What's Headless E-Commerce?

We may be running out of names

Justin
Nov 2, 2021

33 7

The TL;DR

Headless e-commerce is like Shopify, but without most of the visuals you're used to: it's a pure backend and management portal for running and building shops on the web.

- Like any web app, e-commerce sites have two components: **a frontend and a backend**
- The frontend is what you **see and interact with**; the backend is the **database and APIs**
- Headless e-commerce gives you a **pre-built, configurable backend** for your store; think orders, customers, products, etc.
- Things **aren't all code** though: headless products usually give you some sort of admin portal for configuring your setup

The headless concept isn't new – you might have heard the term in the context of a headless CMS or JAMStack apps. Headless basically makes building, configuring, and maintaining your backend easy, while giving you the flexibility to make the frontend look like whatever you want.

Refresher: how apps are built

An ecommerce store on the web is just a type of web application, same as Gmail or Twitter. So to understand what "headless" means, we need to break down how a web app works. Today is your lucky day, as I've already written about this at length here. But because I, too, wouldn't want to click that link, here's a bit of context from that post.

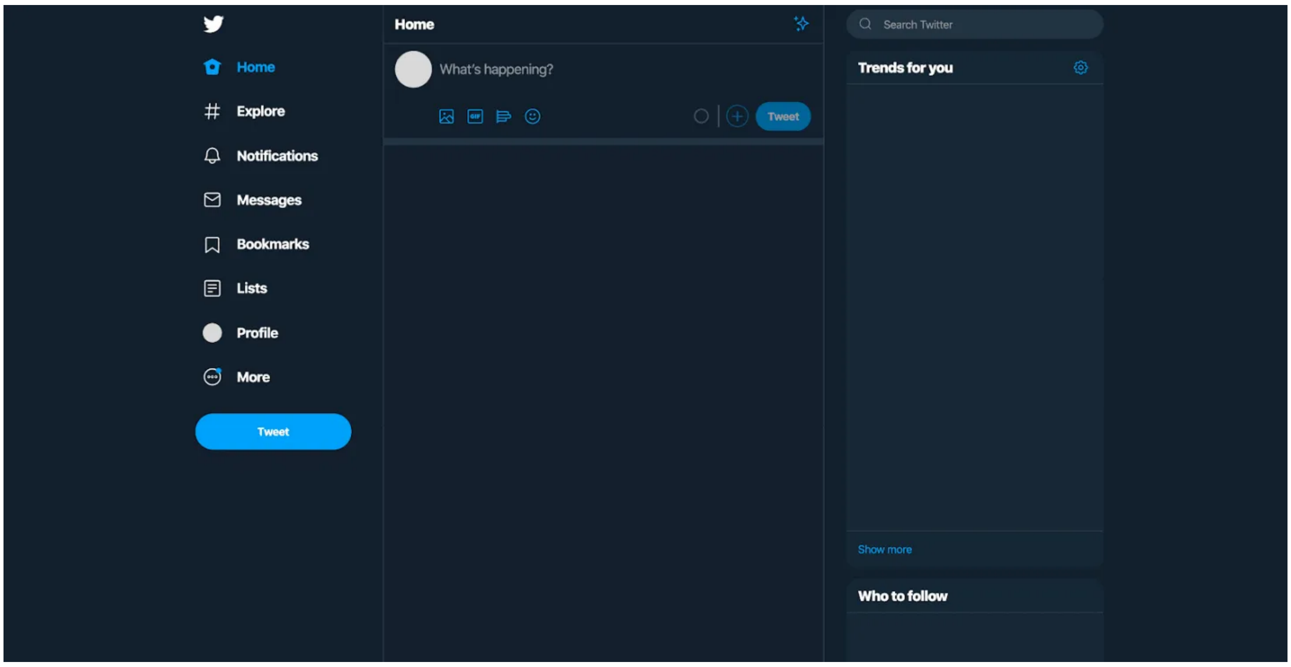
Pretty much all applications out there exist in this paradigm of **frontend** and **backend**. Traditionally, here's how the split works:

