check the message and the page together. An ad that promises a kit must land on the right kit, with matching contents and price. Use readable campaign parameters and don't rename them mid-comparison.

Cross-sell test: choose a compatible, available accessory. Check that it appears only where intended, and test variant selection, quantity, price, cart and removal. Compare contribution per order, take rate, errors and returns. A click on the widget isn't an incremental sale.

For checkout and post-purchase offers, check which surfaces and eligibility actually apply. The theme cart, the checkout and the post-purchase page are different environments, and not every app can place UI at every step.

Email sequences have no universally optimal cadence. Start with a useful message, a frequency cap and exclusion rules. Look at unsubscribes and complaints as well as clicks and attributed revenue; open rates can be distorted by how email clients work.

13 · Run orders and support as one process

Map the transitions: order received, payment and risk check, assignment, picking, fulfilment, transit, delivery and follow-up. Define the source for each state and who can change it.

A warehouse sync can lag or fail. Prepare an exceptions view: product unavailable, incomplete address, split order, missing tracking, refund pending. Don't describe an integrated ERP as "error-free".

Partial-order scenario: two lines, one ready and one delayed. Check which lines are fulfilled, which tracking is sent and what stays open. The notification mustn't say everything has shipped. If the customer cancels the remaining line, check the refund and the inventory separately.

For support, use useful categories and an owner for non-standard cases. A chatbot can collect data or answer from approved instructions, and it must be able to hand the conversation to a person. Don't present an unstaffed channel as an instant reply.

Measure time to first reply, time to resolution, reopenings and contact reason. If many customers ask the same sizing question, fix the product page as well as the canned reply.

14 · Define the measurement model

For each metric, write the name, formula, scope, source, frequency and limit. Don't combine numbers just because their labels look alike.

Metric	Definition to fix
Store conversion	Converted sessions ÷ sessions in the chosen scope
AOV	Revenue counted ÷ orders included, with explicit rules
CAC	Acquisition spend ÷ attributed new customers
Repurchase	Customers with a later purchase within a defined window
Contribution	Revenue minus the variable costs included in the model
Observed CLV	Value earned within the window, kept apart from future forecasts

Shopify's main analytics features and report explorations aren't limited to the higher plans: you can save custom explorations whatever your subscription. Some reports do have their own plan requirements, so before buying another tool, check the data, permissions and reports already in your account.

For GA4, start from Shopify's Google & YouTube channel. Take stock of the pixels, apps and containers already running before you add GTM. Once set up, Shopify tracks ecommerce events such as product views, add to cart and purchase automatically. "Enhanced Ecommerce" isn't a GA4 switch, and recommended events have names and parameters defined by Google.

Check the event, transaction ID, value, currency and items on an authorised test path. Check consent and duplicates. A connected integration doesn't prove every event is right, and GA4 won't necessarily match Shopify because of attribution, consent and scope.

When the numbers disagree, take a small sample of orders and follow each step. Fix the flow before raising the budget on a doubtful report.

15 · Run experiments and read cohorts

Write the hypothesis before you change the page. State the primary metric, harm criteria, audience, planned duration and stop conditions. Don't stop a test at the first favourable result.

Teaching example: group A has 1,000 sessions and 20 converted; group B has 1,000 and 24. The rates are 2% and 2.4%: a 0.4 percentage-point difference, a 20% relative increase. These numbers alone don't show a reliable effect: you need the test design, the sample and the uncertainty.

If traffic is too low for a useful statistical experiment, run usability tests and descriptive comparisons. State the limit rather than turning one week of sales into certain causation. Check seasonality, campaigns, availability and product mix.

For retention, compare cohorts over the same observation window. A thirty-day cohort can't be compared directly with one that has had a year to repurchase.

Monthly routine: check data quality; note what changed; pick one question; analyse the relevant segment; choose an intervention; set the next check. A dashboard with no decisions attached becomes an archive of numbers.

16 · Integrate apps, ERP and APIs with clear ownership

For each piece of data, pick the authoritative system: who decides price, stock, description, consent and fulfilment status? Avoid two systems overwriting each other with no conflict rule.

Document the object, stable ID, direction, frequency, transformations, errors and owner. An SKU can change, so consider the technical identifiers that keep the link. Don't assume a sync is instant because the app says "real time".

