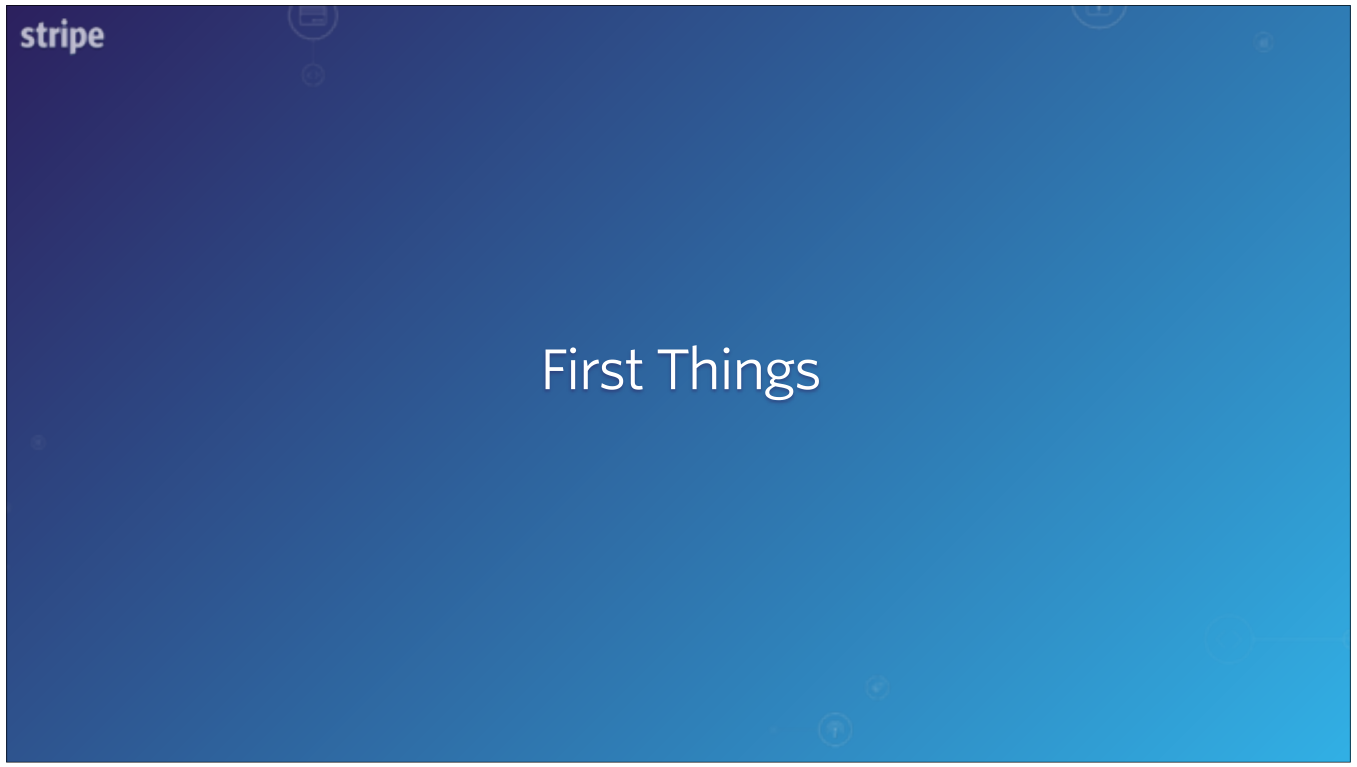
a stripe talk

Writing about products

Andreas Fuchs, @antifuchs
Krithika Muthukumar, @krithix



stripe

First Things

Hi & welcome!

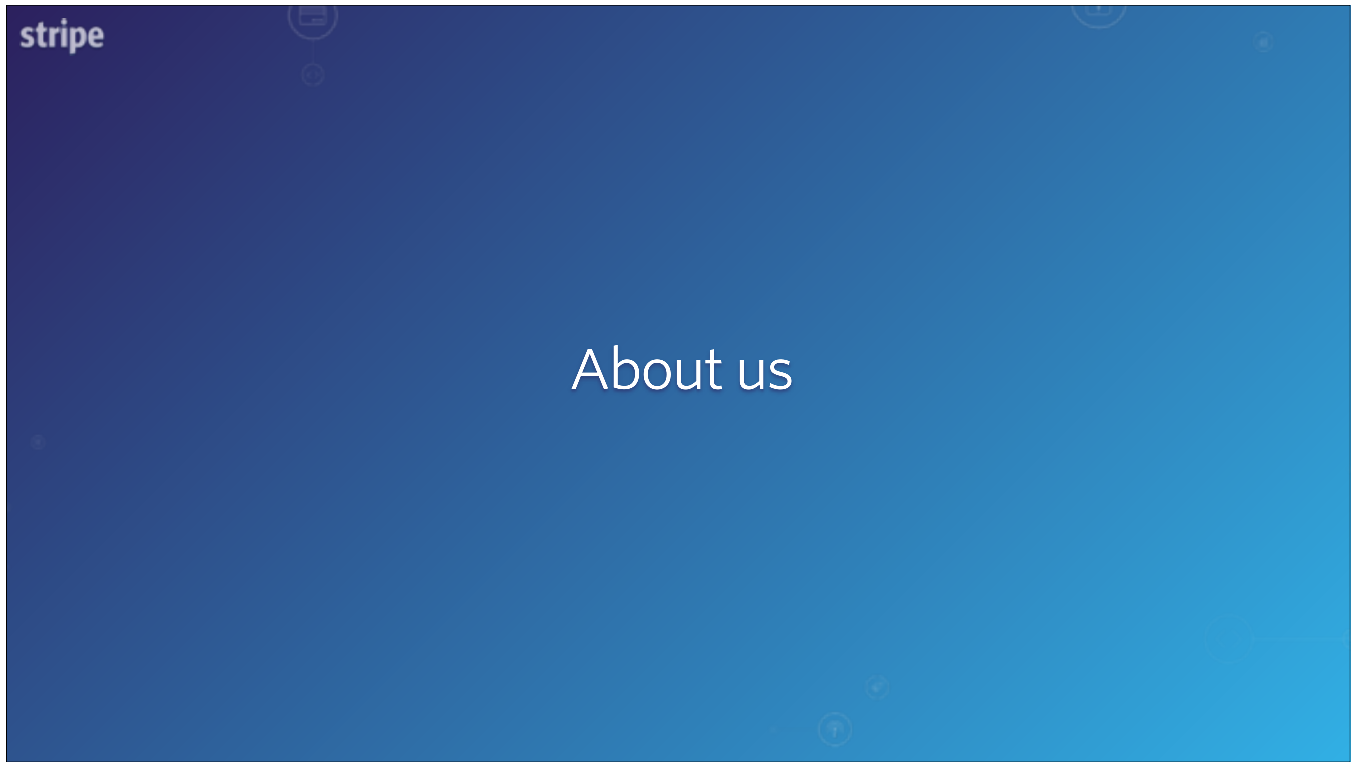
In this workshop, I'll be talking about how we make documentation for people at Stripe, and you will do a few writing exercises yourselves.

I'll be asking you all some audience questions - feel free to shout out suggestions & I'll do my best to write them down on the flipboard pages here.

I will not ask you to read out publicly anything you write here. Any reviews of what you've written will happen between you and one or two of your peers.

If you have any questions, raise your hand and ask! We have an audience mic.

Let's begin.


stripe

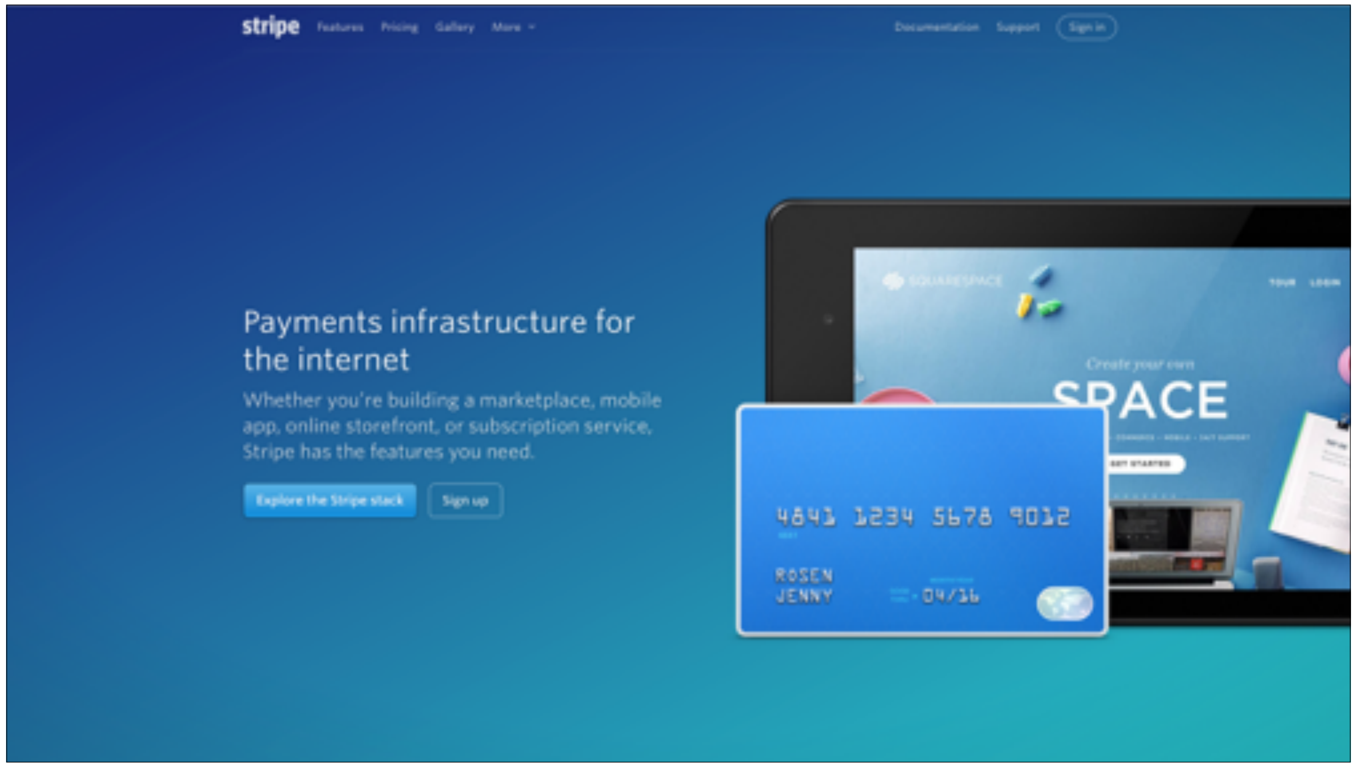
About us

I'm Andreas (hah)

Work as an engineer. I mostly write for developers internally.

ESL speaker - German native speaker, started learning English 24 years ago.

This is Krithika, she is responsible for most of the content on our website. We'll be your hosts today!



Payments infrastructure for the internet

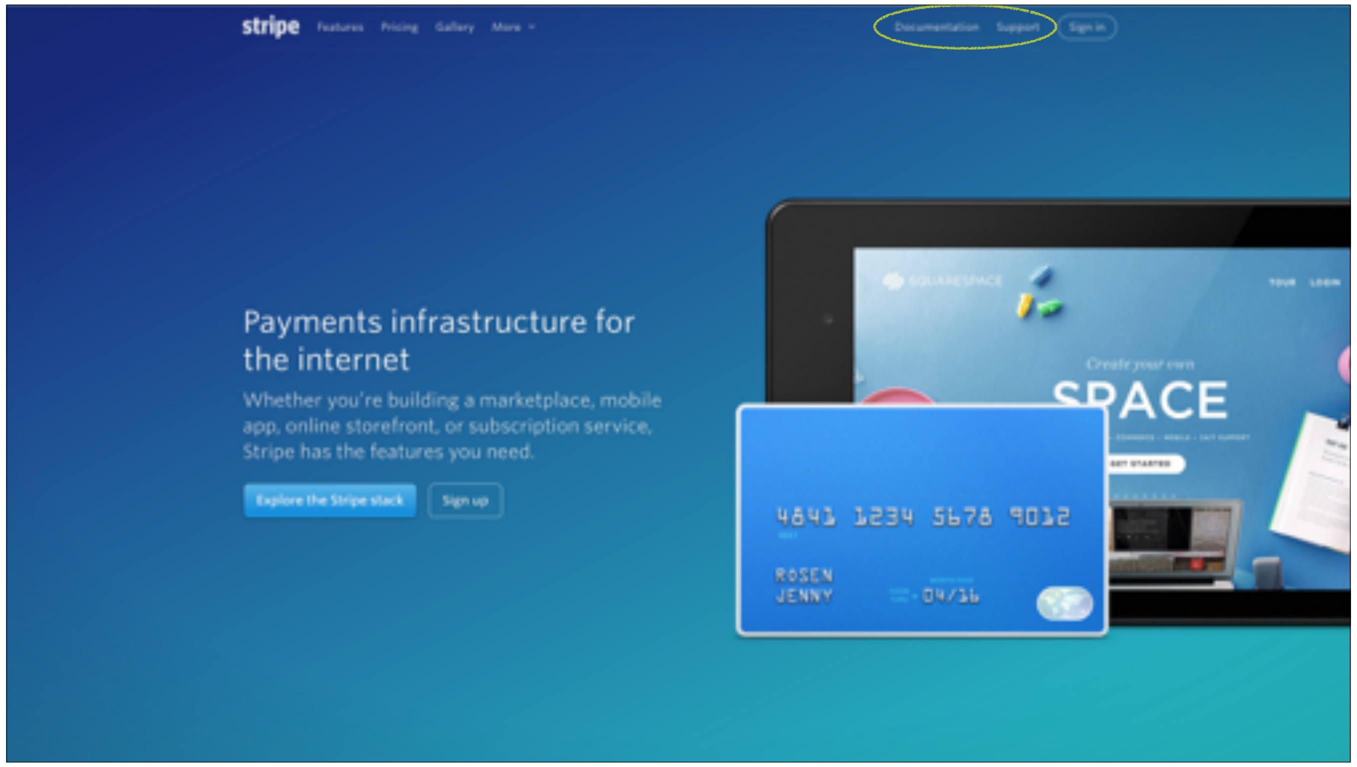
Whether you're building a marketplace, mobile app, online storefront, or subscription service, Stripe has the features you need.