- **The frontend** is what you see and interact with as a user: text, shapes, links, and all of that. It's usually built in some combination of HTML, CSS, and JavaScript.
- **The backend is the data and logic that powers that frontend: APIs, business logics, and a database. It's usually built in a language like JavaScript, Python, Go, C++, etc.**

This always confused me – and the metaphors that you see online didn't help either – until I was able to break down an example. Let's take a look at Twitter.com, which you can load in your browser (skipping over the mobile apps for now):

- Twitter's **frontend** is the UI elements you see on the screen and how you interact with them. It's built in HTML, CSS, and if I believe the internet, React.
- Twitter's **backend** stores all historical tweets, who you follow, what topics are trending, and anything data related. The frontend gets that data to you via API endpoints.

Without the backend, Twitter's frontend might look like this:



Frontend isn't just shapes and colors though; there's plenty of important logic in there that *interacts with* the backend (like making API requests to get data). The reality is that the border between these two things isn't exactly scientific, but it's a useful framework for thinking about apps. It's also how job postings are (sometimes) split:

- **Frontend engineers** will work on the app's UI, and use API endpoints that other engineers have already built
- **Backend engineers work on building and maintaining API endpoints, the database, and other behind the scenes application or infrastructure logic**
- **Full stack engineers do both! (the whole stack)**

With this distinction in mind, let's shift the conversation back into ecommerce. What does the split between frontend and backend look like there?

Frontends and backends in ecommerce

Like Twitter, most stores you see on the web have a frontend and a backend to them.

→ **Ecommerce backends**

Like software engineers need to do when they develop an application from scratch, let's think through the kinds of data that we'll need to store to power our, uh, store. There are a couple of "concepts" that need to exist:

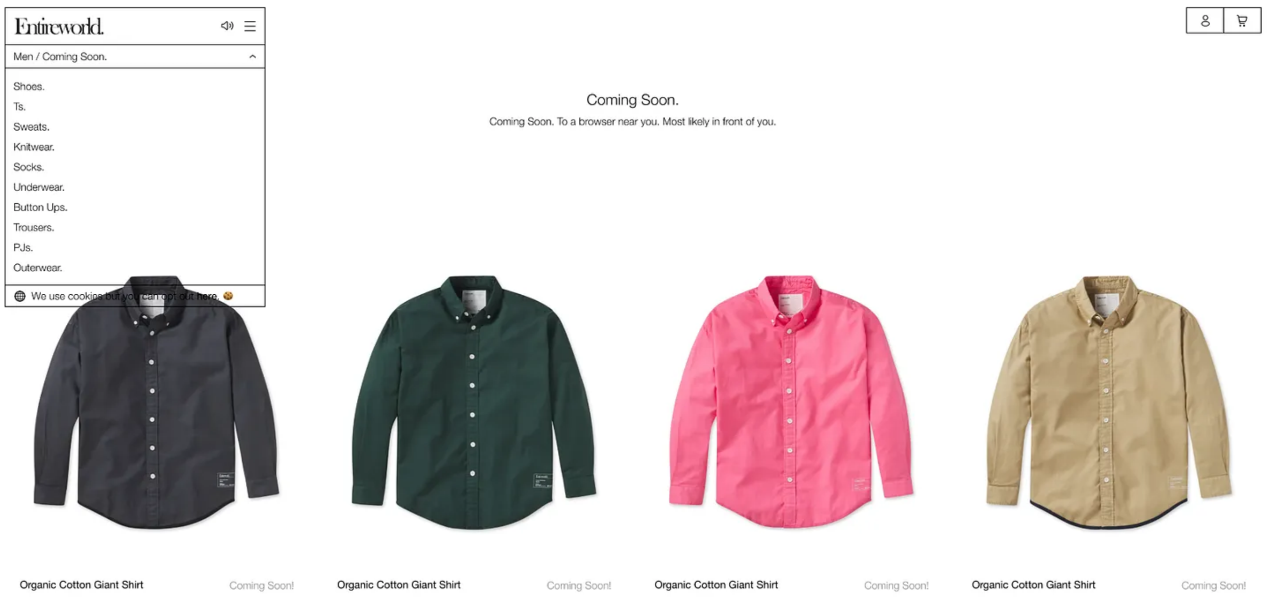
- A **user** – email, address, payment information
- A **product** – details, price, pictures, inventory/stock
- An **order** – size, time

These may be decoupled in the actual database – e.g. we may want to store payment methods in a separate table than users – but the general theory holds.

