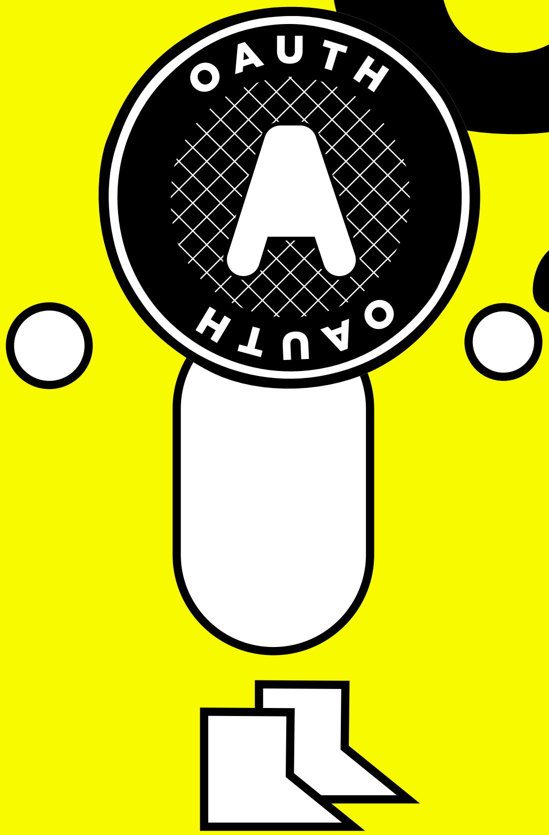
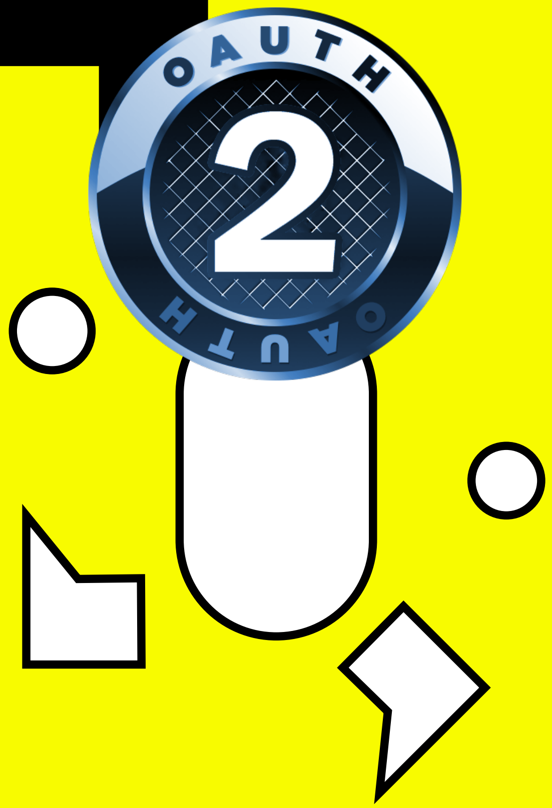
OAUTH