And this is Stripe - our product allows people in many countries around the world to accept payments.

Payments are an incredibly complex topic. There's a lot to say about them, and our product hides a lot of that complexity via abstractions.

But then, we also have a very complex product! So people need to know what they can use, and how to use it.

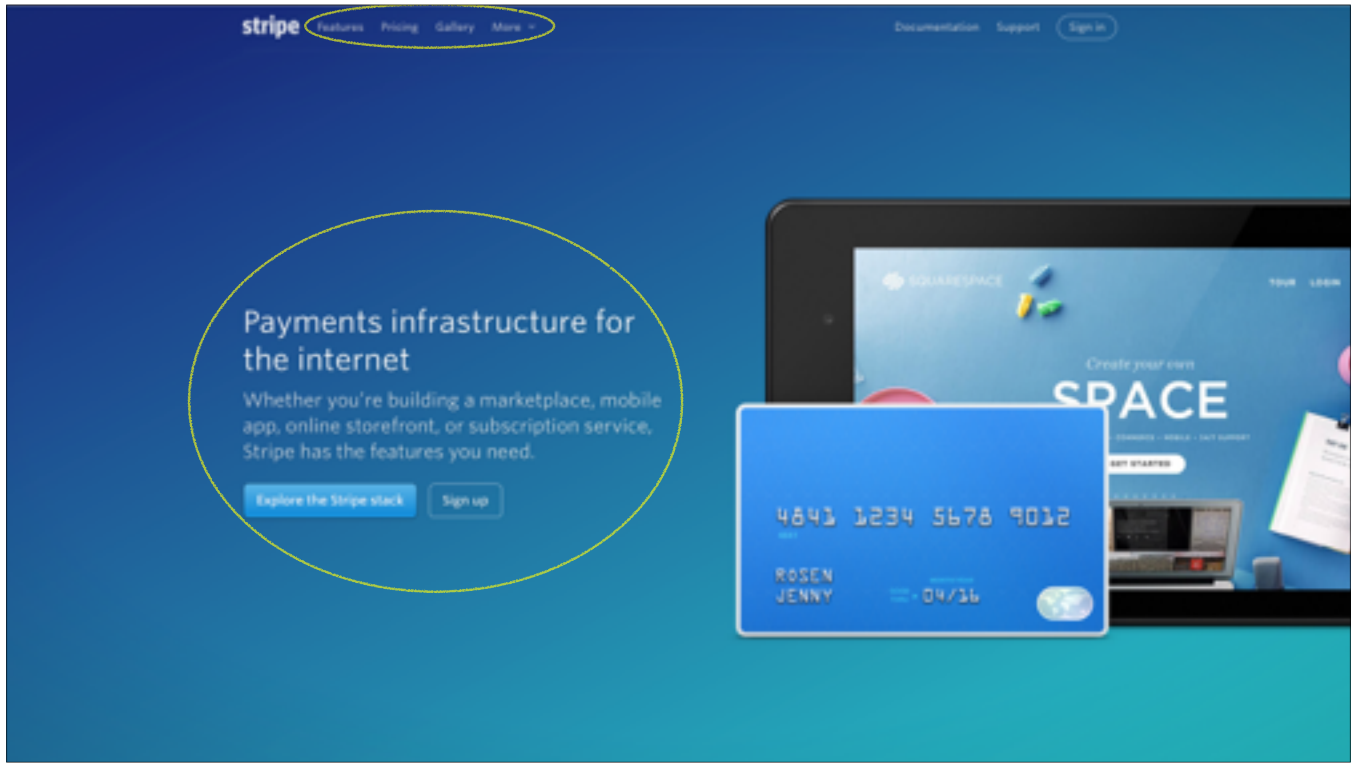
That means we have to have some way to tell them what all we can do, and how to use our product safely!



Payments infrastructure for the internet

Whether you're building a marketplace, mobile app, online storefront, or subscription service, Stripe has the features you need.

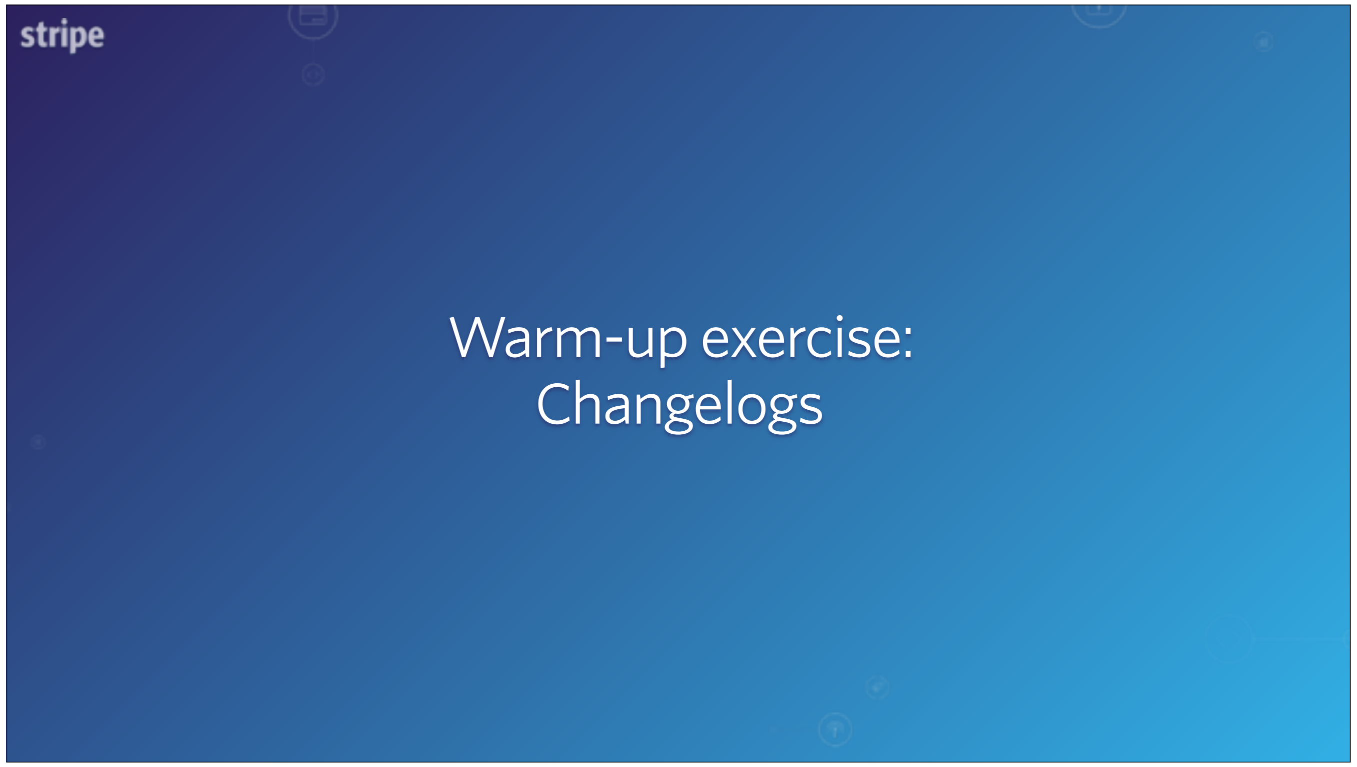
There's the places where we go into the technical details...



Payments infrastructure for the internet

Whether you're building a marketplace, mobile app, online storefront, or subscription service, Stripe has the features you need.

...and places where we explain how stuff works for a less technical audience.

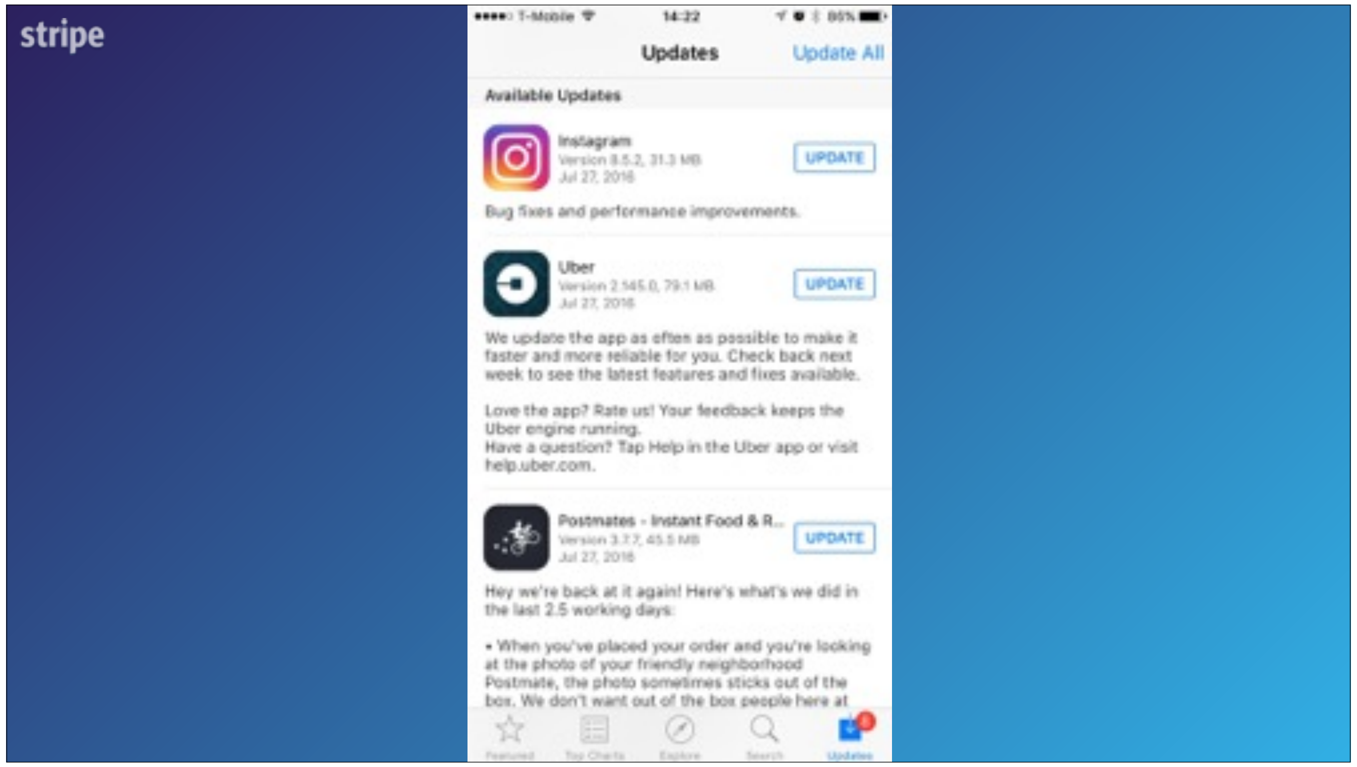

stripe

Warm-up exercise:
Changelogs

One of my favorite examples for pointing to effective technical writing is app changelogs.

[POLL] Who here has a smartphone?

[FOLLOWUP POLL] Who here reads the update messages in your phone's App Store?



Instagram
Version 8.5.2, 31.3 MB
Jul 27, 2016

Bug fixes and performance improvements.

Uber
Version 2.145.0, 79.1 MB
Jul 27, 2016

We update the app as often as possible to make it faster and more reliable for you. Check back next week to see the latest features and fixes available.

Love the app? Rate us! Your feedback keeps the Uber engine running.
Have a question? Tap Help in the Uber app or visit help.uber.com.