We don't want our frontend writing directly to the database, so applications usually have API endpoints that sit in the middle. We might have one called `createOrder` that takes a few parameters – order value, time, and products included – and adds a new row to the `orders` table, while kicking off the packing and shipping process. There might also be endpoints for authentication, so that you can sign into the store and not have to add your shipping address every goddamn time.

→ Ecommerce frontends

The frontend of the store takes all of that nicely stored information from the backend and displays it to our visitors, while letting them interact with it, buy things, etc. Take a look at Entireworld's site, for example:


Entireworld.
Men / Coming Soon.
Shoes.
Ts.
Sweats.
Knitwear.
Socks.
Underwear.
Button Ups.
Trousers.
PJs.
Outerwear.
We use cookies but you can opt out here.
Coming Soon.
Coming Soon. To a browser near you. Most likely in front of you.
Organic Cotton Giant Shirt
Coming Soon!
Organic Cotton Giant Shirt
Coming Soon!
Organic Cotton Giant Shirt
Coming Soon!
Organic Cotton Giant Shirt
Coming Soon!

On the backend, they're storing information about all of these Organic Cotton Giant Shirts, including photos, price, how many are available, what sizes are in stock, all of that. The frontend reads that information and displays it to the user – you – in a visually pleasing format.

> 🚨 **Confusion Alert** 🚨
>
> It can take a while for the split between frontend and backend to make sense. When you look at a web page, try to think about which pieces of information are *data* and which are *style*. The navigation bar, the fact that these shirts are in a grid, where the product information is placed; this is all frontend. The pictures, product names, and availability information *itself* is part of the backend.

> 🚨 **Confusion Alert** 🚨

(sad) edit: Entireworld shut down since I wrote this :(

What's headless ecommerce, then?

With all of that in mind, it should now be relatively easy to understand what headless ecommerce means. Anything that's **headless** – and you may have heard of a headless CMS as well – just means that the frontend is left to you, and is customizable and configurable. A headless ecommerce platform will build these basic backend elements for you, so you can focus on making your store look nice, and get out there and sell.

Let's use Swell as an example. When you set up your store with Swell, you use their SDK to get your backend in order instead of building it from scratch. Swell-provisioned backends have utilities for creating, reading, updating and deleting:

- **Orders**
- **Products**
- **Stock**
- **Carts**
- **Payments**
- **Shipments**
- **Subscriptions**
- + more

Instead of creating these tables and endpoints yourself (and dealing with error handling and edge cases), Swell does all of that for you. They also take care of sending webhooks when events occur, localizing to different languages, and advanced aggregation-like queries for analytics. If you want to create an order, you'd just use their SDK:

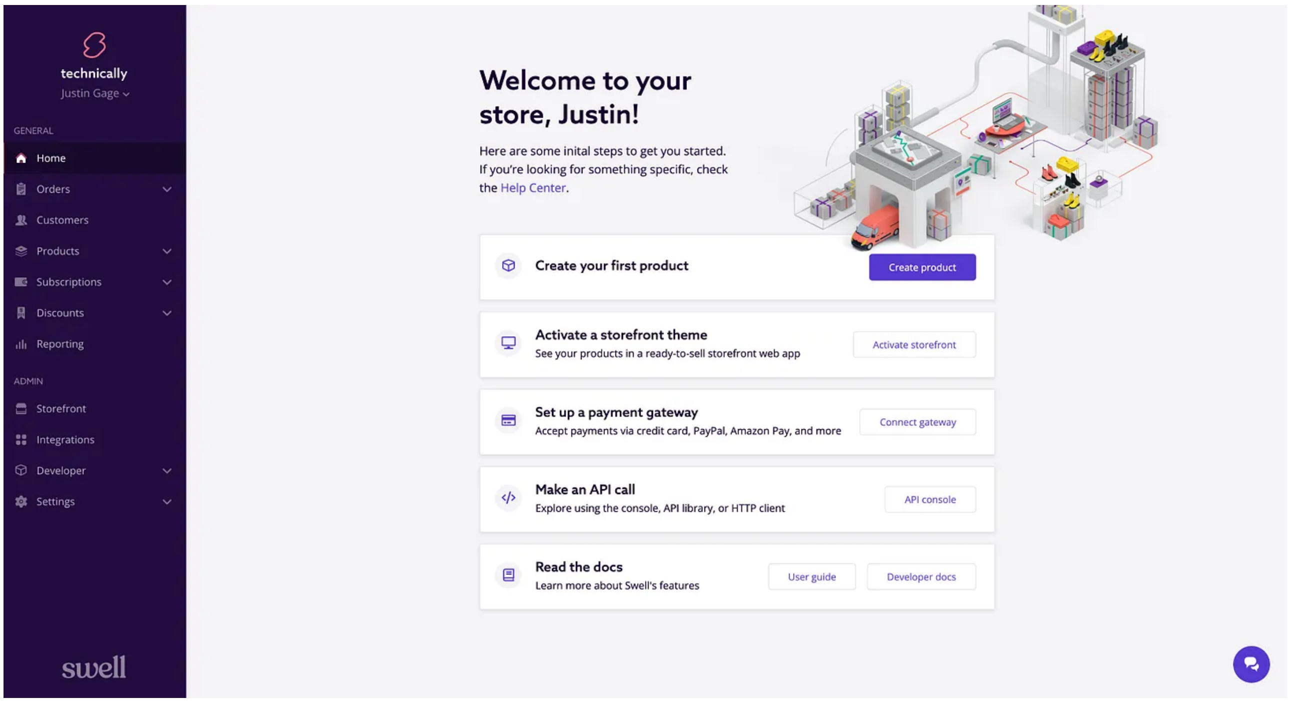
```
await swell.post('/orders', {
  account_id: '5a9ea7ba3f95740a914267f1',
  items: [
    {
      product_id: '5cad15bc9b14d1990724663a',
      quantity: 2,
    }
  ],
  billing: {
    ...
  },
  shipping: {
    ...
  },
  coupon_code: 'FREESHIPPING',

});
```

🤔 Undefined Terms 🤔

An SDK is a series of APIs grouped together for some unified purpose. Swell has many, many endpoints you can use to build your store, and all together they can be referred to as an SDK. More on this in the Technically post about APIs.

🤔 Undefined Terms 🤔

Finally, beyond just the backend, Swell gives you a nice admin dashboard for managing the logistics behind your store. You can upload new products, adjust inventory, and update customer information without having to write any code.



Welcome to your store, Justin!

Here are some inital steps to get you started. If you're looking for something specific, check the Help Center.

Create your first product — Create product

Activate a storefront theme
See your products in a ready-to-sell storefront web app — Activate storefront

Set up a payment gateway
Accept payments via credit card, PayPal, Amazon Pay, and more — Connect gateway

Make an API call
Explore using the console, API library, or HTTP client — API console