For new integrations, use the GraphQL Admin API on a supported version. Shopify has classed the REST Admin API as legacy since 1 October 2024, and since 1 April 2025 new public apps must use GraphQL only. Swapping the URL isn't enough: check queries, mutations, permissions, pagination, limits and application errors.

Webhooks can be delivered more than once. Verify authenticity with the HMAC signature (the `X-Shopify-Hmac-SHA256` header, computed over the raw body) and handle duplicates with idempotent operations. If that isn't possible, deduplicate on `X-Shopify-Webhook-Id`. Receiving the same event twice must not create two orders or two refunds.

Integration tests:

- ☐ Valid event, object found.
- ☐ Missing or deleted object.
- ☐ Duplicate event.
- ☐ Timeout after a successful write.
- ☐ Late-arriving event.
- ☐ API limit hit or permission revoked.
- ☐ Restart after an outage.

When a response is uncertain, read the state before repeating a mutation. Keep logs without credentials or unnecessary personal data, an error queue and a periodic reconciliation process. A "done" notification doesn't prove the receiving system applied the change.

17 · Protect access, backups and recovery

Keep an inventory of users, apps, tokens and operational owners. Use least-privilege permissions and strong authentication. Don't demand from every app the same attestation you'd expect from a payment system: judge its role and the data it handles.

For backups, list what is covered and what isn't: theme, catalogue, media, content, settings, app data and relationships. An exported file is only useful if you know how to restore it. CSV exports don't cover every object: orders, discount codes, settings and app data need other procedures.

Recovery test: pick a non-critical object in a suitable environment, keep the starting version, change the data and restore it. Check references, images and linked fields too. Record how long it took and how much data loss is acceptable. A file on disk isn't enough to call a backup complete.

Don't add a generic CSP through a meta tag, or an old CAPTCHA, as a universal recipe: you can block scripts you need without covering the risk. Design every theme security change for the Shopify environment and test it with the real integrations.

In an incident, preserve the evidence, limit the compromised access with the owner, contact the relevant support and involve whoever assesses notification obligations. Don't message customers on the basis of unverified assumptions.

18 - Evaluate dynamic pricing and promotions

Set the constraints before the algorithm: minimum contribution, variation cap, frequency, markets, excluded products and approval. A competitor's price doesn't contain your costs and shouldn't be copied automatically.

For a bulk update, prepare a proposal with old price, new price, reason and estimated impact. Check the tax basis, currencies and promotion rules. Test a small batch, re-read the values and check storefront and checkout before extending.

Hypothetical example. Revenue €50, variable costs €30, fee 2% plus €0.25, with the percentage applied to the same revenue. Fee = €1.00 + €0.25 = €1.25. Contribution = 50 – 30 – 1.25 = €18.75. With a 20% discount, revenue falls to €40, the fee to €0.80 + €0.25 = €1.05, and contribution to 40 – 30 – 1.05 = €8.95. That's a drop of about 52.27%, not 20%.

VIP prices and bundles need a verified access rule and a check of how they combine. AOV isn't the only objective. For formulas and the scenario sheet, use the dedicated pricing and margins resource.

19 - Choose Plus, Functions and checkout customisation

Judge a higher plan on demonstrated requirements and total cost: current subscription, implementation, apps, maintenance and work saved. Don't rely on the old \$2,000-a-month figure, and don't promise the investment pays back. Ask Shopify for the current price.

Checkout UI extensions in the information, shipping and payment steps require Shopify Plus. Other surfaces have their own availability, so check the exact target. Plus doesn't mean freedom to rewrite any part of the checkout.

Shopify Functions customises backend logic through apps. Public apps distributed in the Shopify App Store that contain Functions work on any plan; custom apps that use the Function APIs require Plus, and some capabilities carry further limits even in public apps. Functions doesn't replace apps: it's an extension mechanism.

Feasibility sheet: requirement, surface, API or extension, plan needed, limits, minimum test, cost and fallback. For a discount rule, test an eligible cart, an excluded one, a mixed one, a different currency and combinations. For a UI, check both the accelerated and the standard path where supported.

If your current plan already meets the requirement, don't migrate just because the project is called Advanced.

20 - Use AI, retention and community with concrete checks

Use AI tools for drafts, classifications or hypotheses. Give them approved data and ask them to flag what's missing. Check names, sizes, terms and sources before publishing: fluent text can contain an invented feature.