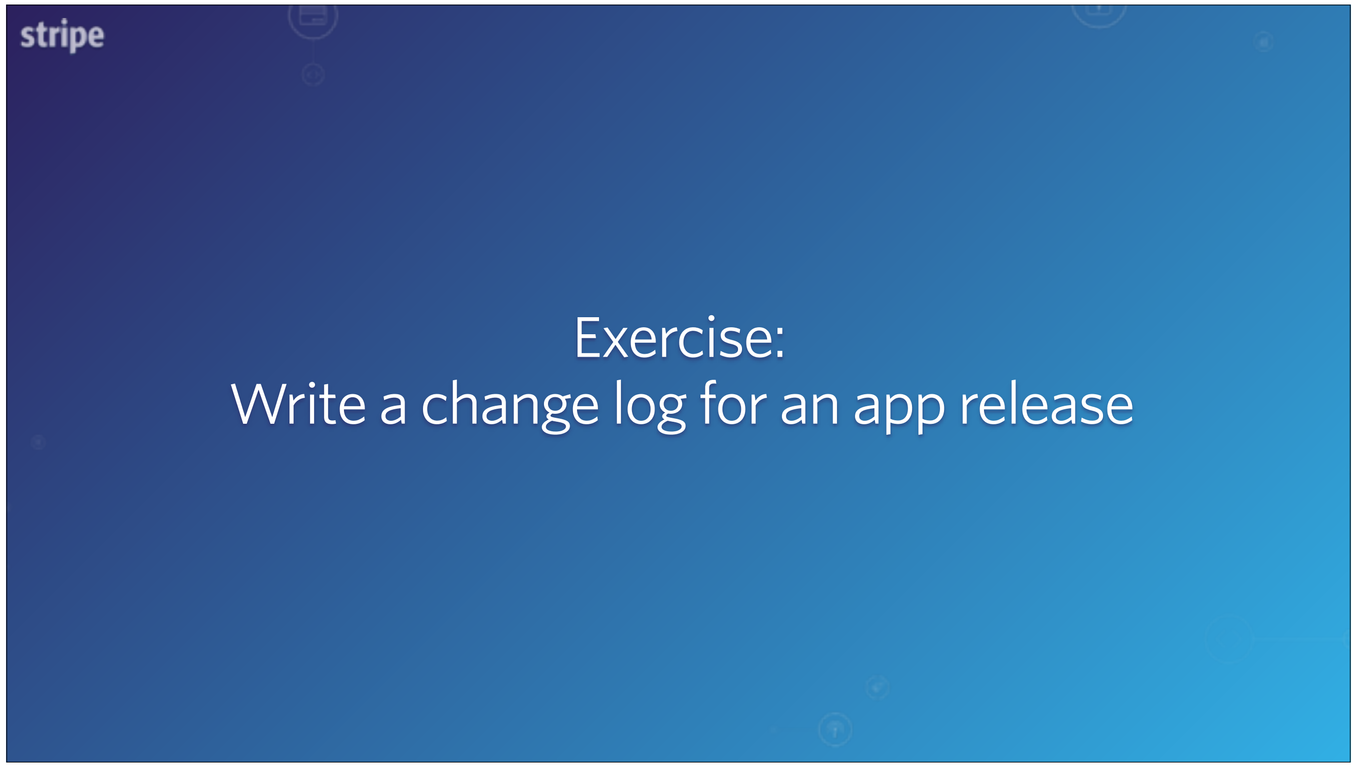
Postmates - Instant Food & R...
Version 3.7.7, 45.5 MB
Jul 27, 2016

Hey we're back at it again! Here's what's we did in the last 2.5 working days:

• When you've placed your order and you're looking at the photo of your friendly neighborhood Postmate, the photo sometimes sticks out of the box. We don't want out of the box people here at

Here are three apps that I can update on my phone (out of a bunch more).

[QUIZ] How do these three differ?



stripe

Exercise:
Write a change log for an app release

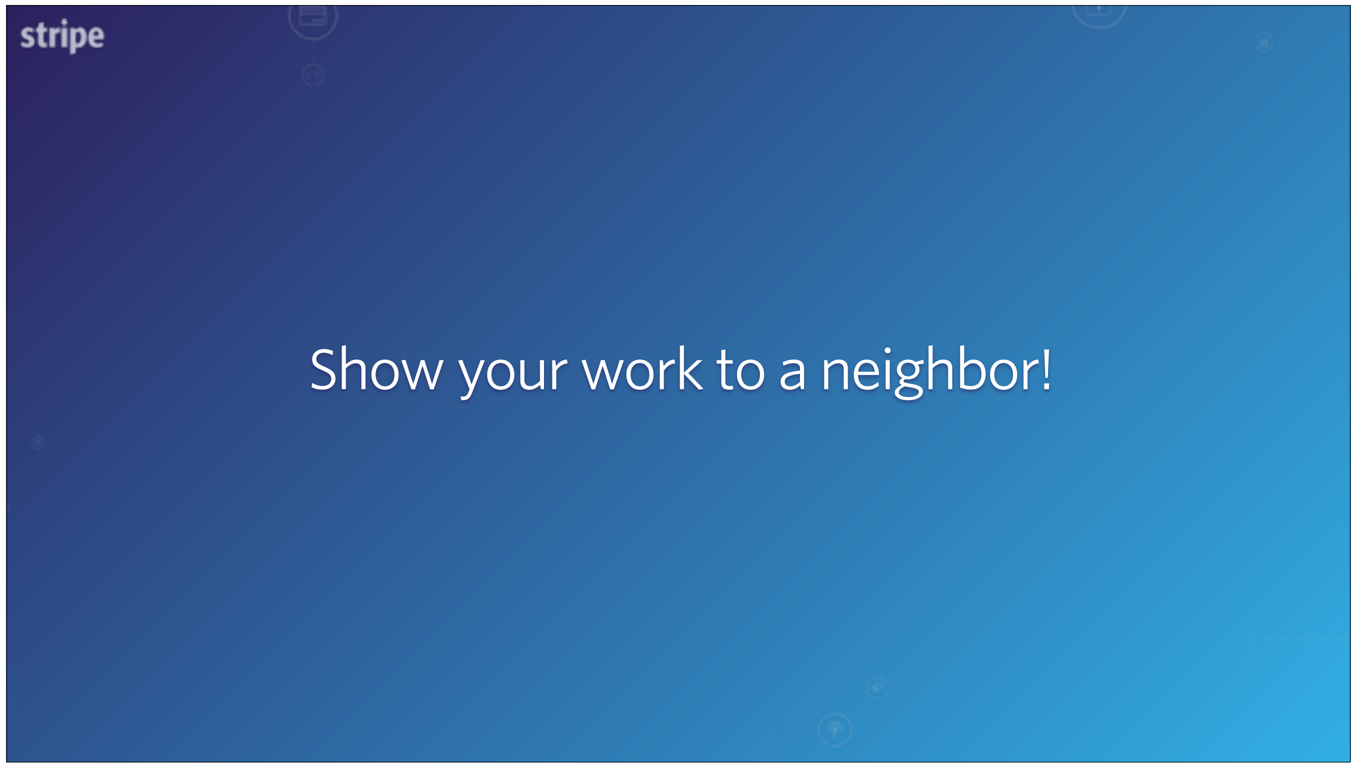
Come up with a cool change that you'd like to see, for an app you use often - that could be a web app or a mobile app, it doesn't matter!

Imagine this is your app.

Write a changelog entry for that feature. Make it informative, and make it as inviting for your app's users to upgrade as you can.

Giving you a writing assignment right now is a bit unfair, I know - we haven't even told you our secrets to great documentation yet. This is just to get the creative juices flowing, loosen your muscles a little.

You have 15 minutes to write, then swap with a neighbor to collect feedback for 5 minutes.

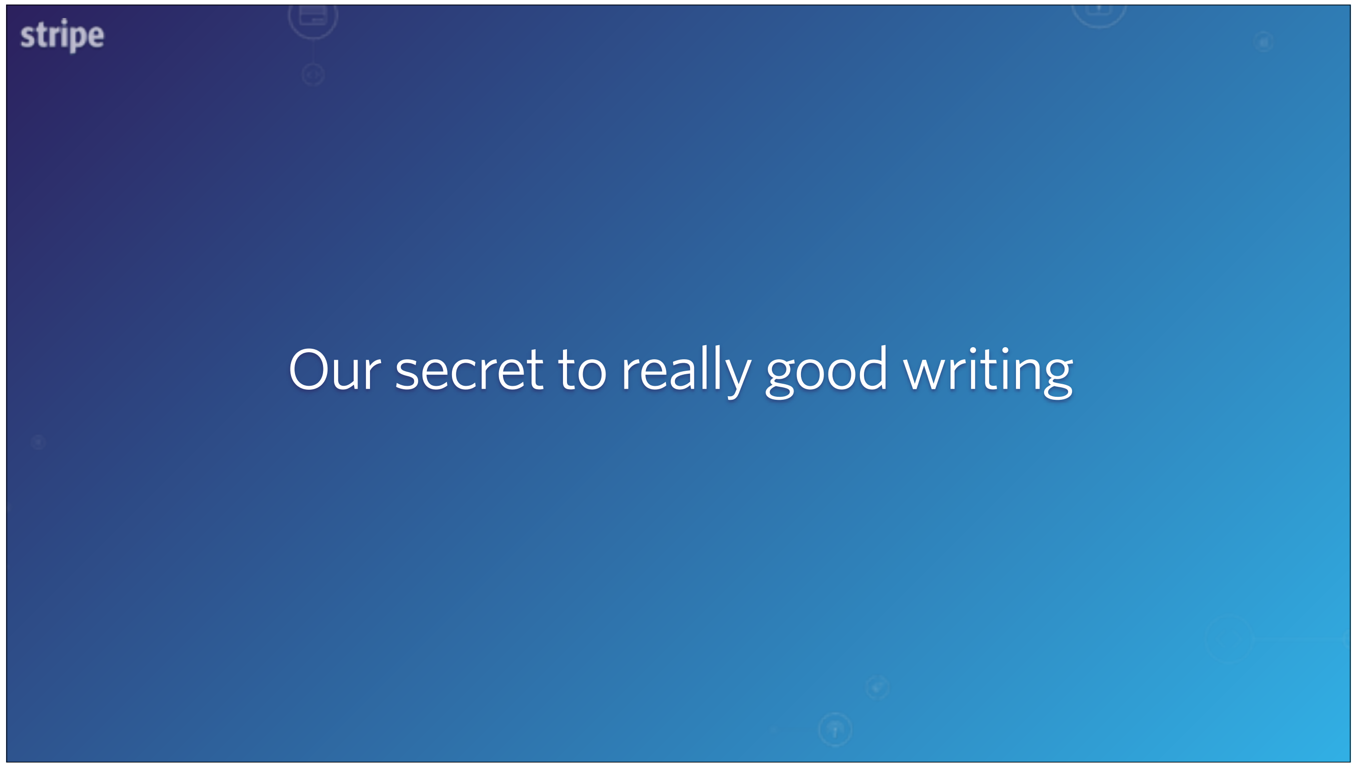


stripe

Show your work to a neighbor!

Time's up, now turn around and show what you wrote to your neighbor and read what they wrote! Talk about what you found difficult!

[QUIZ] What works well? What doesn't?

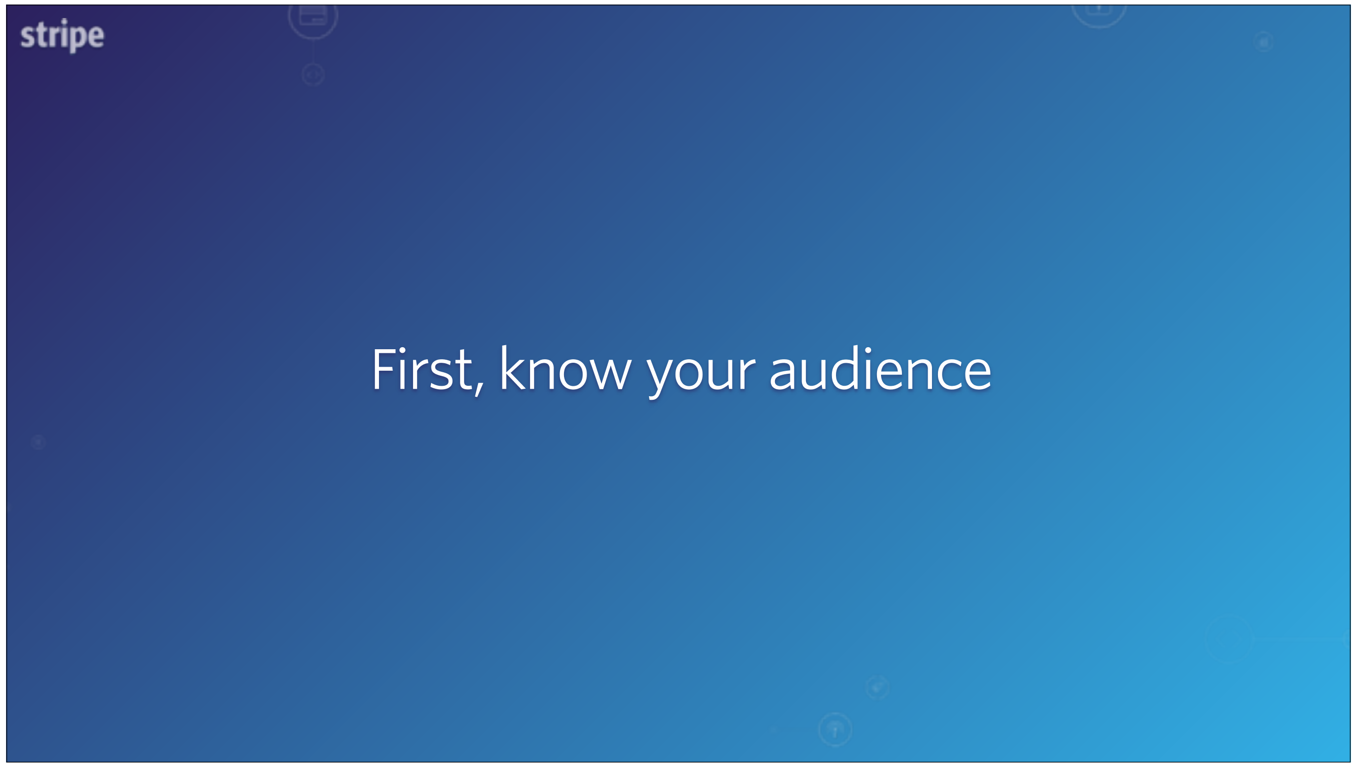


stripe

Our secret to really good writing

Before this exercise, I mentioned that I hadn't even given away all our secrets for making good documentation and blog posts.

