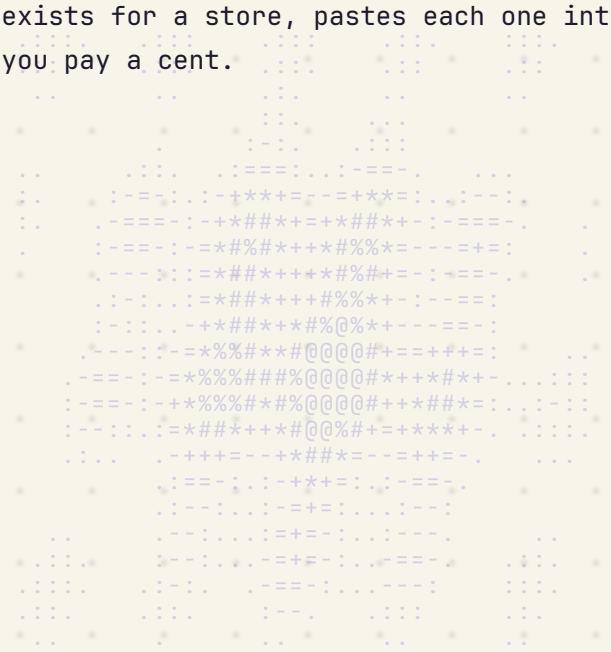
// SECTION 01 · PROMPT PACK

NEVER PAY FULL PRICE

Four prompts that turn Claude into your discount agent. Sweep every code that exists for a store, test them one by one in your real cart, stack the layers a code cannot reach, and claim the price drop after you pay.

```
> boot opusjake_os
> resource: never-pay-full-price
> version: v1.0 · 2026
> status: ready_
```

Coupon sites are mostly dead codes and ads. The real discount is whichever code actually clears in your cart, and the only way to know is to try them. With Cowork on, Claude does that part: it hunts down every code that exists for a store, pastes each one into your real cart, and reports the winner before you pay a cent.



THE SWEEP

Every code you will ever be offered comes from one of five places, and the ones that still work are almost never the ones ranked first on Google. Build the cart first: sign in, correct size, correct shipping address, correct quantity. The total has to be real before anything below means anything.

Turn on Cowork so Claude gets its own browser. Then send this.

// PROMPT 1 — FIND EVERY CODE THAT EXISTS

```
I am buying [ITEM] from [STORE URL]. My cart total is [$TOTAL] before
shipping and tax.

Find every discount code that could apply to this order today. Check:
- the coupon aggregators (RetailMeNot, Slickdeals, CouponFollow,
  Honey's public code list)
- creator and affiliate codes in YouTube video descriptions and
  Instagram bios from the last 90 days
- the brand's own first-order popup offer for a new email or SMS signup
- student, military, teacher, first-responder, and nurse discounts
  (SheerID or ID.me verified)
- the brand's current sitewide sale, outlet section, and any bundle
  price for the same item

Give me a table: code, claimed discount, source, date last reported
working, and any minimum spend. Sort by expected savings. Flag anything
that looks auto-generated or unverifiable so I know not to waste a try.
```

You will usually get 8 to 20 candidates. Most are dead. That is fine, because you are not the one testing them.

THE TEST

Here is the part people quit on. Codes fail silently, one at a time, and after four rejections most of us shrug and pay. Claude does not get bored on attempt eleven.

// PROMPT 2 — TRY THEM ALL IN THE CART

```
Now test them. Open my cart at [STORE URL] and apply each code from
your table one at a time, top to bottom. After each attempt record:
the code, whether it was accepted, the exact new total, and the error
message if it was rejected.

Most stores only allow one code per order, so I want the single best
one, not a combination. If the site does allow stacking, test the best
pairs too.

When you are done, leave the winning code applied, show me the full
before-and-after math, and stop at the review screen. Do not enter
payment details and do not place the order. I will click pay.
```

Two rules keep this clean. Claude stops before payment, always, and it never creates accounts or types your card number. You want a hunting agent, not a spending one.

One trap worth naming: a code that pulls you under the free-shipping threshold can leave you worse off. Prompt 2 measures the final total, not the discount percentage, which is why the math is stated in full.

THE STACK

A store may cap you at one code, but a code is only one of five discount layers, and the other four ignore each other entirely. This is where a 15 percent order quietly becomes 35.

// PROMPT 3 — LAYER THE DISCOUNTS A CODE CANNOT REACH

```
The code is locked in. Now find the layers that stack on top of it for
this order:

1. Cashback portals (Rakuten, TopCashback, Capital One Shopping).
   Compare current rates for this store and tell me which pays most.
2. Discounted gift cards for this brand on the resale marketplaces.
   If I can buy the balance below face value, that comes straight off.
3. Card-linked offers already sitting in my Amex, Chase, or Citi
   account for this merchant.
4. Price matching. Check whether a competitor sells the identical SKU
   cheaper and whether this store honors a match.

Rank them by dollars saved on my specific total, note the order I have
to do them in (portal click first, gift card purchased before checkout),
and tell me which ones actually combine with the code I already have.
```

Sequence matters more than effort here. A cashback portal only pays if you click through it before the cart is filled, so this prompt runs before you pay, not after.

// PROMPT 4 — CLAIM THE DROP AFTER YOU BUY

```
I paid [$TOTAL] for [ITEM] on [DATE]. Set up a scheduled task that
checks this item's price at [STORE URL] every day for the next 30
days. Look up this retailer's price adjustment policy and tell me the
exact window. If the price drops inside that window, alert me with the
new price, the policy language, and a short message I can send to
support to claim the difference.
```

Most retailers refund the difference if the price falls within 14 to 30 days, and almost nobody claims it because nobody is watching. Now something is.

BEFORE YOU RUN IT

- **Build the cart first.** Signed in, right address, right quantity. Shipping and tax move the total, and every prompt above is doing math against it.
- **Assume one code per order.** The goal is the biggest single code plus the non-code layers from Prompt 3. Sites that truly stack are rare and Prompt 2 will find them.
- **The brand's own popup is often the winner.** A fresh email or SMS signup is routinely 10 to 20 percent and it beats most aggregator codes. Use an alias address if you do not want the list.
- **Never hand over payment details.** Claude fills the cart and applies codes. You type the card and click pay. That line stays fixed.
- **Skip the code generators.** Codes come from real promotions, real creators, and real signup offers. Anything a site claims to have generated for you is a fake, and Prompt 1 is written to flag them.
- **Walk away for a day when you can.** Leave a full cart, close the tab, and the abandoned-cart email often shows up in 24 to 48 hours with a better offer than anything you found.

WHERE DISCOUNTS HIDE

The levers Prompt 1 pulls, worth knowing so you recognize a win when you see one.

- Creator and affiliate codes in video descriptions, which stay live far longer than aggregator codes
- The first-order email or SMS popup, and the second one that appears when you try to leave
- Abandoned cart emails, 24 to 48 hours after you stop
- Cashback portals, which pay on top of any code
- Gift cards bought below face value on resale marketplaces
- Card-linked offers already sitting unused in your bank app
- Student, military, teacher, and first-responder verification
- The outlet or last-chance section carrying the same SKU
- Price matching against an identical listing elsewhere
- The price adjustment window after you have already paid

Run Prompt 1 the moment the cart is full. The whole sequence takes about five minutes of Claude working while you do nothing, and the worst outcome is that you pay exactly what you were already about to pay.