Subscription APIs
for recurring revenue

Subscription APIs
for recurring revenue

“For the record, I would pay $2 a month for Tweetie.”

@rands

 Oct 11, 2009

Riverfold Software

- Tweet Marker — optional subscription
- Watermark — tweet archive and iPhone app
- Searchpath — web site search

In-app purchase

- How to use the Store Kit framework.
- Tips for getting your subscription approved by Apple.

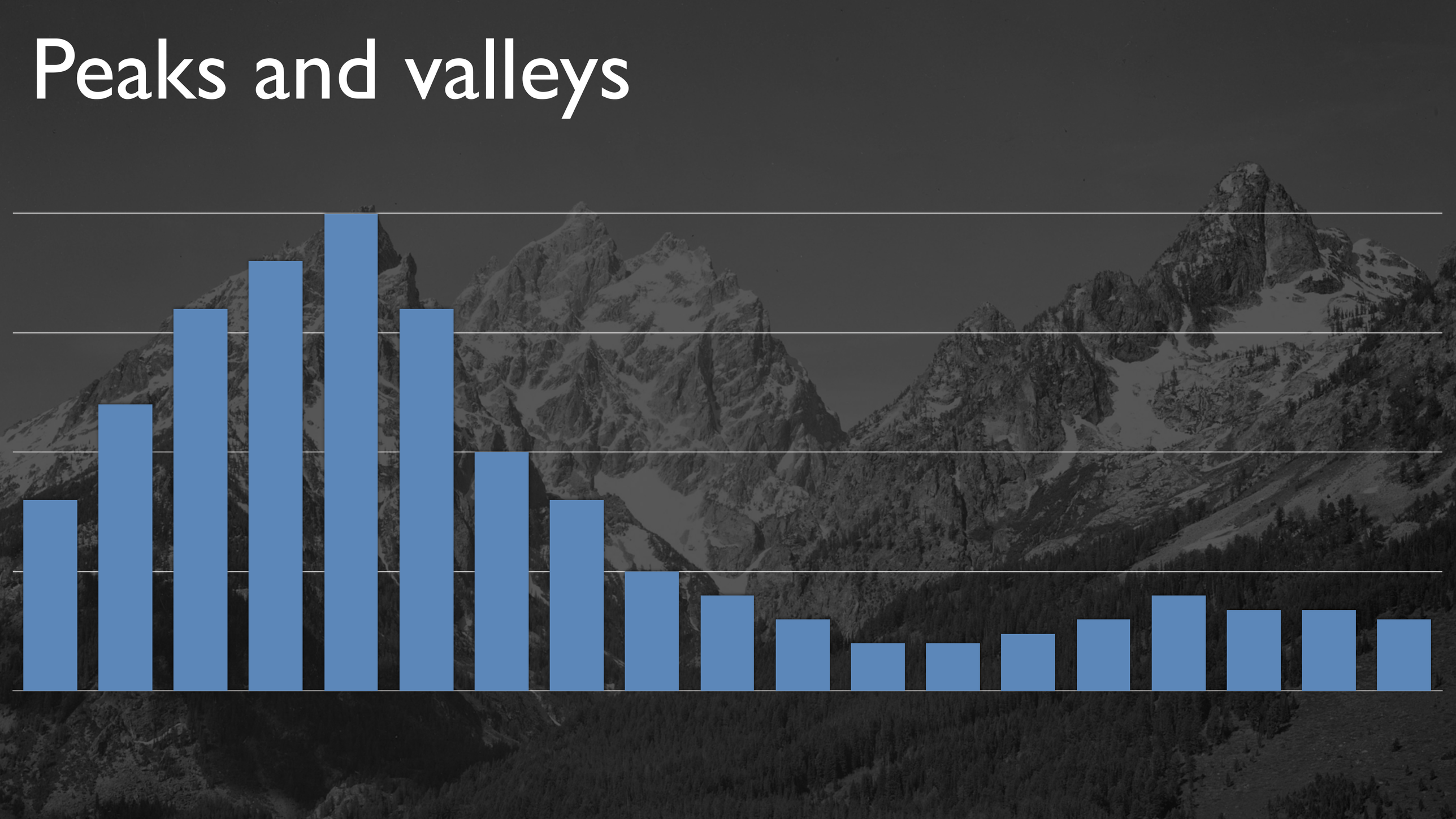
Stripe

- How to implement Stripe in a Mac app.
- JavaScript / Ruby for the web parts.

Why subscriptions?