Let's fix that. There are three things you need to know to write great content.



stripe

First, know your audience

This is the most important thing. You have to know:

Knowledge level

Interests

It determines your tone, and what you're even going to write about.

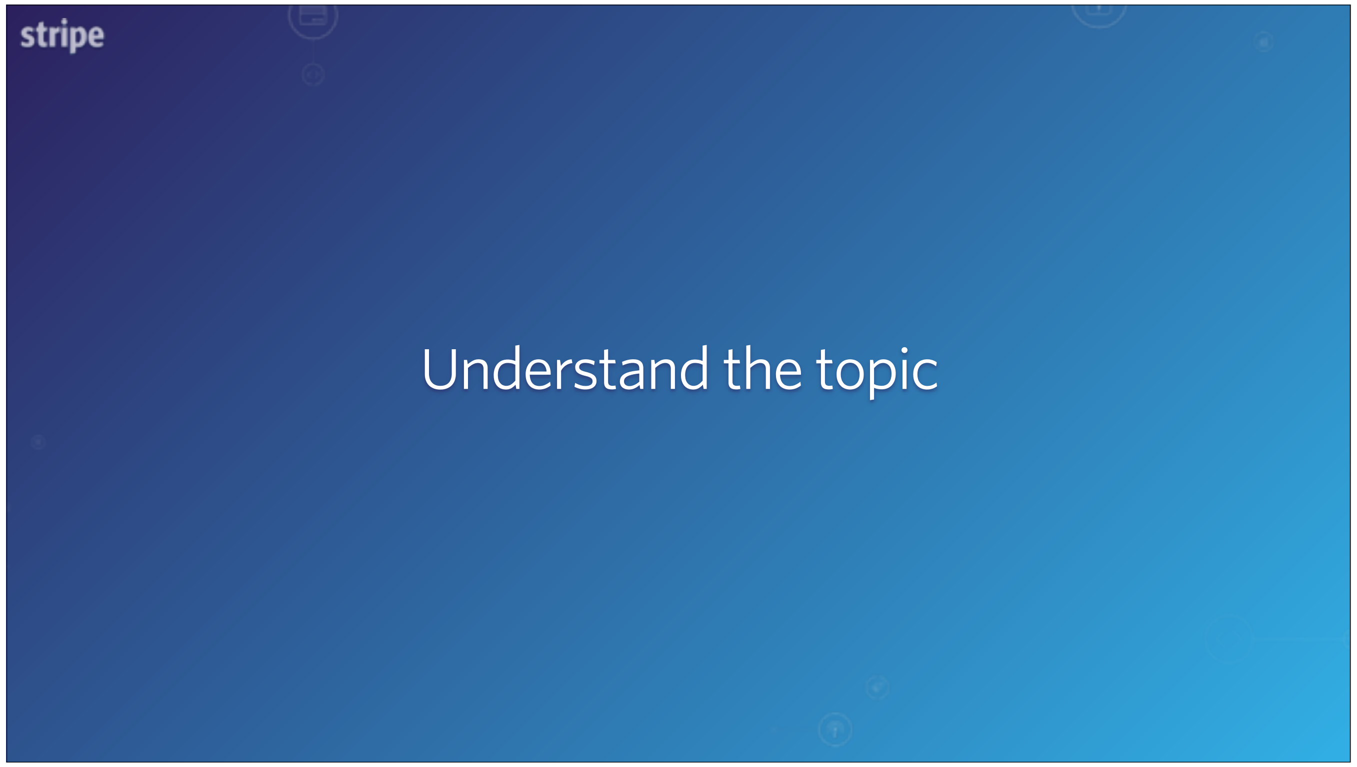
The Stripe approach:

Come up with a person you know who will benefit, and write to them.

To find your audience, talk to people: potential users, customers.

Taking our own advice:

[Show of hands] How many people here have written some form of documentation?



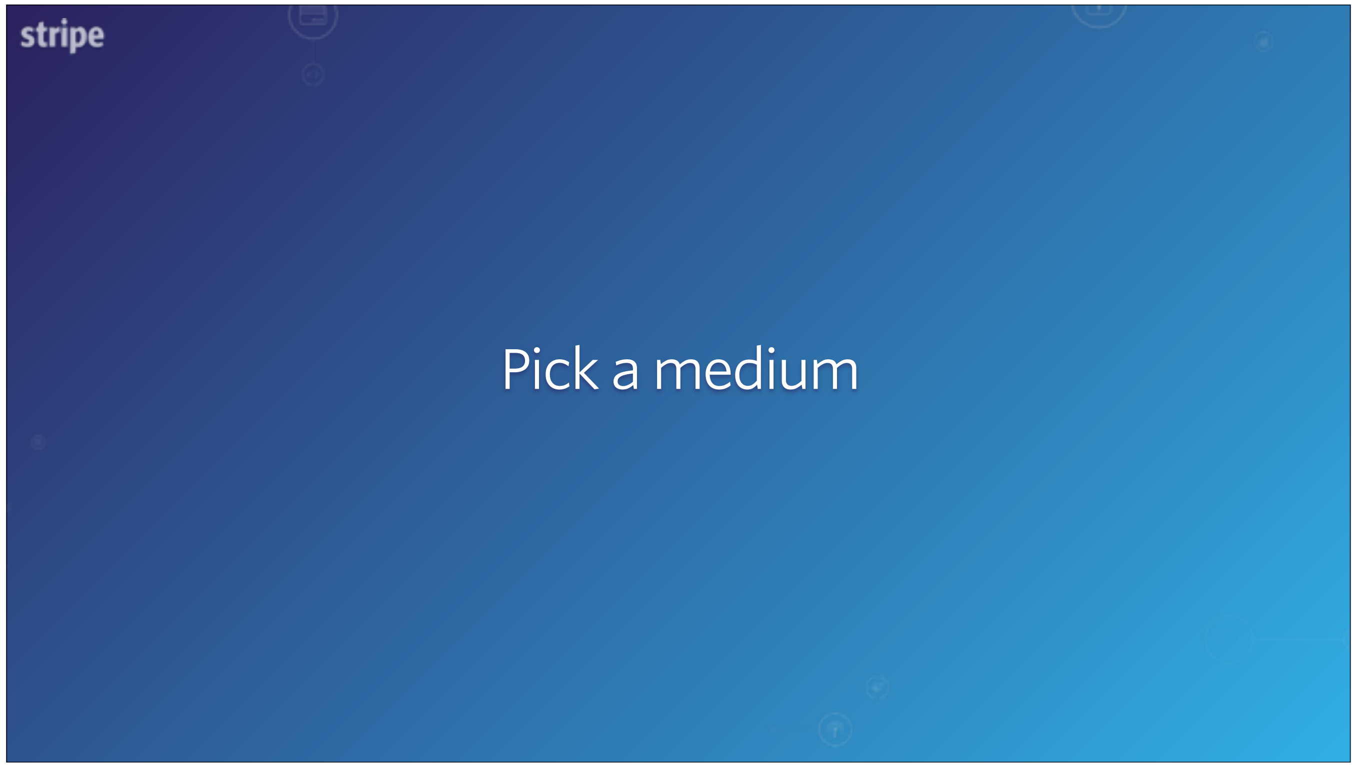
stripe

Understand the topic

The better you understand a thing, the better your writing can help others understand it

- Talk to the people making the product, or write documentation yourself
- Try out the product

Time: about 10% actually spent writing copy, the rest is research / talking.



Pick a medium

Different media are great for different things!

Blog posts are great for announcing new features, or for sharing things you learned

A website - talking to prospective users

API Documentation - a technical audience

[QUIZ] other examples of media?

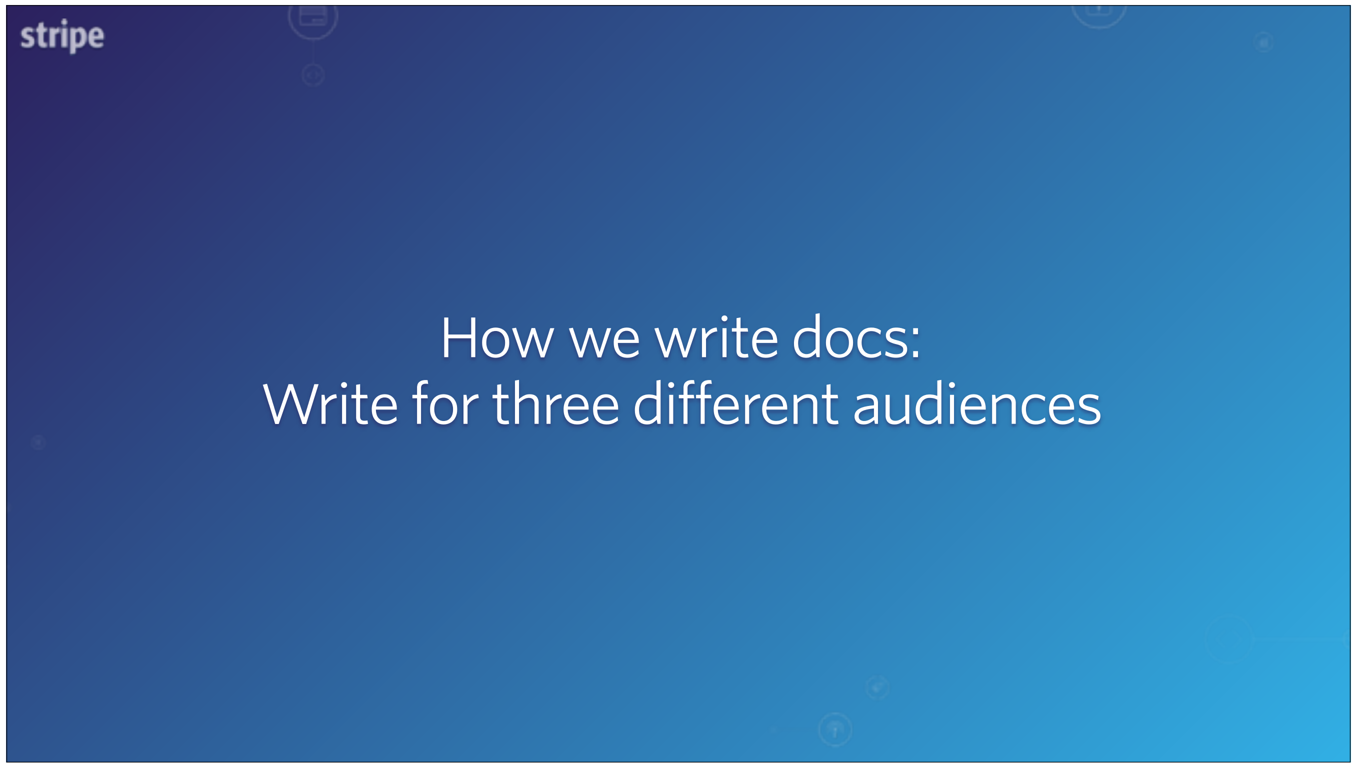
- Emails
- Video walkthroughs
- Talks

Mnemonic:
💳💰 A T M 💰💳

These three get pretty easy to remember as A, T, M. (I'm sorry about the emoji)

What's great about all these questions is that you can vary the answers and produce wildly different outcomes.

At Stripe, we use this to great effect! Let me demonstrate!



stripe

How we write docs: Write for three different audiences

At Stripe, we split our web site content up in three ways, one per slice of the audience we want to reach!

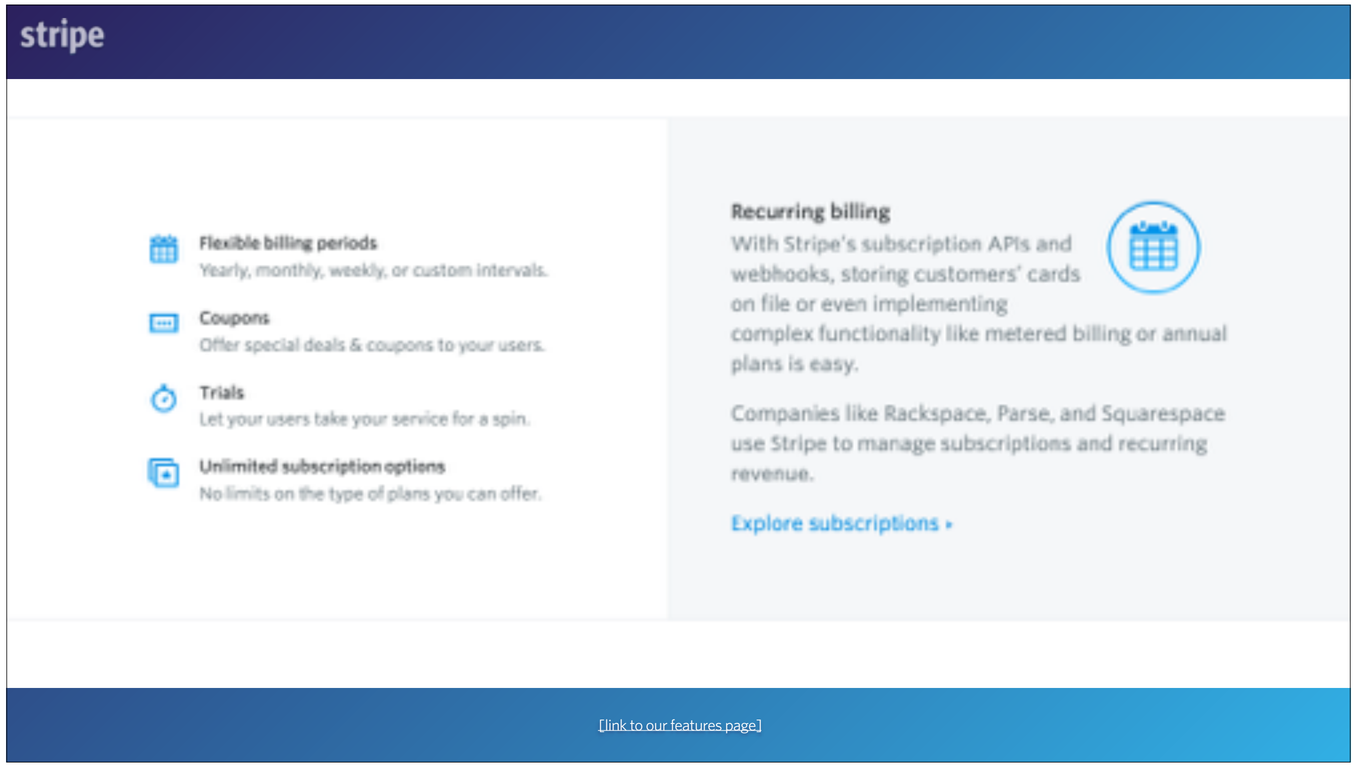
For illustrating this, we'll be looking at Stripe's subscription billing product a bit.