For descriptions and translations, prepare a glossary and a list of permitted claims, then compare the draft with the spec sheet. For analyses, ask for the formula, scope and data used; a recommendation isn't a validated forecast. Don't attribute current capabilities to old product names or retired tools without proof.

For retention, identify why people come back: a consumable they reorder, a compatible accessory, useful content or a service. Don't assume keeping a customer always costs less than winning a new one. Measure contribution after incentives and support.

A points programme explains how points accrue, how they're used, when they expire and how refunds are handled. A referral scheme needs rules against abuse and costs you can sustain. For subscriptions, test renewal, failed payment, change and cancellation with the integration you chose.

A community needs moderation, a sustainable schedule and someone who replies. Without a reason to take part, it's just one more channel to run. Start from an observed need and judge usefulness by relevant participation and feedback, without virality targets.

Final lab: three scenarios with acceptance criteria

A - Technical catalogue and a second market

Prepare ten sample products, two compatibility rules and one translation. Check attributes, filters, fallbacks and the purchase path in the pilot market.

Close-out A. Product, price, delivery and support match the specification. A changed language isn't enough.

B · Internal-request automation

Build an inactive flow for low stock. Test it with a valid event, a repeated one and an irrelevant one.

Close-out B. The duplicate event creates no double work and errors have a recipient. Activation is a separate decision.

C · Product-page experiment

Define the hypothesis, metric and harm conditions. Prepare two versions with identical data and a single meaningful difference. Check usability and measurement before assigning traffic.

Close-out C. At review, separate a descriptive result, sufficient evidence and an inconclusive outcome.

Acceptance log

Area	Proof	Result	Issue and owner
Catalogue and data	Sample and cases: ...	...	...
Accessibility and performance	URLs and conditions: ...	...	...
Markets and checkout	Matrix: ...	...	...
Automations	Runs and duplicates: ...	...	...
Measurement	Events and reconciliation: ...	...	...
Integrations and recovery	Errors and restore: ...	...	...

Use explicit results: **verified**, **needs fixing**, **unverified**, **not applicable** (with the reason). The next review starts from the problems and what has changed, not from a checklist ticked once and for all.

The Basic guide covers the first operations. The SEO, KPI and Funnel checklists develop the specialist checks. This guide connects architecture, implementation and operational decisions.

Sources

- Shopify Help Center, duplicating a store: <https://help.shopify.com/en/manual/shopify-admin/duplicate-store/>
- Shopify Help Center, variants: <https://help.shopify.com/en/manual/products/variants/add-variants>
- European Commission, European Accessibility Act: https://commission.europa.eu/strategy-and-policy/policies/justice-and-fundamental-rights/disability/european-accessibility-act-eaa_en
- W3C, WCAG 2.2: <https://www.w3.org/TR/WCAG22/>
- Google Search Central, updates (FAQ rich result retirement): <https://developers.google.com/search/updates>
- Shopify Help Center, local currencies: <https://help.shopify.com/en/manual/markets/customizations/local-currencies>
- Google, Consent Mode: <https://developers.google.com/tag-platform/security/concepts/consent-mode?hl=en>
- Shopify Help Center, Shopify Flow: <https://help.shopify.com/en/manual/shopify-flow>
- Shopify Help Center, Shopify Messaging automations: <https://help.shopify.com/en/manual/promoting-marketing/create-marketing/shopify-messaging/marketing-automations>
- Shopify Help Center, reports overview: <https://help.shopify.com/en/manual/reports-and-analytics/shopify-reports/report-types/reports-overview>
- Shopify Help Center, Google Analytics setup: <https://help.shopify.com/en/manual/reports-and-analytics/google-analytics/google-analytics-setup>
- Shopify Dev, REST Admin API (legacy): <https://shopify.dev/docs/api/admin-rest>
- Shopify Dev, verifying webhooks and handling duplicates: <https://shopify.dev/docs/apps/build/webhooks/verify-deliveries>
- Shopify Dev, checkout UI extensions: <https://shopify.dev/docs/api/checkout-ui-extensions/latest>
- Shopify Dev, Shopify Functions: <https://shopify.dev/docs/apps/build/functions>

Sources checked on 17 October 2026.