Peaks and valleys

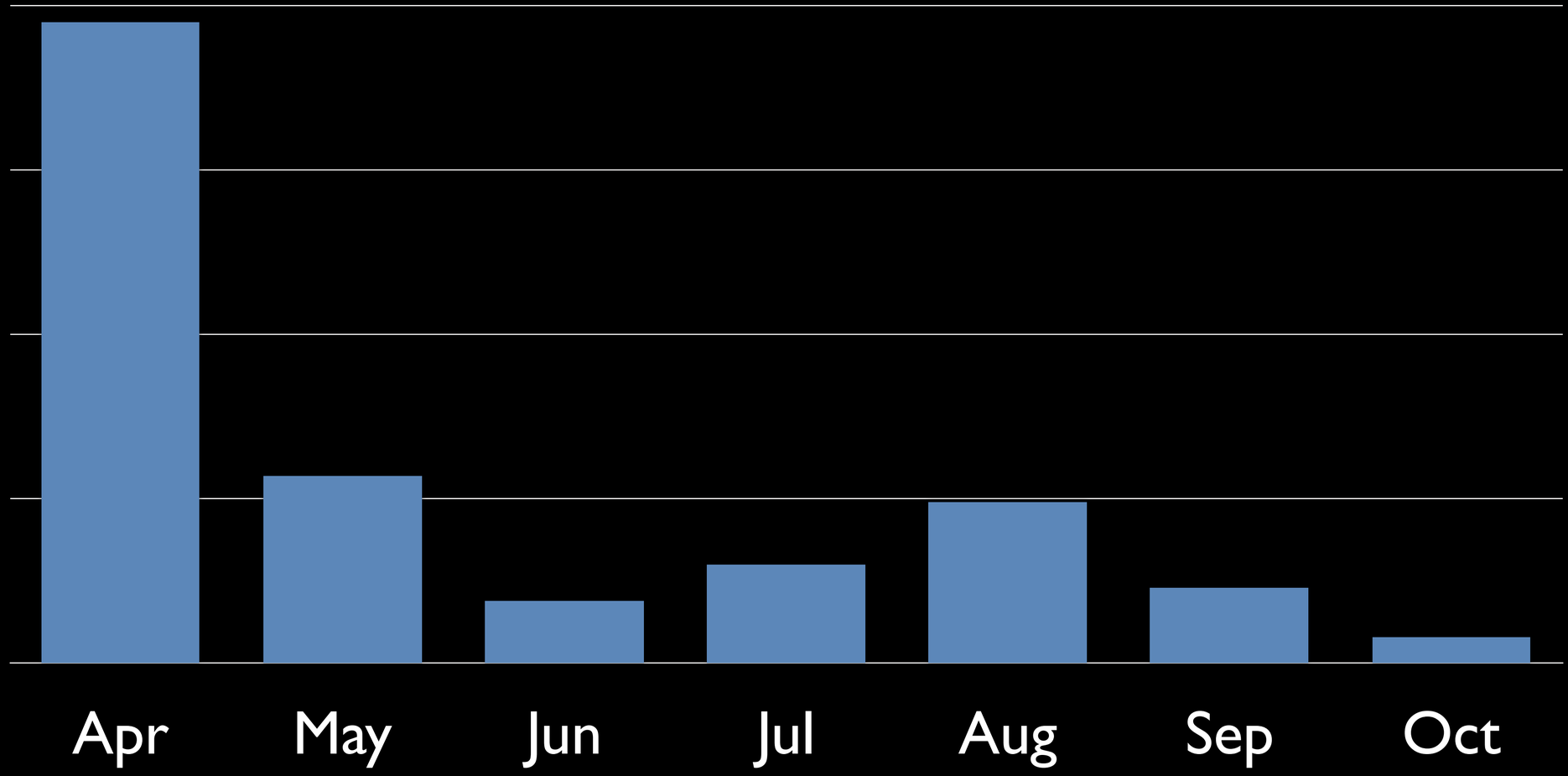
Peaks and valleys



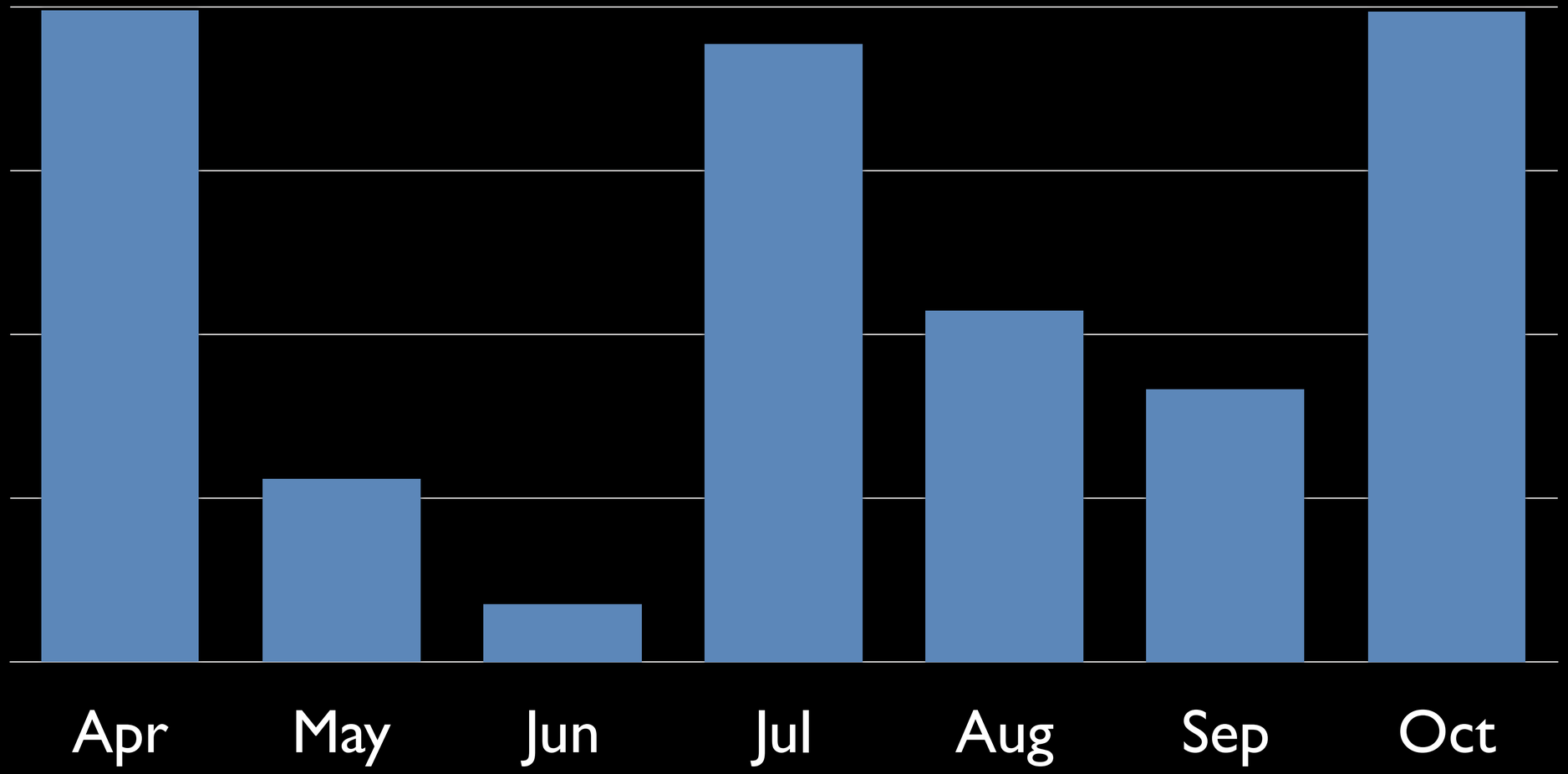
“Those spikes are where we build up our war chest to survive the lean months of working on a new product and watching the long tail of the old product dwindle.”