Subscriptions allow our user to ask a customer for their payment details once, and then charge them money for an ongoing service in regular intervals.

They're are notoriously difficult, and they always affect the relationship a stripe user has with their customer! Both must be aware of changes.

Edge cases: Expired cards, prorating, discounts

So, when different people look at our website, they want to know different things.



stripe

Flexible billing periods
Yearly, monthly, weekly, or custom intervals.

Coupons
Offer special deals & coupons to your users.

Trials
Let your users take your service for a spin.

Unlimited subscription options
No limits on the type of plans you can offer.

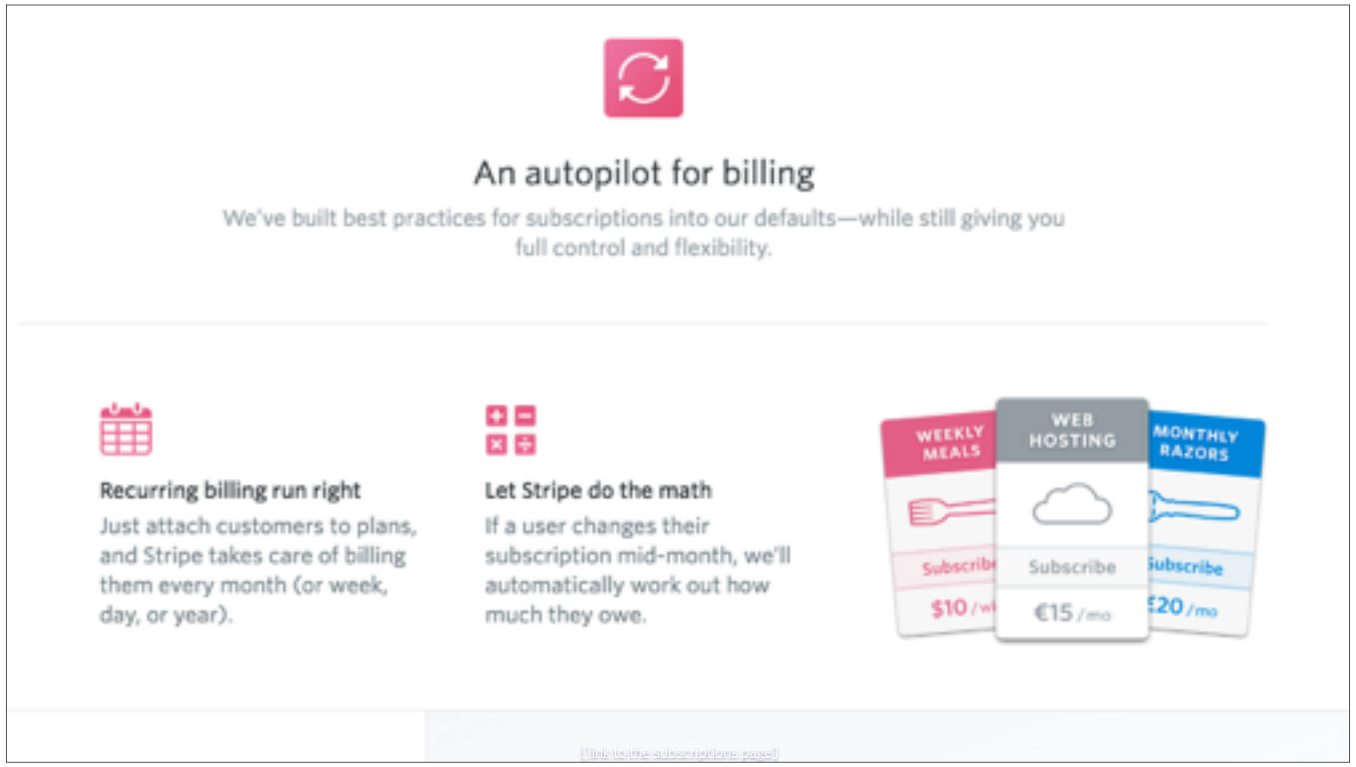
Recurring billing
With Stripe's subscription APIs and webhooks, storing customers' cards on file or even implementing complex functionality like metered billing or annual plans is easy.

Companies like Rackspace, Parse, and Squarespace use Stripe to manage subscriptions and recurring revenue.

Explore subscriptions ▸

[link to our features page]

A skimmable version on a dedicated page for somebody who is considering using the product and is comparing how it shapes up.


An autopilot for billing

We've built best practices for subscriptions into our defaults—while still giving you full control and flexibility.

Recurring billing run right

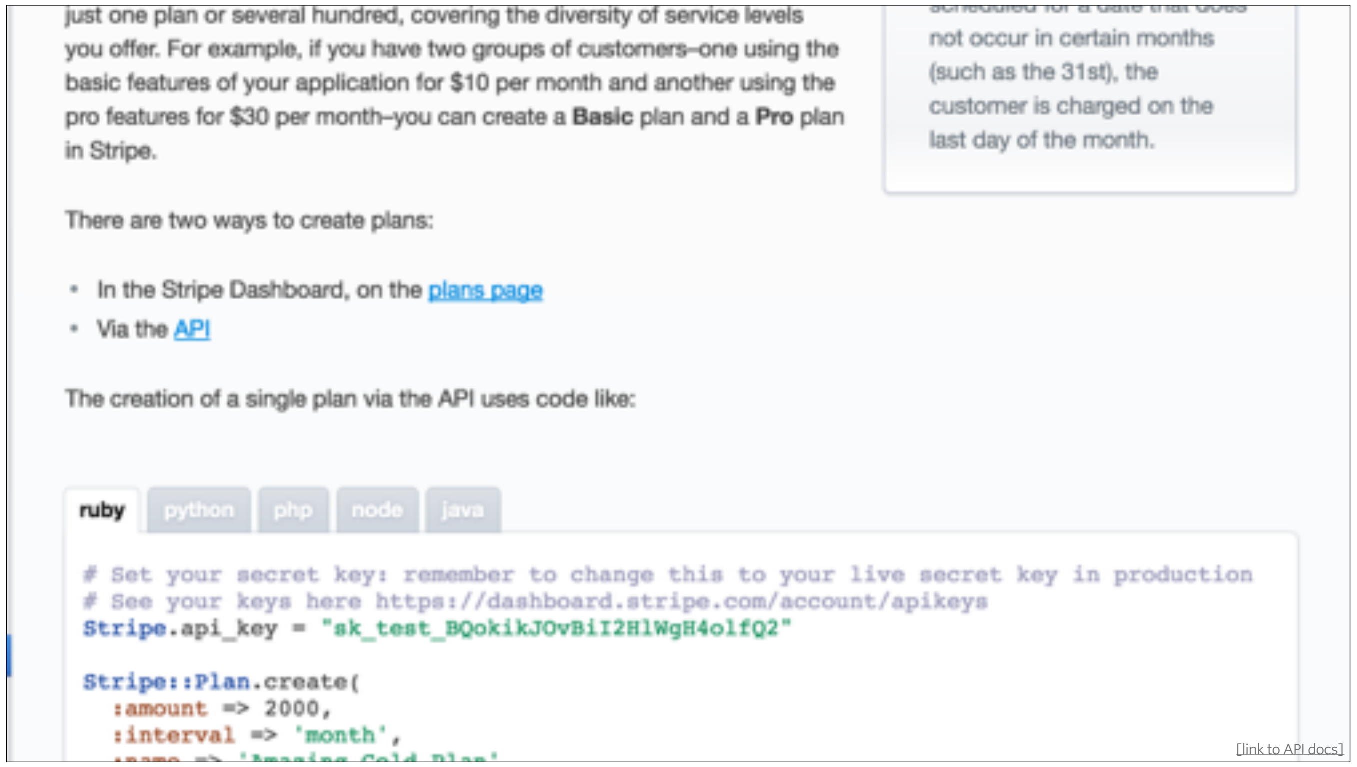
Just attach customers to plans, and Stripe takes care of billing them every month (or week, day, or year).

Let Stripe do the math

If a user changes their subscription mid-month, we'll automatically work out how much they owe.

[link to the subscriptions page]

A very high-level overview of a product, for a person who just heard about us.



just one plan or several hundred, covering the diversity of service levels you offer. For example, if you have two groups of customers–one using the basic features of your application for $10 per month and another using the pro features for $30 per month–you can create a **Basic** plan and a **Pro** plan in Stripe.

There are two ways to create plans:

- In the Stripe Dashboard, on the plans page
- Via the API

The creation of a single plan via the API uses code like:

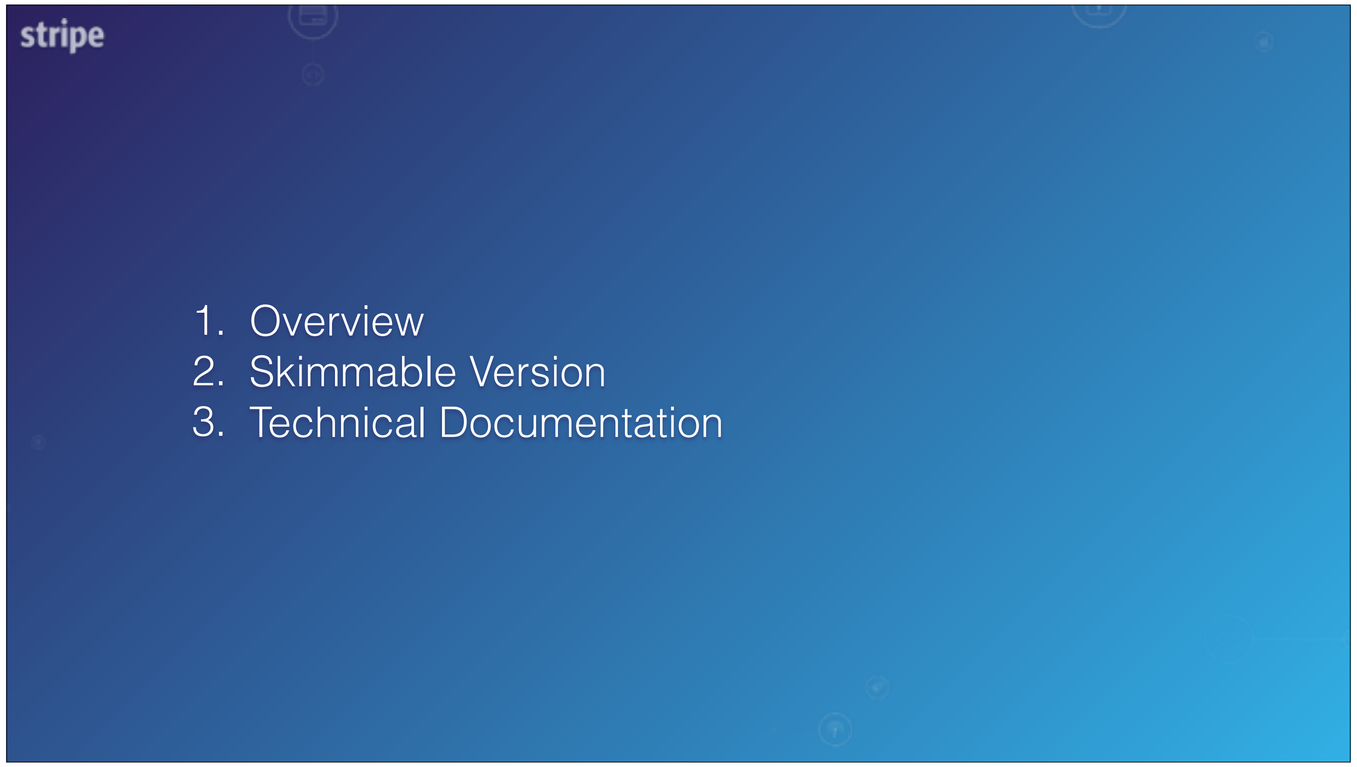
scheduled for a date that does not occur in certain months (such as the 31st), the customer is charged on the last day of the month.

```ruby
# Set your secret key: remember to change this to your live secret key in production
# See your keys here https://dashboard.stripe.com/account/apikeys
Stripe.api_key = "sk_test_BQokikJOvBiI2HlWgH4olfQ2"

Stripe::Plan.create(
  :amount => 2000,
  :interval => 'month',
  :name => 'Amazing Gold Plan'
```

[link to API docs]

And last, a set of pages dedicated to very in-depth technical documentation: For a person tasked with integrating it.