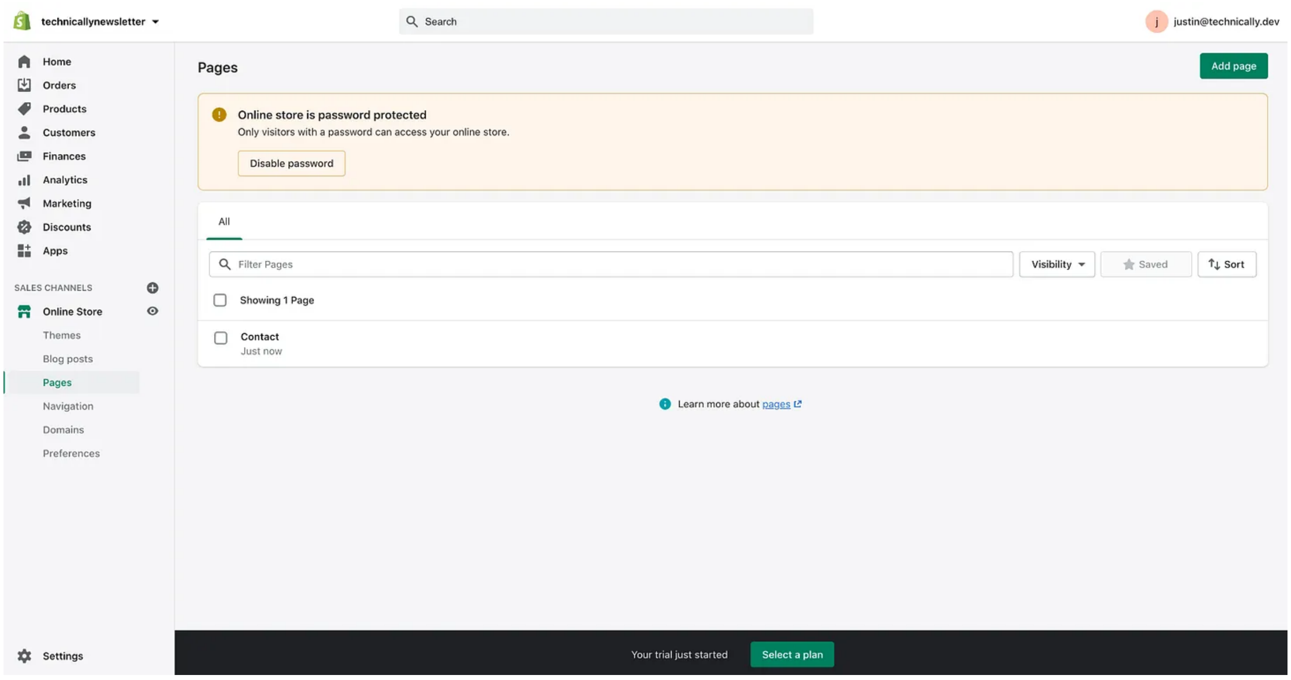
Read the docs
Learn more about Swell's features — User guide — Developer docs

For a useful mental model, you can think about products like Swell as a **Stripe for ecommerce**. The target audience is developers, and the product's goal is to make tedious backend work easy peasy; and alongside that, you get a nice UI for admin work.

Headless ecommerce vs. Shopify

The dominant standard for building web stores is obviously Shopify. Standard Shopify is not headless – when you use it, you use it to build **both your frontend and backend**. In other words, they take care of the backend (orders, products, etc.) but you don't have full creative control over your store's frontend; instead, you need to use their native website builder.

Shopify stores are organized as a series of pages:

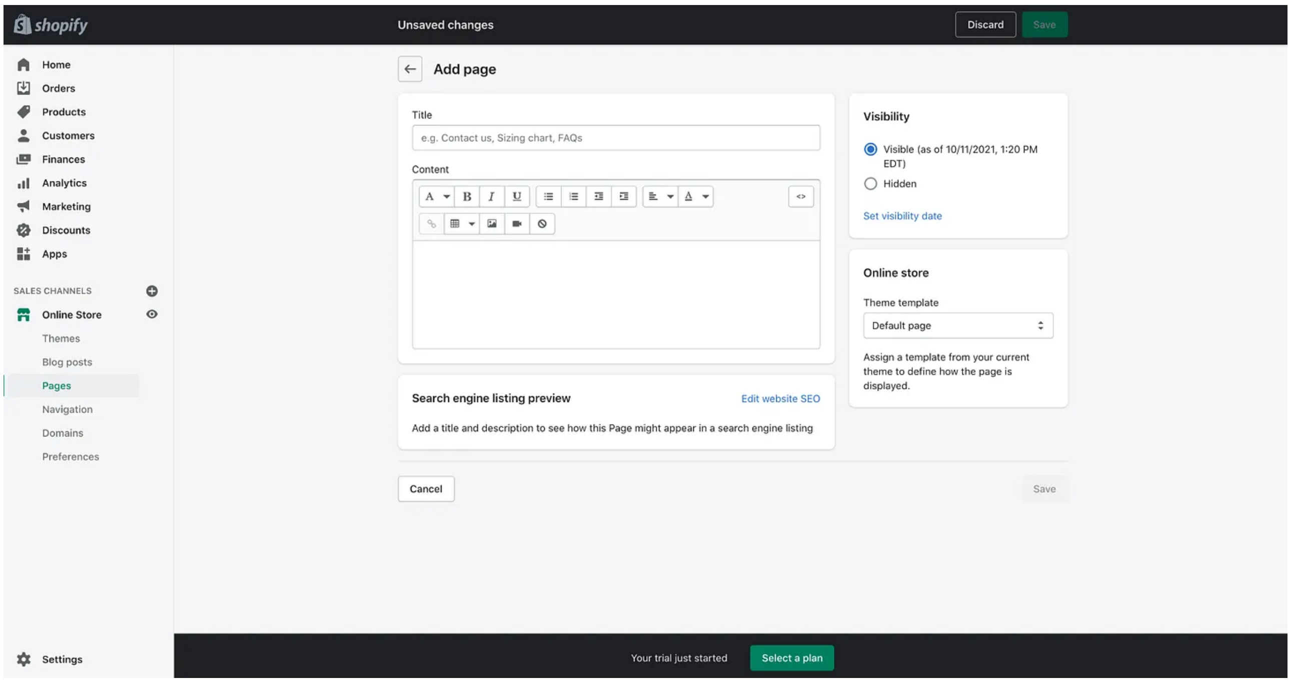


Online store is password protected
Only visitors with a password can access your online store.

Showing 1 Page
Contact
Just now

Learn more about pages

When you create a page, you get a limited set of customizable attributes:



e.g. Contact us, Sizing chart, FAQs

Visible (as of 10/11/2021, 1:20 PM EDT)

Default page

Assign a template from your current theme to define how the page is displayed.

Search engine listing preview
Add a title and description to see how this Page might appear in a search engine listing

When you're just trying to get started – or even more pressingly, if you don't have developers on staff – this is a godsend. But if you want to really customize how your store looks and feels – which is increasingly important – you're kind of stuck. Shopify Plus can run headless, but it's a lot more expensive and meant mostly for enterprises [1].

So in summary: headless ecommerce is about **flexibility**. Using Shopify or means you're locked into their site builder and styling, while a headless solution lets you build your frontend – and beyond that, infinite customizability in backend details – in the way you want to.

This post is sponsored by Swell, a headless e-commerce backend for your storefront. Swell makes it easy to set up storefronts that scale: you get a pre-built set of data models and APIs, along with a slick management console and model explorer.

swell

---

1 Shopify has been working on something internally called Hydrogen (which they released a very early-looking version of recently).

---

7 Comments

Write a comment...

Brendan Keeler Writes Health API Guy Nov 3, 2021

"An SDK is a series of APIs grouped together for some unified purpose."

I don't know that I agree here. Typically the SDK is the set of tools that play with the API - the code snippets or autogenerated packages in various languages, I thought.

♡ 2 Reply Gift a subscription Collapse •••

1 reply

machmember Nov 3, 2021

BigCommerce is in the MACH alliance, the industry standard for headless ecommerce platforms so it's probably worth removing it from this

♡ 1 Reply Gift a subscription Collapse •••

4 replies by Justin and others

5 more comments...