Wil Shipley

Sign-ups

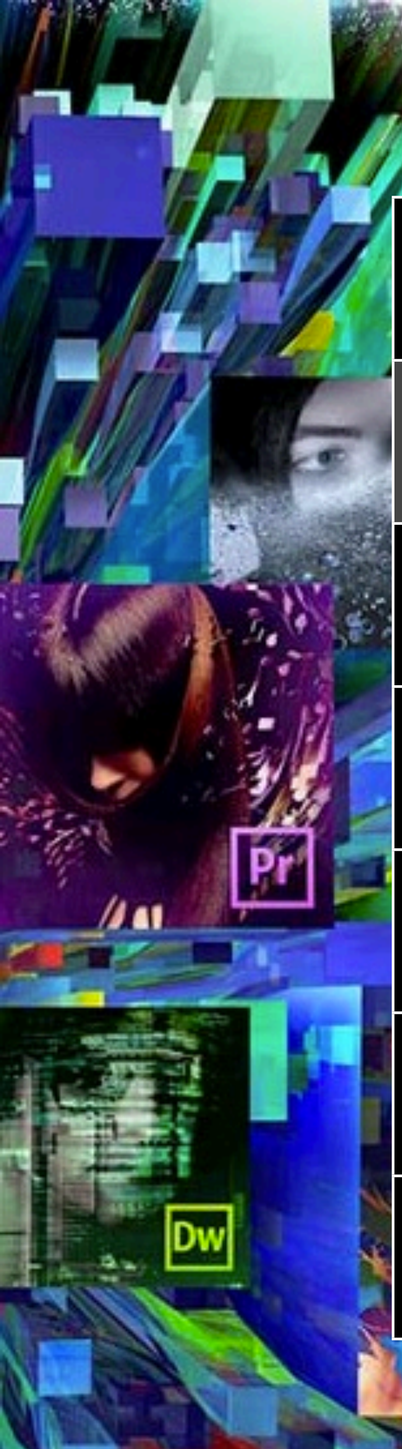


Revenue



Examples


Sg
Ae
Pl
Pr
Au
Dw
Ps
Adobe

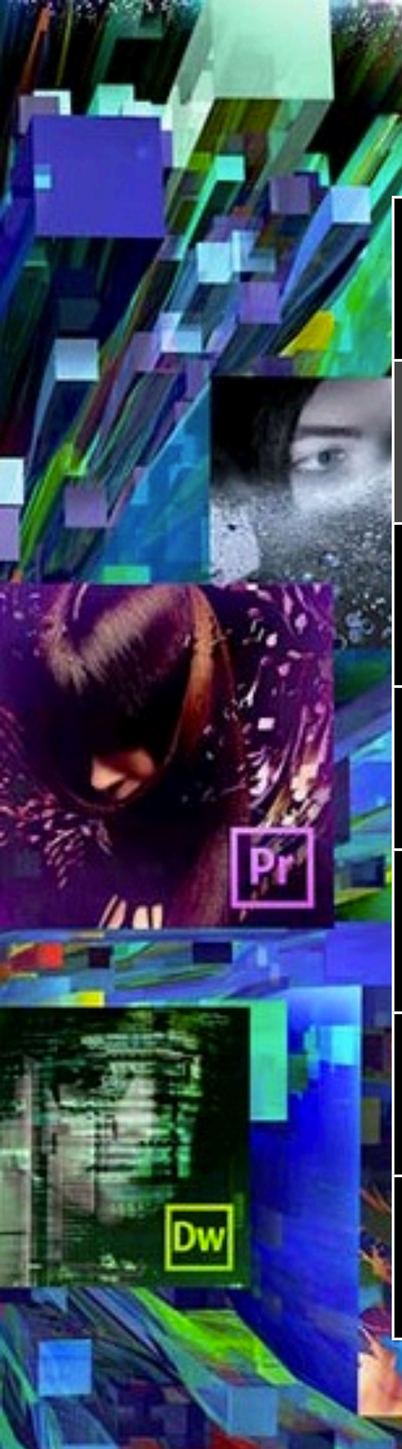


		Design Standard	Production Premium	Master Collection
Products	Price	$1299	$1899	$2599
Photoshop	$699	●	●	●
Lightroom	$149			
Illustrator	$599	●	●	●
InDesign	$699	●		●
Flash	$699		●	●

So what do we do?

So what do we do?

Smaller amounts, more often.



		Design Standard	Production Premium	Master Collection	Creative Cloud
Products	Price	$1299	$1899	$2599	$50/month
Photoshop	$699	●	●	●	●
Lightroom	$149				●
Illustrator	$599	●	●	●	●
InDesign	$699	●		●	●
Flash	$699		●	●	●

Office 365 Home Premium

Get the latest Office applications and more, all in one convenient subscription.

$99.99 per year Buy now

$9.99 per month Buy now

For 5 PCs or Macs[4]

plus select devices[1]

SkyDrive skype

+20GB SkyDrive storage

60 minutes of Skype calls per month[3]

Always up-to-date

Learn More

Office for a single PC

Office Home & Student 2013

1 install includes:

Office Home & Business 2013

1 install includes:

Office Professional 2013

1 install includes:

Buy Office for Business

Compare Office suites

Common questions

Free Office trial

System requirements

Marvel

Comics Unlimited

iPad app or web

$10/month

Billings Pro

with Marketcircle Cloud

1 invoice: free

5 invoices: $10/month

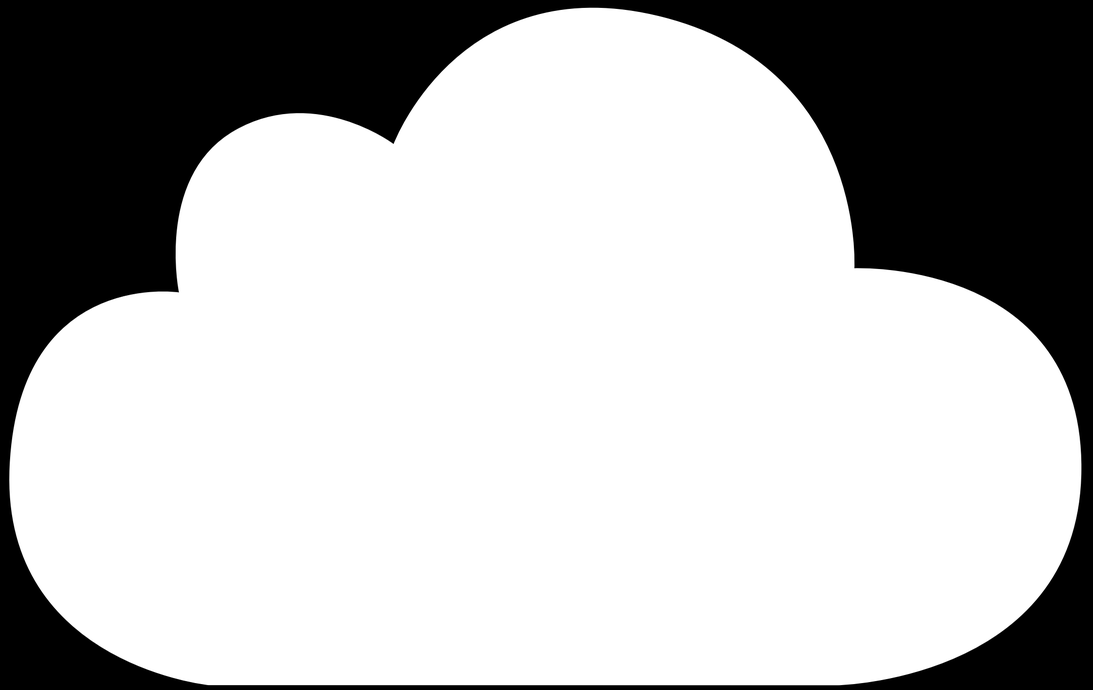

I'LL SPARE YOU
THE NUTS & BOLTS
of the numerous and surprising
ways that the room in which
wealthy persons remove their shoes
might contribute to causing
the carefully-coiffed, 60-year-old
woman to come unglued over
"priorities." Suffice to say, fixing
this problem was a "high priority"
for her. So, she said, repeatedly, as
shifted my wallet, her ray, coffee go
cold, and balled the little blue pillow
under my neck. Except. The lady in
the seat directly behind me was
having grave problems with her
"mud room." Big mud room prob-
Because, depending on who you
talk to, most people say "prioritiz-
ing" is either a giant problem, an
underused skill, or a "Get out of
jail free" card. Me? I think priori-
ties are simple to understand
precisely because their influence is
so staggeringly clear and unavoid-
able to behold, then act upon.
Ready for this one? A priority is
observed, not manufactured or
assigned. Otherwise, it's necessar-
ily not a priority. Got that? You
can't "prioritize" a list of 20 tasks.
MANUFACTURED
OR ASSIGNED

"What I saw was a fluke in App Store pricing."

Marco Arment

npr Feb 21, 2013

"To make something special, you just have to believe it's special."



Adobe	100 GB cloud storage
Microsoft	7 GB cloud storage
Marvel	all back issues downloadable
Billings Pro	cloud sync
Instapaper	full-text search via cloud

Users pirate apps because they feel the downloaded bits are worthless.

Cloud services have
visible cost and value.

In-app purchase

- Non-renewing subscriptions
- Auto-renewable subscriptions

Rejected

“Your In-App Purchase is currently an Auto-Renewable Subscription. However, it would be more appropriate to use the Non-Renewing Subscription In-App Purchase type. Auto-Renewable Subscriptions are intended for periodical apps, such as magazines and newspapers.”

 Apple — Resolution Center

“Apps may only use auto renewing subscriptions for periodicals (newspapers, magazines), business Apps (enterprise, productivity, professional creative, cloud storage) and media Apps (video, audio, voice), or the App will be rejected.”

App Store Review Guidelines

section 11.15

Rejected again!