stripe

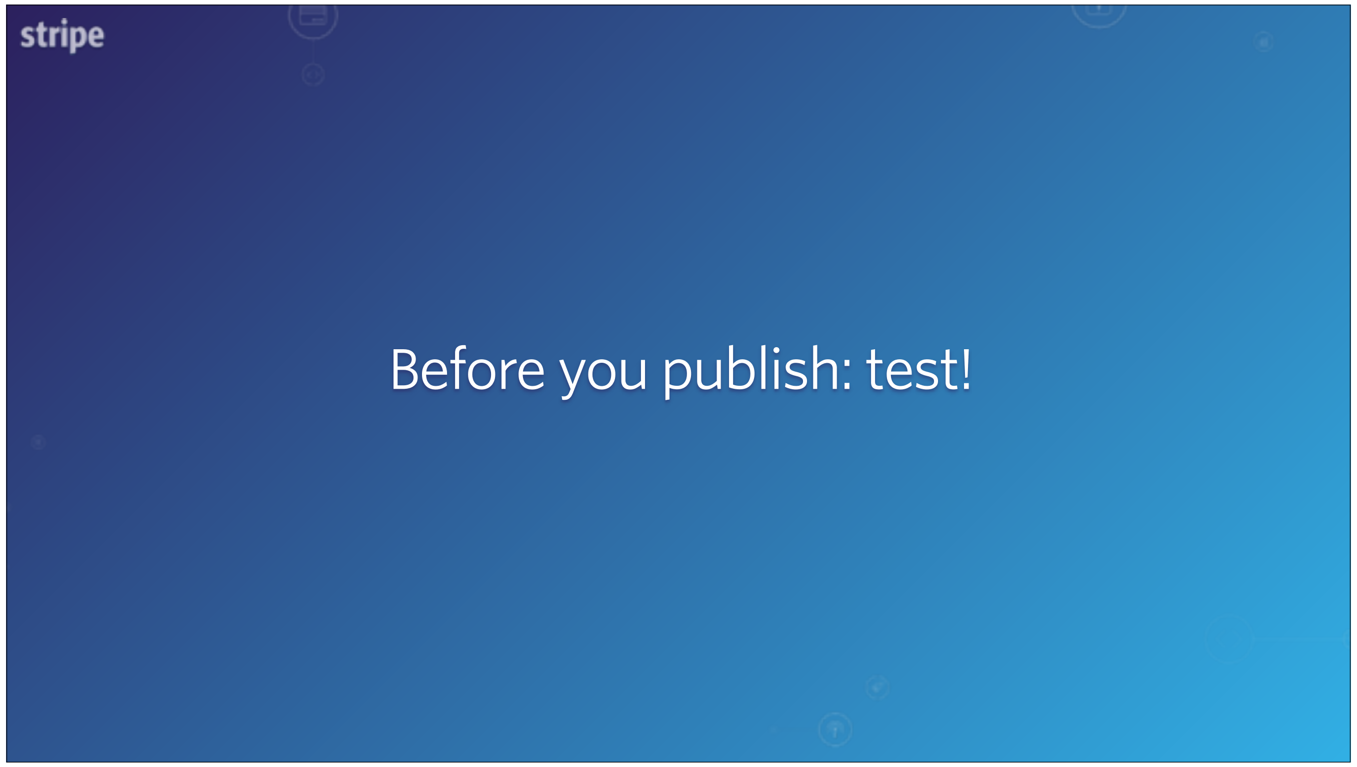
1. Overview
2. Skimmable Version
3. Technical Documentation

[QUIZ]: What sorts of questions do you expect each of these to answer?

Overview: what's the business case?

Skimmable version: What can it do?

Technical documentation: What do I have to do to make it work?

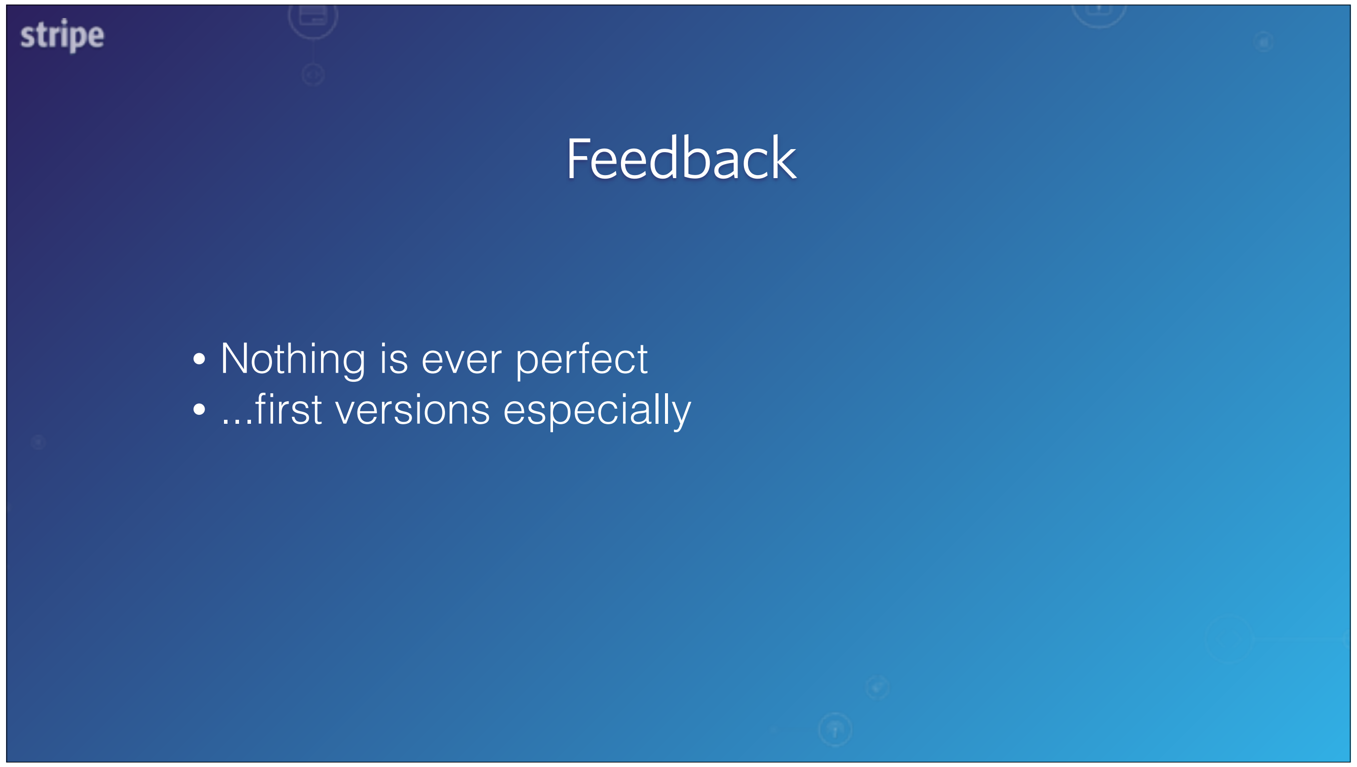

stripe

Before you publish: test!

We treat our documentation like a product itself:

Before we launch any docs or website changes, it goes through a beta phase.

How does that work? Find 5-6 people and have them read & give feedback.



stripe

Feedback

- Nothing is ever perfect
- ...first versions especially

The first version of everything will always have problems, it's a fact of life. In fact, our writing advice documentation states:

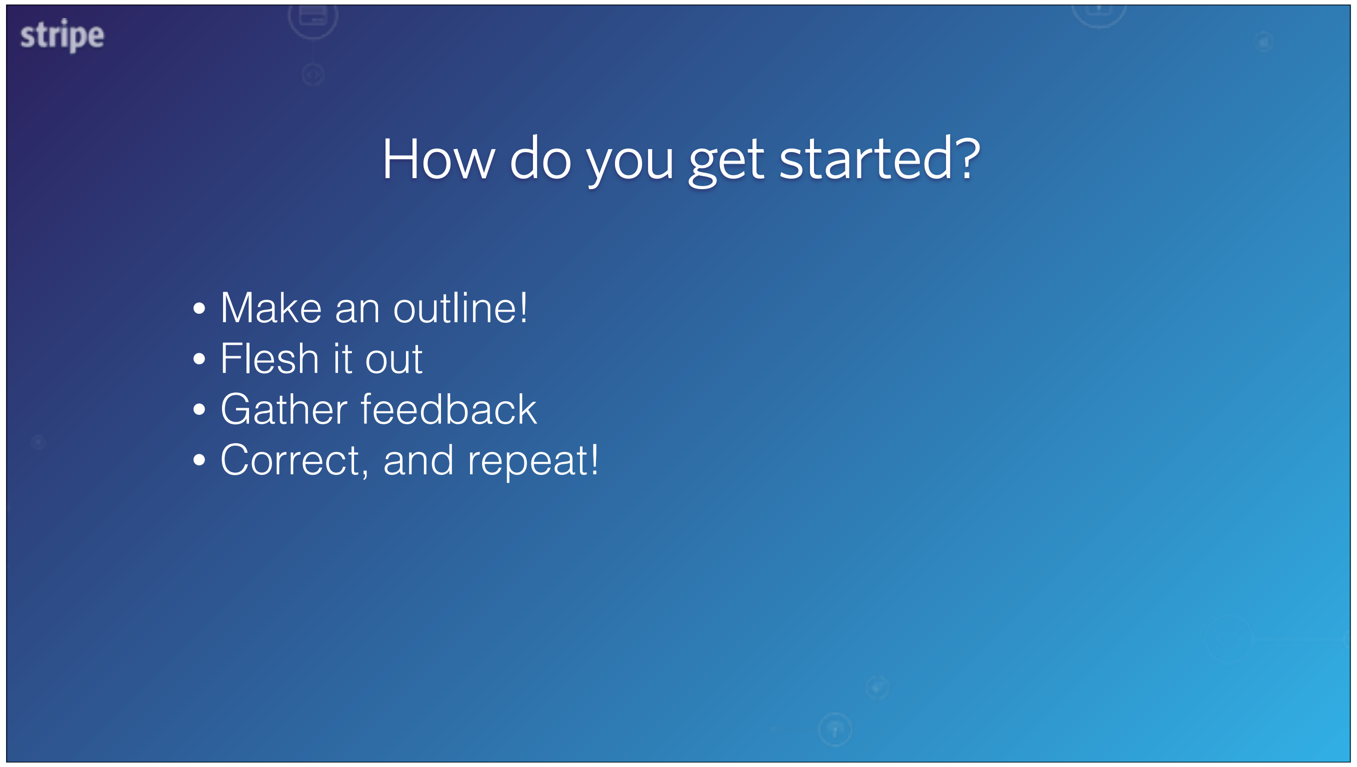
Your initial draft exists only so you have something to rewrite: Good writing is rewriting.

Getting others to look at your text really helps you find things that need to be improved, or ones that can be cut out entirely.

Some question we found very useful when asking for feedback after a person read a piece of documentation:

What did you learn from this?

What was unclear?

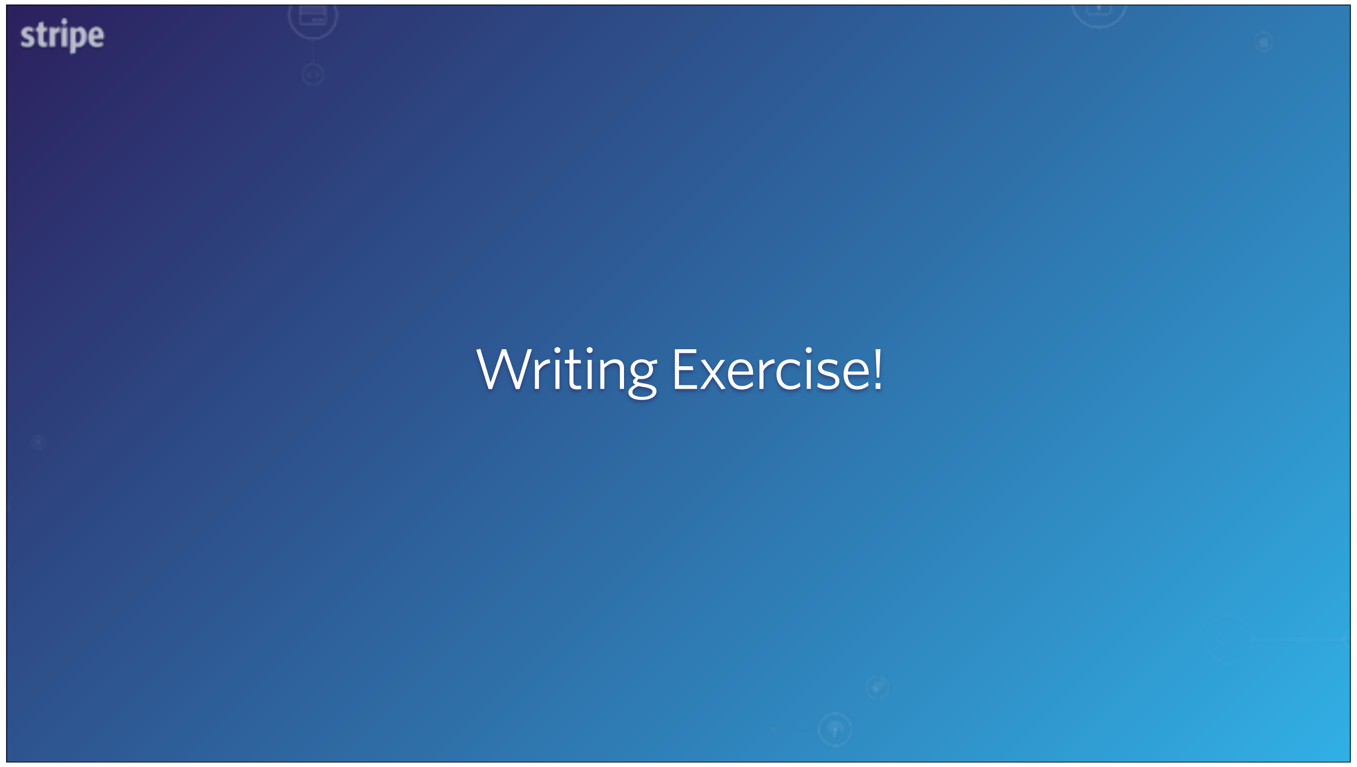


How do you get started?