“Specifically, we found that your Application Description did not include:

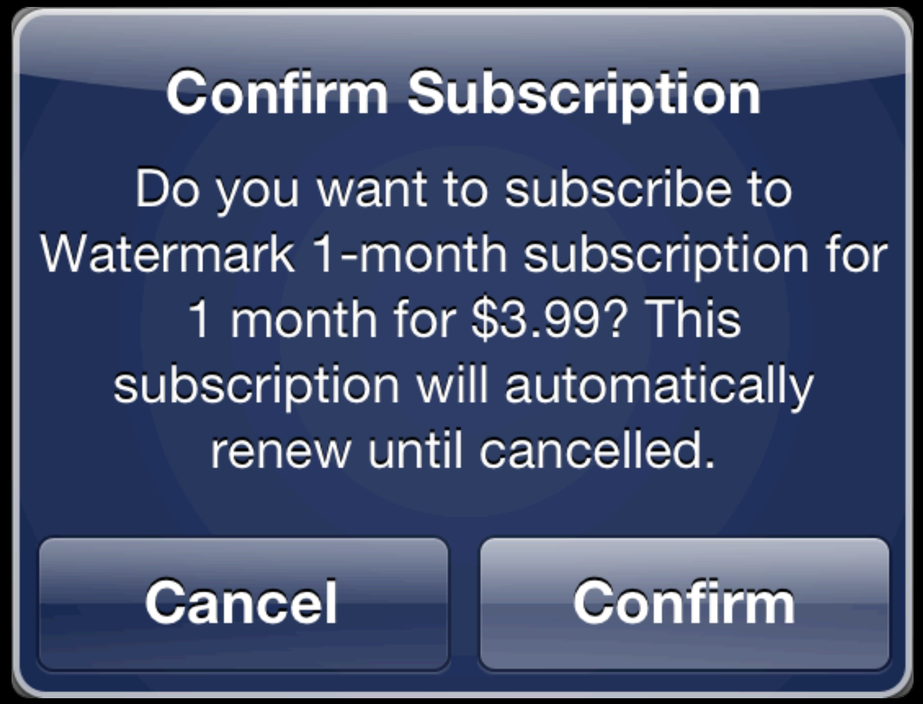
1) The length of the subscription
2) The price of the subscription
3) Information about the auto-renewable nature of the subscription
4) Links to the privacy policy and terms of use”

 Apple — Resolution Center

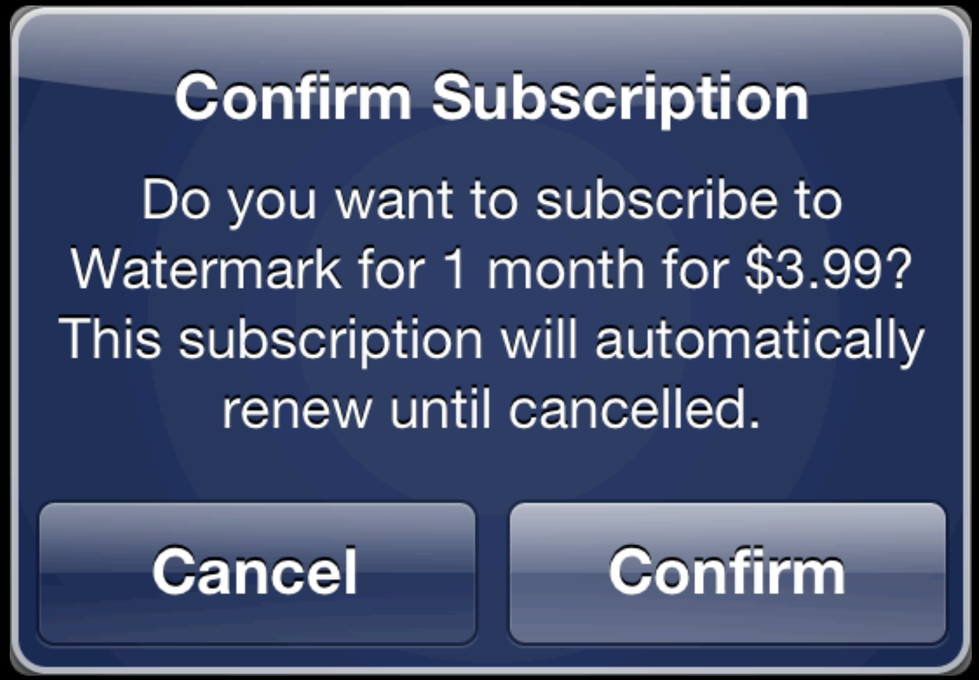
iTunes Connect

- Duration
- Price
- Metadata
- Screenshot
- Language

Display name:
“Watermark 1-month subscription”



Display name:
“Watermark”



StoreKit.framework

- Ask for product info on startup with SKProductsRequest:
    - receive product info
    - SKProductsRequestDelegate
- Initiate a payment with SKPaymentQueue and SKPayment:
    - receive notice of the transaction
    - SKPaymentTransactionObserver

```
SKProductsRequest* request = [[SKProductsRequest alloc]
initWithProductIdentifiers:
    [NSSet setWithObjects:
        @"com.myapp.subscription.1month",
        @"com.myapp.subscription.6months",
        @"com.myapp.subscription.1year",
        nil
    ]];
request.delegate = self;
[request start];
```

```
NSURL* url = [NSURL URLWithString:
                @"https://secure.myapp.com/products/identifiers"];
NSURLRequest* request = [NSURLRequest requestWithURL:url];
```

```
GET /products/identifiers

[
   "com.myapp.subscription.1month",
   "com.myapp.subscription.6months",
   "com.myapp.subscription.1year"
]
```

```
AFJSONRequestOperation* operation = [AFJSONRequestOperation
    JSONRequestOperationWithRequest:request
    success:^(NSURLRequest* request, NSHTTPURLResponse* response, id json)
{
        SKProductsRequest* store_request = [[SKProductsRequest alloc]
initWithProductIdentifiers:[NSSet setWithArray:json]];
        store_request.delegate = self;
        [store_request start];
}
failure:nil];

[operation start];
```

```
AFJSONRequestOperation* operation = [AFJSONRequestOperation
    JSONRequestOperationWithRequest:request
    success:^(NSURLRequest* request, NSHTTPURLResponse* response, id json)
{
        SKProductsRequest* store_request = [[SKProductsRequest alloc]
initWithProductIdentifiers:[NSSet setWithArray:json]];
        store_request.delegate = self;
        [store_request start];
}
failure:nil];

[operation start];
```

```
- (void) productsRequest:(SKProductsRequest *)request
didReceiveResponse:(SKProductsResponse *)response
{
    for (SKProduct* product in response.products) {
        NSLog (@"product ID: %@", product.productIdentifier);
        NSLog (@"price: %@",
            [NSString stringWithFormat:@"%@%@",
                [product.priceLocale objectForKey:NSLocaleCurrencySymbol],
                product.price
            ]
        );
    }
}
```

```
- (void) productsRequest:(SKProductsRequest *)request
didReceiveResponse:(SKProductsResponse *)response
{
    for (SKProduct* product in response.products) {
        NSLog (@"product ID: %@", product.productIdentifier);
        NSLog (@"price: %@",
            [NSString stringWithFormat:@"%@%@",
                [product.priceLocale objectForKey:NSLocaleCurrencySymbol],
                product.price
            ]
        );
    }
}
```

```
if (![SKPaymentQueue canMakePayments]) {
    // ...
}

[[SKPaymentQueue defaultQueue] addTransactionObserver:self];
```

```
SKProduct* selected_product = ...;
SKPayment* payment = [SKPayment
    paymentWithProduct:selected_product];
[[SKPaymentQueue defaultQueue] addPayment:payment];
```

```
- (void) paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions
{
    for (SKPaymentTransaction* transaction in transactions) {
        switch (transaction.transactionState) {
            case SKPaymentTransactionStatePurchased: {
                [self sendReceipt:transaction];
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
            } break;
            case SKPaymentTransactionStateFailed: {
                [self showError:transaction.error];
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
            } break;
        }
    }
}
```

```
- (void) paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions
{
    for (SKPaymentTransaction* transaction in transactions) {
        switch (transaction.transactionState) {
            case SKPaymentTransactionStatePurchased: {
                [self sendReceipt:transaction];
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
            } break;
            case SKPaymentTransactionStateFailed: {
                [self showError:transaction.error];
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
            } break;
        }
    }
}
```

```
- (void) paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions
{
    for (SKPaymentTransaction* transaction in transactions) {
        switch (transaction.transactionState) {
            case SKPaymentTransactionStatePurchased: {
                [self sendReceipt:transaction];
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
            } break;
            case SKPaymentTransactionStateFailed: {
                [self showError:transaction.error];
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
            } break;
        }
    }
}
```

```
NSMutableDictionary* params = [[NSMutableDictionary alloc] init];

[params setObject:_username forKey:@"username"];
[params setObject:transaction.payment.productIdentifier
   forKey:@"product_id"];
[params setObject:
   [NSString stringBase64WithData:transaction.transactionReceipt]
   forKey:@"receipt"
];
```

```
NSURL* url = [NSURL URLWithString:@”https://secure.myserver.com/”];
AFHTTPClient* client = [[AFHTTPClient alloc] initWithBaseURL:url];
client.parameterEncoding = AFJSONParameterEncoding;
NSMutableURLRequest* request = [client requestWithMethod:@"POST"
                                                    path:@"/receipts/verify"
                                                    parameters:params];

AFJSONRequestOperation* operation = [AFJSONRequestOperation
JSONRequestOperationWithRequest:request success:^(...) {
    // unlock content, show full UI, etc.
    // ...
} failure:^(...) {
}];

[operation start];
```