- Make an outline!
- Flesh it out
- Gather feedback
- Correct, and repeat!

An empty page could become anything, so how do you get from there to documentation?

The thing that helps me the most for getting off the ground is to make an outline. Brainstorm and write down all the things that come to mind when thinking about the topic. These can be the headlines you start from!



stripe

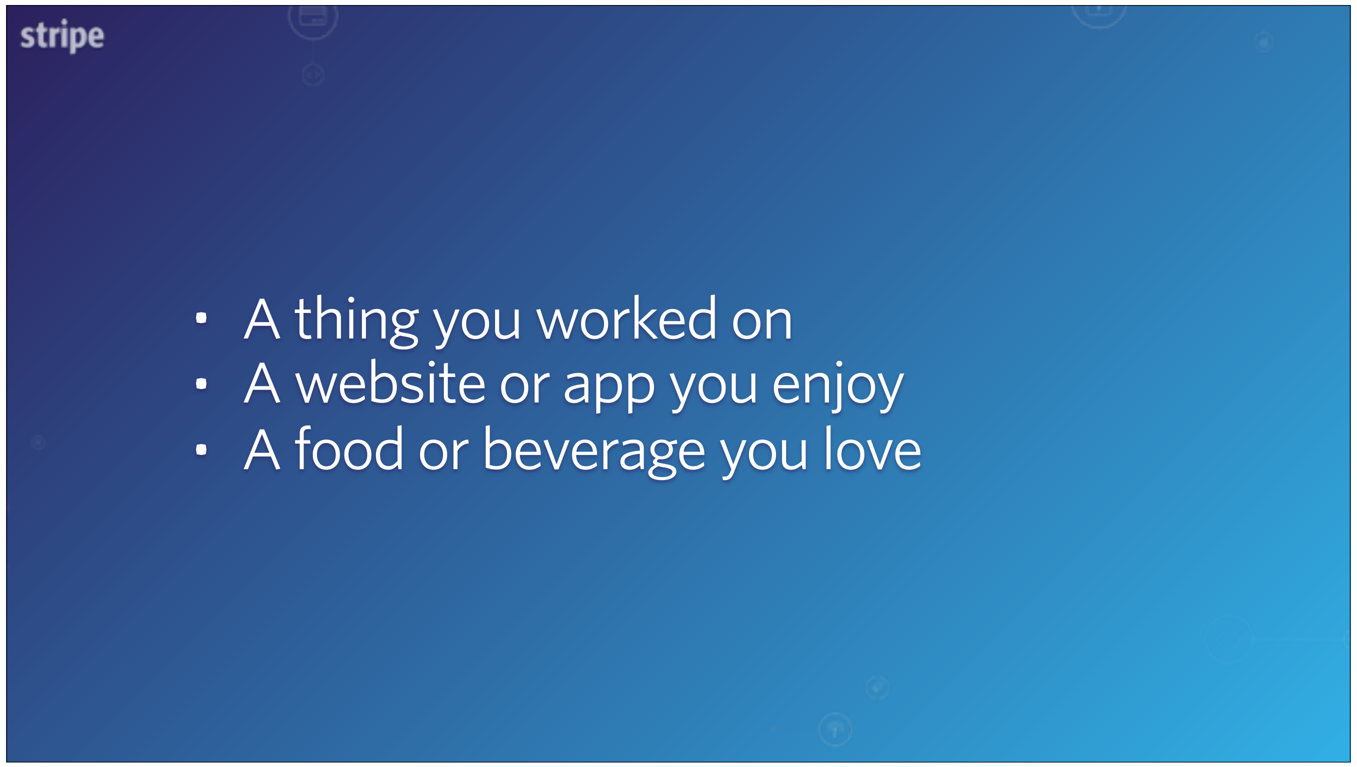
Writing Exercise!

POLL: I hear there was a hackathon recently, how many of you have worked on something there?

I want you to write an outline for an overview page for the thing you built. If you didn't build a thing or you're not comfortable writing about it, I have some more prompts on the next slide.

Write up an intro the headlines making up the key parts that somebody else should know about! and if you have time left, write some content to go with the headlines.

You have 20 minutes to write.
Then find your neighbor again, read their text, and give them feedback - 10minutes.

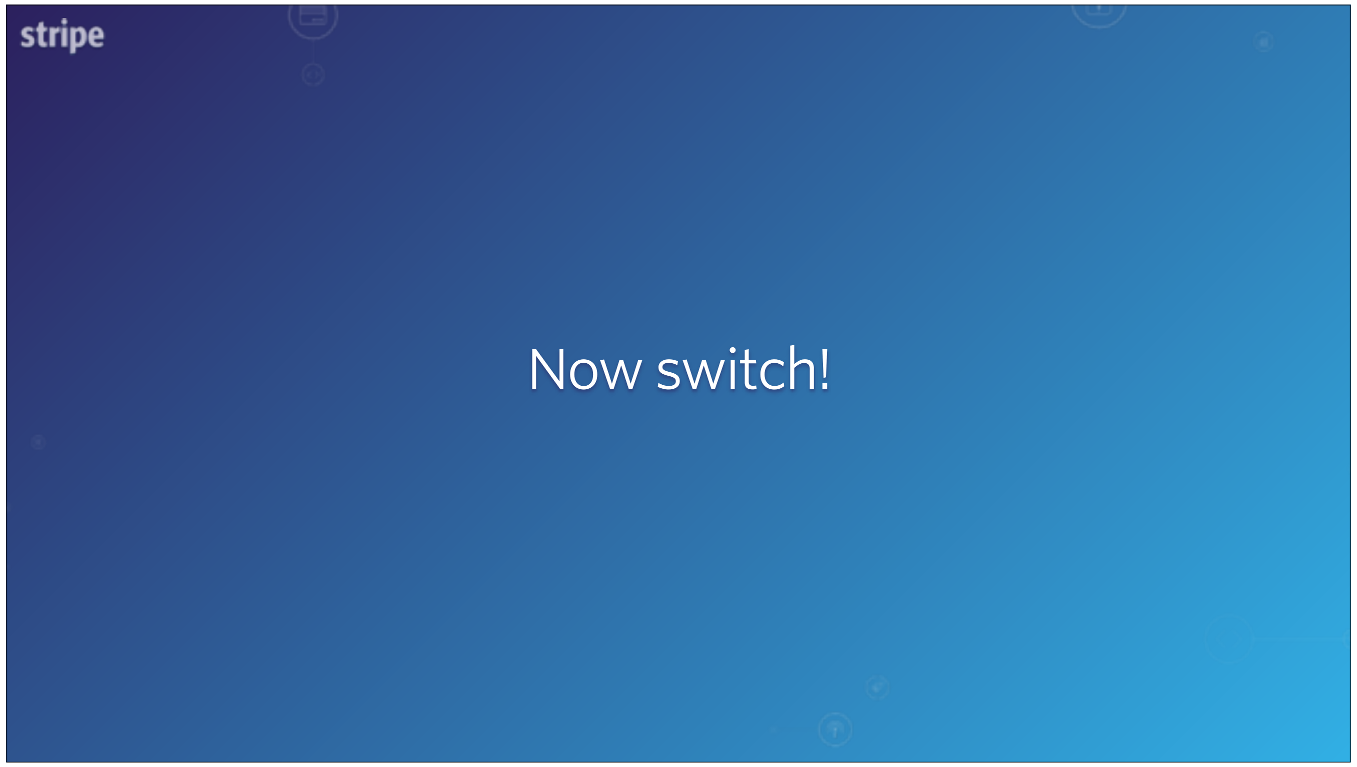


stripe

- A thing you worked on
- A website or app you enjoy
- A food or beverage you love

All of these are valid and great things to write about, and each will exercise your creative muscles in a slightly different way.

I'll be here, in front. Feel free to ask me if you need any help.



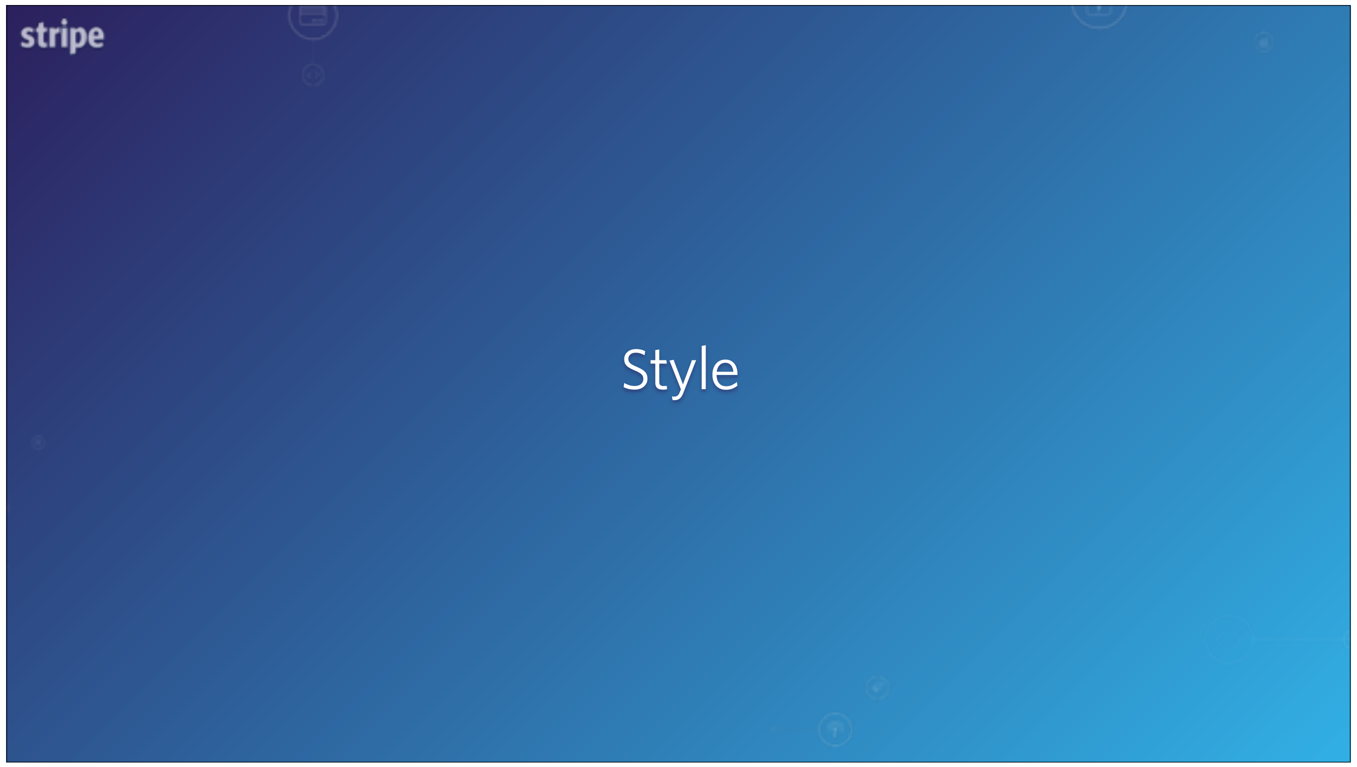
stripe

Now switch!

Now it's time to switch & read what your neighbor wrote.

When you read their text, try to get a clear picture of what your neighbor is describing.