The server

“Nearly everything I want to do requires a web backend. I need to make more friends with web devs.”

@justin

 Jan 23, 2013

Custom cloud support is a competitive advantage.

Ruby Sinatra

```ruby
get “/some/path” do
    lookup_something
    erb :my_template
end

post “/another/path” do
    save_something
    return “OK”
end
```

```
post “/receipts/verify” do
    content_type :json
    return “{}”
end
```

```
post “/receipts/verify” do
    check_receipt
    update_user
    return “{}”
end
```

```ruby
http = Net::HTTP.new("buy.itunes.apple.com", 443)
http.use_ssl = true
http.start do |http|
    req = Net::HTTP::Post.new("/verifyReceipt")
    fields = {
         "receipt-data" => params[:receipt],
         "password" => "abcdefghi"
    }
    req.body = fields.to_json
    response = http.request(req)
    info = JSON.parse(response.body)
end
```

```
{
  “status” : 21006,
  “receipt” : {
    “product_id” : “com.myapp.subscription.1month”,
    “expires_date” : 1365198070234,
    “transaction_id” : 1000000070220898,
      ...
  }
  “latest_expired_receipt_info” : {
      ...
  }
}
```

```
user = User.find(:username => params[:username])

user.paid = true
user.subscribed_product = info["receipt"]["product_id"]
user.expires_at =
          Time.at(info["receipt"]["expires_date"] / 1000)

user.save
```

[[SKPaymentQueue defaultQueue] restoreCompletedTransactions];

```
- (void) paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions
{
    for (SKPaymentTransaction* transaction in transactions) {
        switch (transaction.transactionState) {
            case SKPaymentTransactionStateRestored: {
                // ...
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
            } break;
        }
    }
}
```

restoreCompletedTransactions

vs.

your own server accounts

Stripe

Reasons to consider Stripe:

- You can’t get auto-renewing IAP approved.
- You can't sell in the iOS App Store — Mac or web.
- You want more control over the process and fees.

2.9% + 30¢ vs. 30%

2.9% + 30¢ vs. 30%

500 subscribers, $1.99/month, revenue after 1 year

2.9% + 30¢ vs. 30%

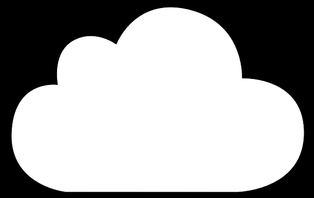
500 subscribers, $1.99/month, revenue after 1 year

2.9% + 30¢ vs. 30%

500 subscribers, $1.99/month, revenue after 1 year

Mac app

Stripe

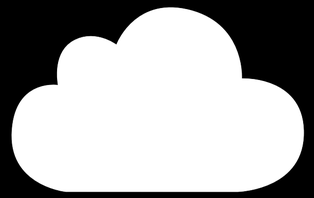


your server

Mac app

1234–1234–1234–1234
expires: 12-2014

Stripe



your server

Mac app

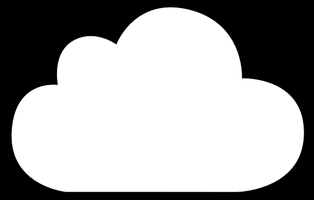
token

Stripe

your server

Mac app

token



your server

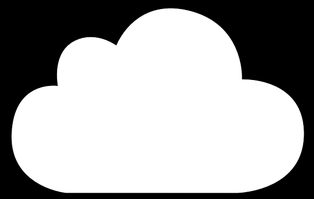
Stripe

Mac app

Stripe

create a customer

token + plan



your server

Stripe SDK

`github.com/stripe/stripe-ios`

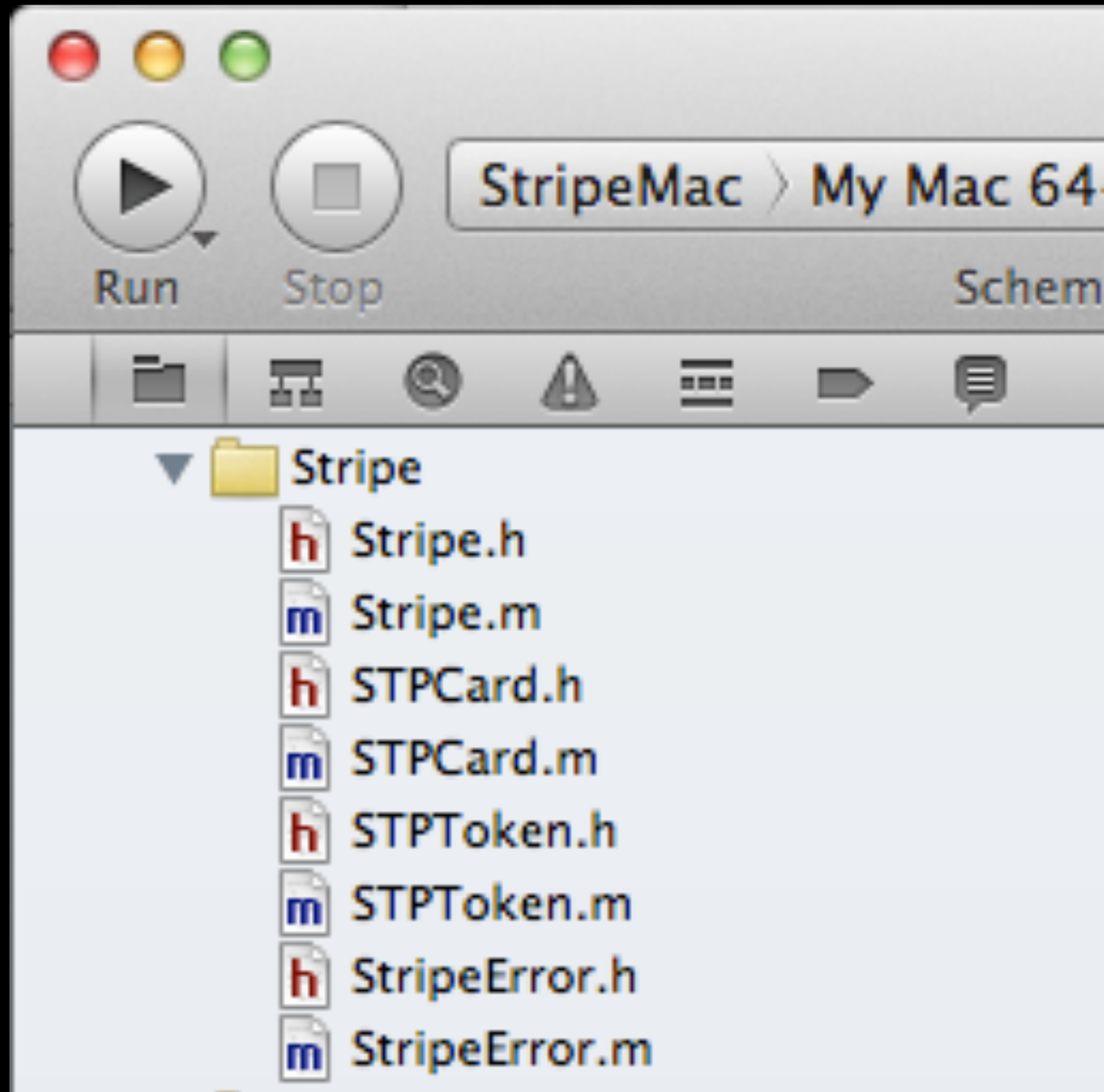
Mac or iOS classes:

- Stripe
- STPCard
- STPToken
- StripeError

iOS-only:

- STPView

CocoaPods?


Run
Stop
StripeMac
My Mac 64
Schem
Stripe
Stripe.h
Stripe.m
STPCard.h
STPCard.m
STPToken.h
STPToken.m
StripeError.h
StripeError.m

```
[Stripe setDefaultPublishableKey:@"pk_abcdef123456"];
```

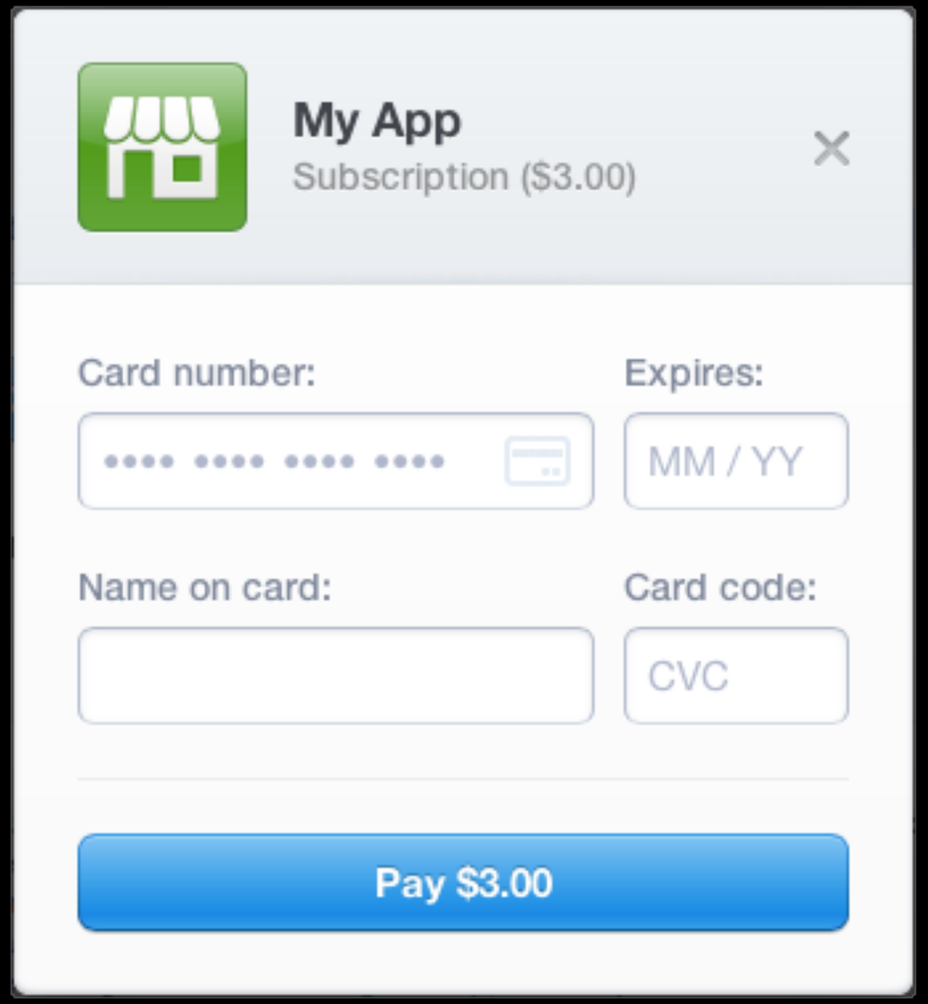
```objc
STPCard* card = [[STPCard alloc] init];

card.number = @"4242424242424242";
card.expMonth = 1;
card.expYear = 2014;
card.cvc = @"123";
```

```
[Stripe createTokenWithCard:card
        completion:^(STPToken* token, NSError* error) {

    NSLog (@"Received token: %@", token.tokenId);

    // submit to server here
    // ...

}];
```

```
NSMutableDictionary* params = [[NSMutableDictionary alloc] init];

[params setObject:token.tokenId forKey:@"stripe_token"];
[params setObject:_email forKey:@"email"];
[params setObject:transaction.payment.productIdentifier
   forKey:@"plan"];

NSURL* url = [NSURL URLWithString:@”https://secure.myserver.com/”];
AFHTTPClient* client = [[AFHTTPClient alloc] initWithBaseURL:url];
client.parameterEncoding = AFJSONParameterEncoding;
NSMutableURLRequest* request = [client requestWithMethod:@"POST"
                                                  path:@"/stripe"
                                                  parameters:params];
// ...
```

```
post “/stripe” do
  stripe_token = params[:stripe_token]
  email = params[:email]
  plan = params[:plan]

  stripe_customer = Stripe::Customer.create(
    :card => stripe_token,
    :plan => plan,
    :email => email,
    :description => site
  )

  update_user
end
```

Stripe from a web app




```
<script
    src="https://checkout.stripe.com/v2/checkout.js"
    class="stripe-button"
    data-key="pk_abcdef123456"
    data-amount="300"
    data-name="My App"
    data-description="Subscription ($3.00)"
    data-image="/128x128.png">
</script>
```

You will be charged $8 each month and can cancel any time. If you have any questions, please email support@riverfold.com.

Not ready to subscribe? Skip to the Searchpath dashboard.

Cards are processed securely by Stripe.

Name on card:

Card number: •••• •••• •••• ••••

Expires: MM YY

Card code: CVC

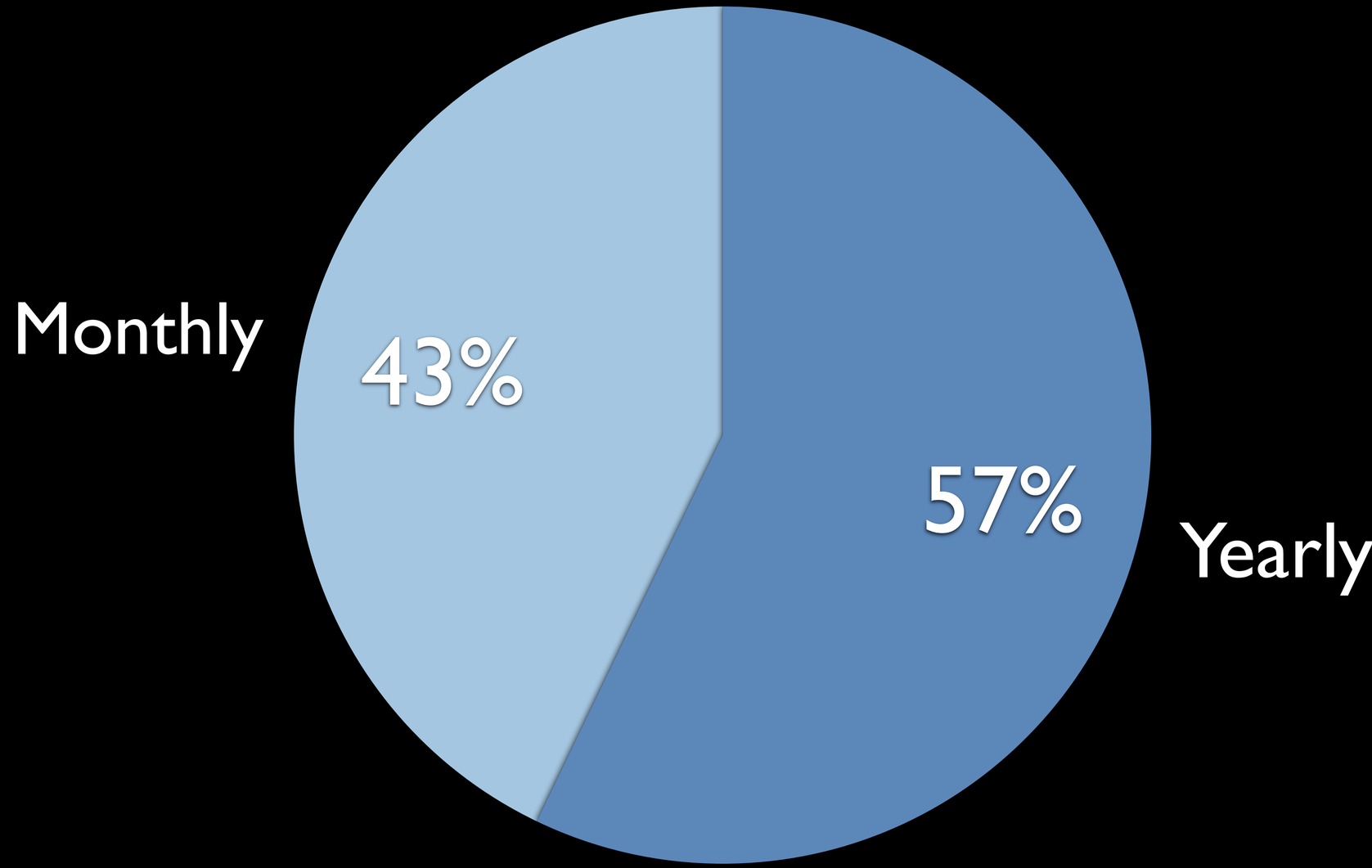
Billing ZIP code: 8.25% sales tax will be added for Texas addresses.

- (•) $8/month
- ( ) $75/year (save 20%)

Subscribe

“What if I give you 80 bucks a year rather than 8 a month? I'd love to.”

customer email


Monthly
43%
57%
Yearly

You will be charged $8 each month and can cancel any time. If you have any questions, please email support@riverfold.com.

Not ready to subscribe? Skip to the Searchpath dashboard.

Cards are processed securely by Stripe.

Name on card:

Card number: •••• •••• •••• ••••

Expires: MM YY

Card code: CVC

Billing ZIP code: 8.25% sales tax will be added for Texas addresses.

- (•) $8/month
- ( ) $75/year (save 20%)

Subscribe

```
<form method="POST" action="/stripe">

Name: <input name="full_name" id="subscribe_name" /><br />
Credit card: <input id="subscribe_cardnumber" /><br />
Expires: <input id="subscribe_expiresmonth" /><br />
ZIP: <input name="zipcode" id="subscribe_zipcode" /><br />

...

</form>
```

```
$("#payment_form").submit(function(event) {

    Stripe.createToken({
        number: $('#subscribe_cardnumber').val(),
        cvc: $('#subscribe_cvc').val(),
        name: $('#subscribe_name').val(),
        exp_month: $('#subscribe_expiresmonth').val(),
        exp_year: $('#subscribe_expiresyear').val(),
        address_zip: $('#subscribe_zipcode').val()
      }, stripeResponseHandler);

    return false;
});
```

```
function stripeResponseHandler(status, response) {
    if (response.error) {
        $("#payment_errors").show();
        $("#payment_errors").text(response.error.message);
    }
    else {
        var form$ = $("#payment_form");
        var token = response['id'];
        form$.append("<input type='hidden' name='stripe_token'
value='" + token + "'/>");
        form$.get(0).submit();
    }
}
```

```
post “/stripe” do
  stripe_token = params[:stripe_token]
  email = params[:email]
  plan = params[:plan]

  stripe_customer = Stripe::Customer.create(
    :card => stripe_token,
    :plan => plan,
    :email => email,
    :description => site
  )

  update_user
end
```

Sending invoices and receipts

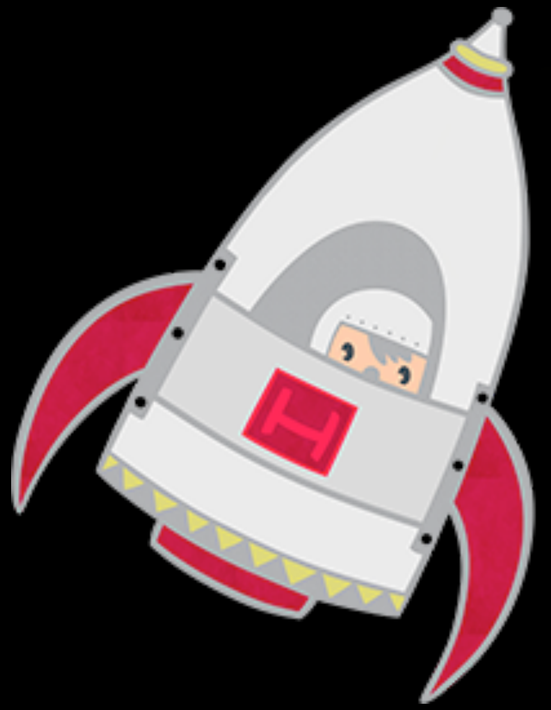
```
INVOICE RECEIPT
---------------

Thanks for subscribing to Searchpath! Your card has been
charged for the period ending 2013-04-14. You can keep this
receipt for your records.

Invoice ID: in_1SYQ5XYCc2jDpJ
Date: 2013-03-14 15:30

CUSTOMER
---------------

Manton Reece
manton@manton.org

SUBSCRIPTION
---------------

Plan: Searchpath
Period: 2013-03-14 to 2013-04-14

Subtotal: $8.00
Sales tax: $0.66

TOTAL: $8.66 (USD)

Charged to card: Visa, **** **** **** 1234

Questions? Please email support@riverfold.com.
```

Shouldn’t most of this be open source?