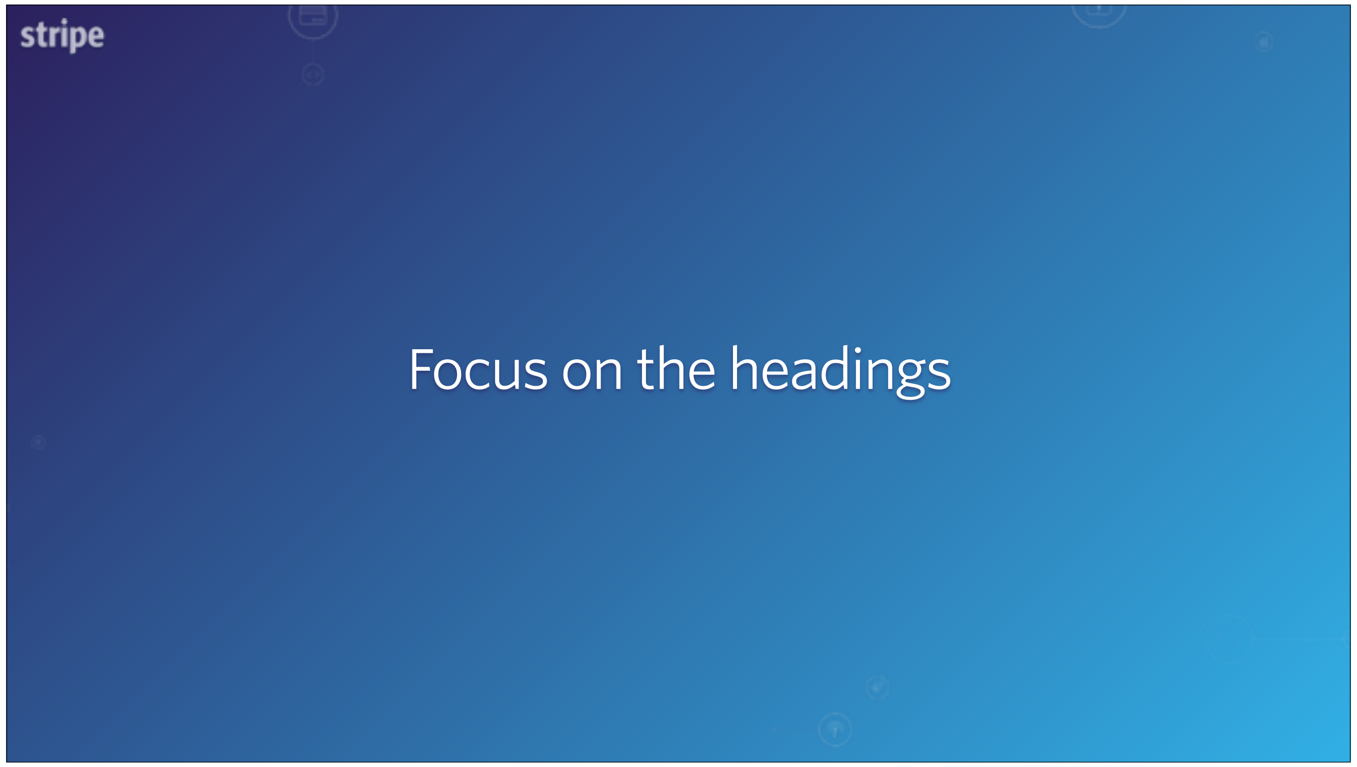
Tell them the things that are very compelling, and what you found confusing!


stripe

Style

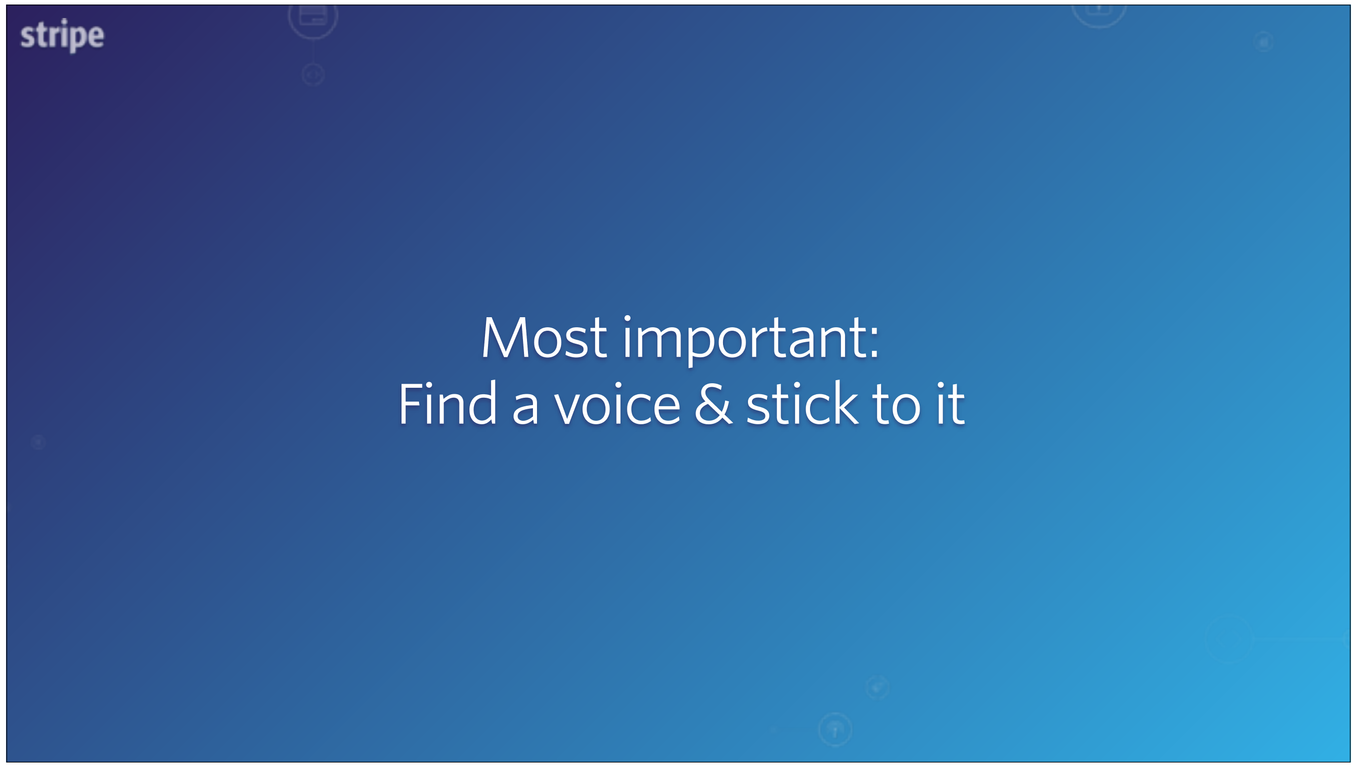
All this advice was about structuring content so far, we didn't touch on style at all!

What follows are some pretty common pieces of advice from our style guide:


stripe

Focus on the headings

The headings will be the bulk of what your readers will take away from your text, so you should polish these the most, and first.


stripe

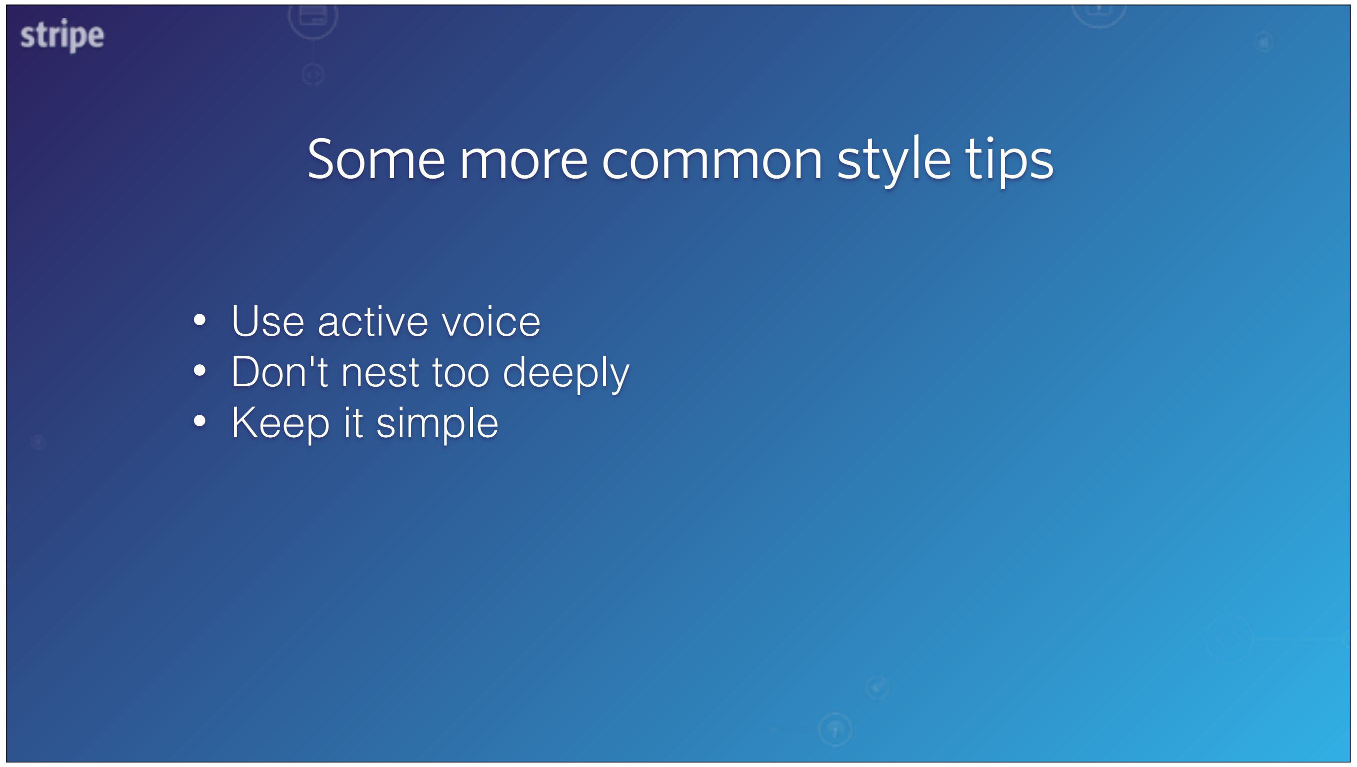
Most important:
Find a voice & stick to it

There are many, many legitimate tones in documentation - how much humor do you use? Do you use an Oxford comma?

Sometimes, that tone will already be selected for you, for example if you join a company like Stripe, where we have a docs style guide.

But some times, you can pick your style yourself. Write down your choices! Make sure your documentation sticks to these choices!

One of my favorite anti-examples is the localization in Fallout 3: German formal and informal second-person pronouns in the same dialog, breaking immersion and causing frustration. They should have stuck to a guide!



stripe

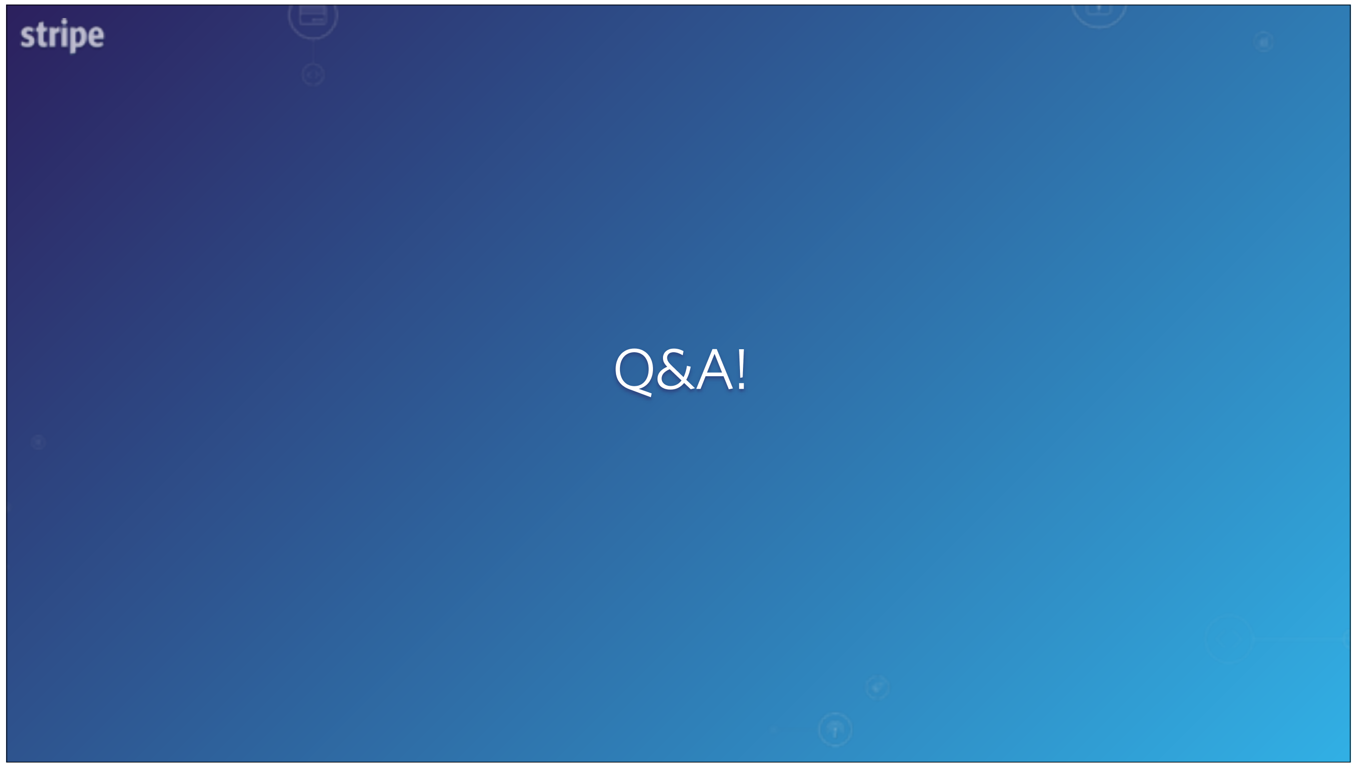
Some more common style tips

- Use active voice
- Don't nest too deeply
- Keep it simple

Pretty much every technical writing guide will tell you the same things that Strunk&White's Elements of Style tells you: Avoid the passive voice, keep your sentences short and to the point.

And of course, don't nest your sentences too deeply either - hi German native speakers; this is one for us.

Whatever you will work on will probably have an established style guide. If you are interested to see the variations out there, I have some links to existing style guides in the one-pager handout.


stripe

Q&A!

I have some questions for you that didn't fit anywhere else:

What do you find frustrating about documentation out there, and how do you wish it was better?

What's the hardest part about writing technical docs for you?

We're also ready to take your questions! Fire away!

stripe

Thanks!

Talk to us:
krithix@stripe.com
asf@stripe.com

Download these slides:
https://github.com/antifuchs/technical-writing-workshop