- It would use Ruby Sinatra. Easy to deploy.
- It would have a database to store products and receipts.
- It would provide the basic server endpoints to call from iOS.

Helios

by Mattt Thompson



helios.io

Summary

- In-app purchase:
  - Auto-renewing subscriptions are great.
  - Appeal a rejection if you think your app fits the guidelines.
- Stripe:
  - Great option for subscriptions outside the App Store.
  - Mac or a web app.

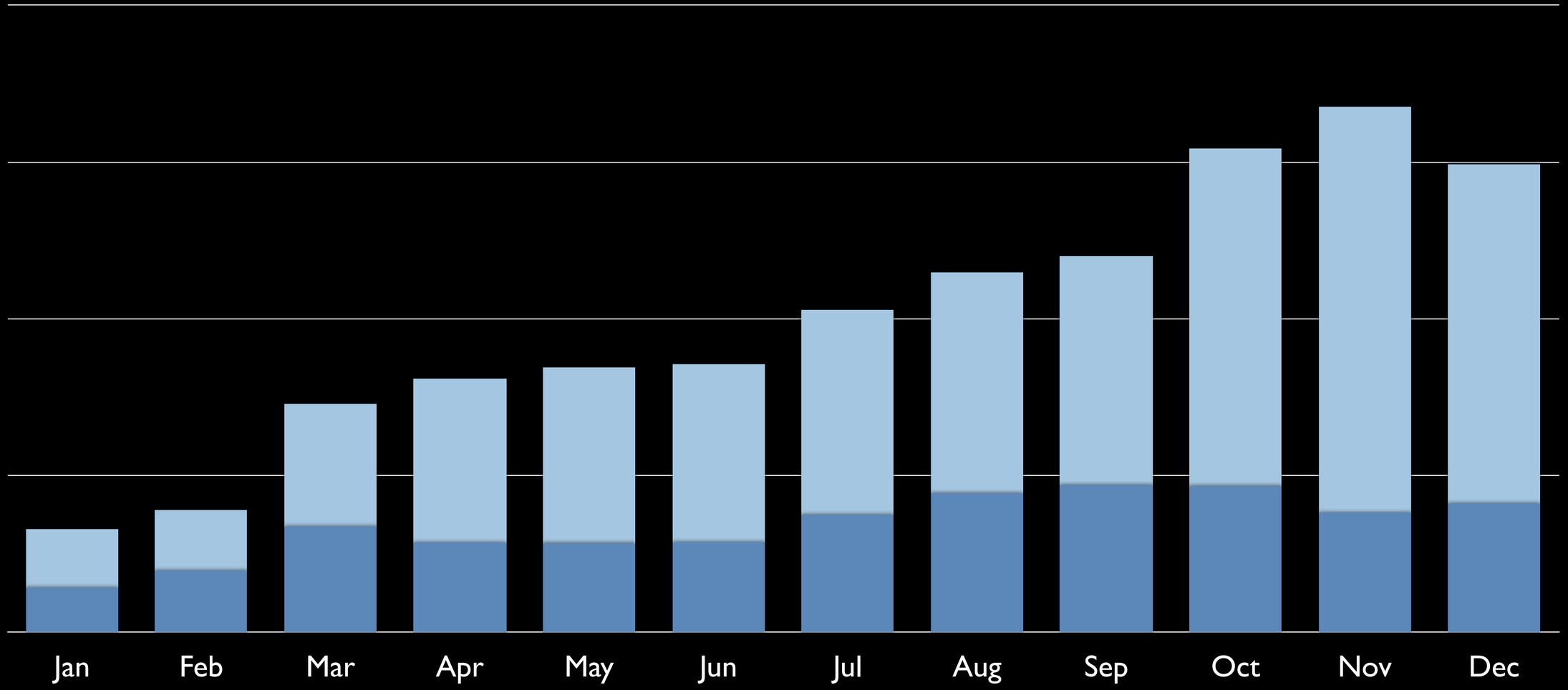
Subscription fatigue

“I really like Tweet Marker, but $1/month seems like way too much for me. If I paid this much for each micro-service like Tweet Marker I would run out of money soon.”

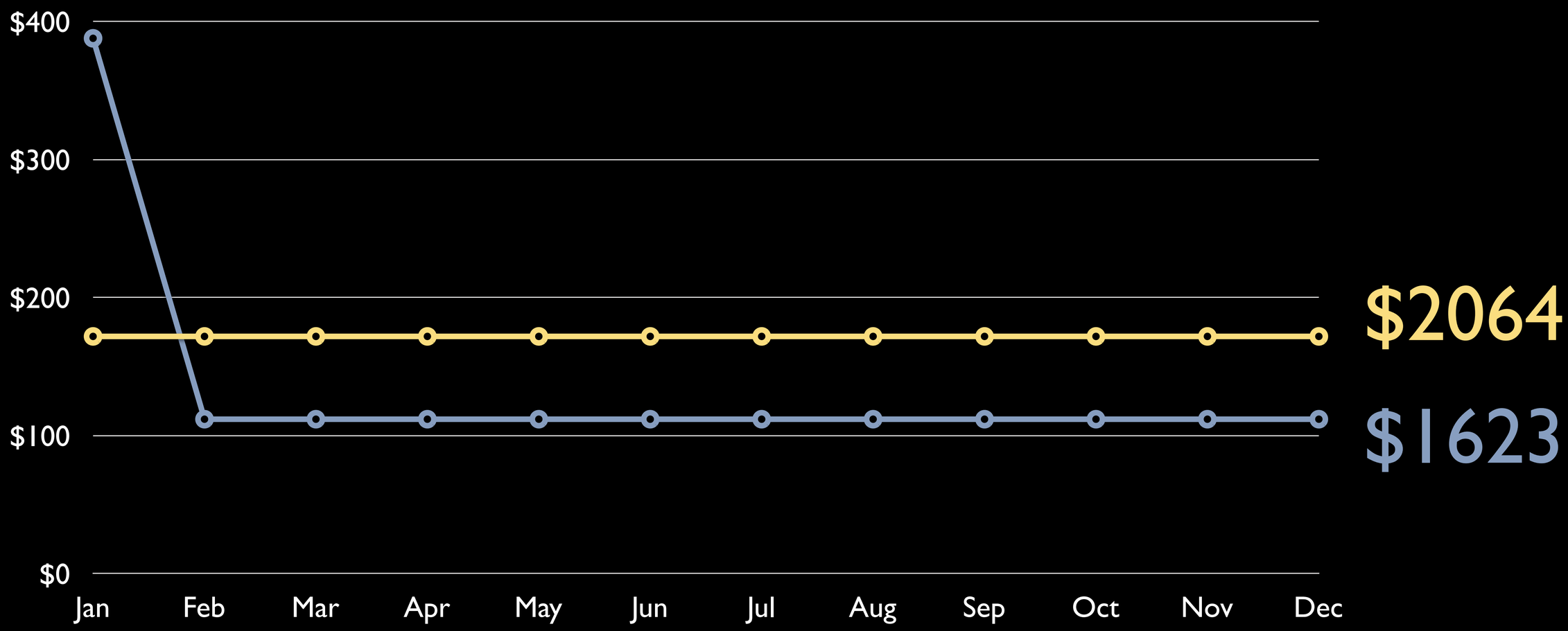
customer email

Mistakes

Hosting costs



Amazon “reserved” instances



“You don’t want to limit the success of your app just because you didn’t want to write your own server.”

Brent Simmons

Thanks!

@manton on App.net

manton**@manton.org**