MICHAEL BLEIGH PRESENTS

THE PRESENT FUTURE OF OAUTH



with drawings



PROLOGUE

MY NAME IS

MICHAEL BLEIGH

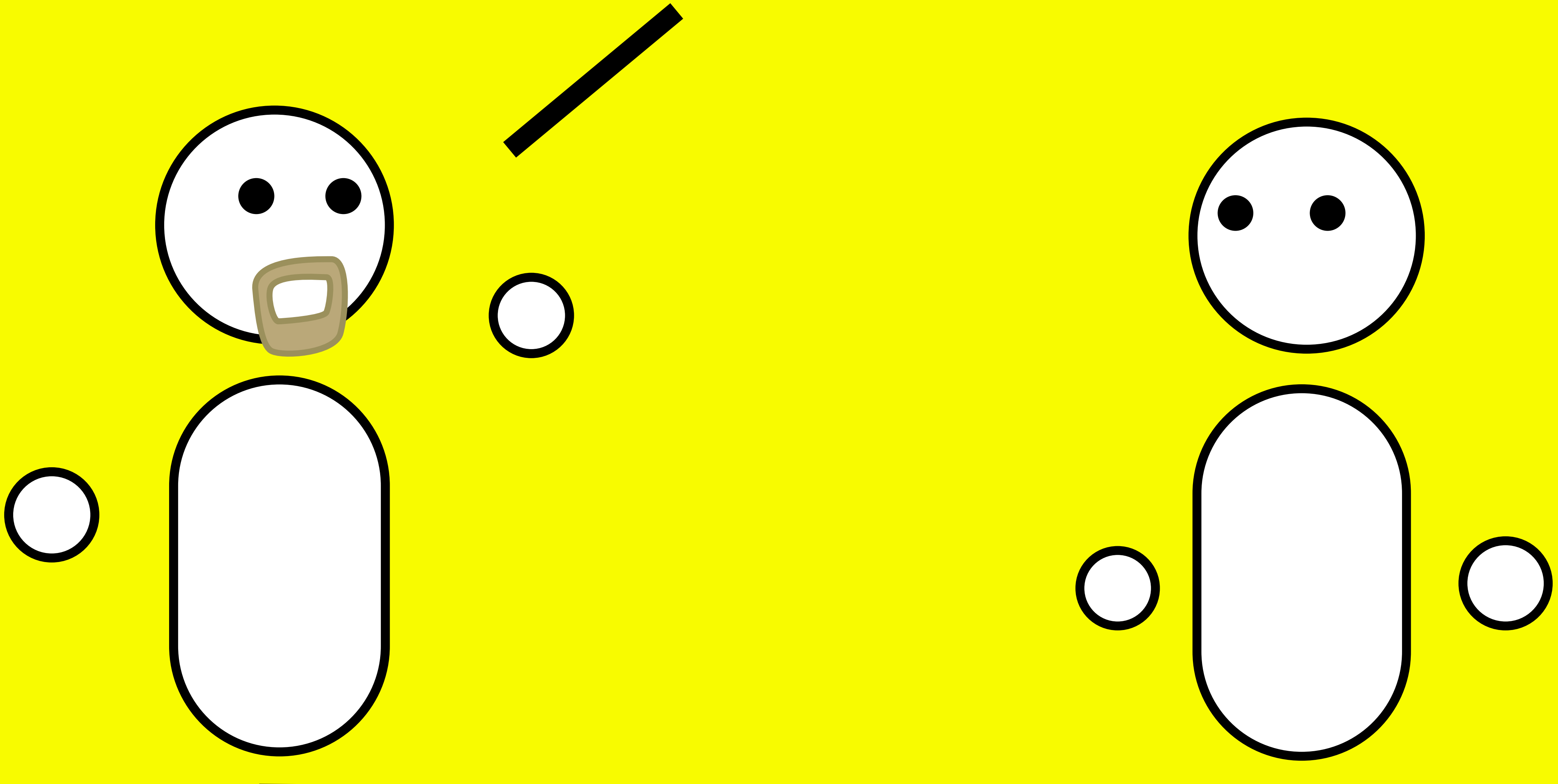
I WORK AT

INTRIDEA

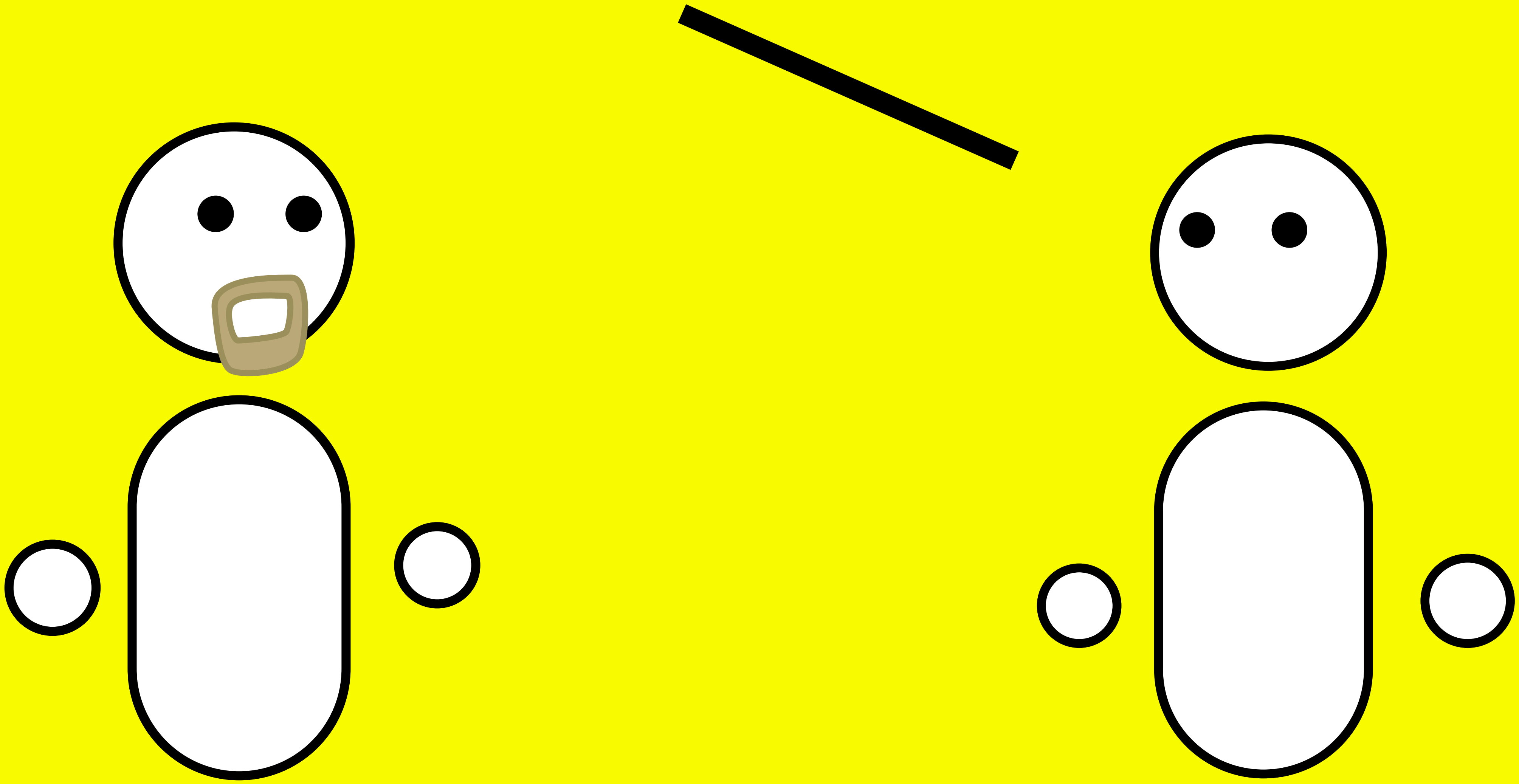
ON TWITTER

@MBLEIGH

“HEY, WOULD ANYONE BE INTERESTED IN GIVING A TALK ABOUT OAUTH AT RAILSCONF?”



“NO WAY, I MIGHT FALL ASLEEP WHILE SPEAKING”



“HMM…I’D BETTER ADD SOME DRAWINGS.”

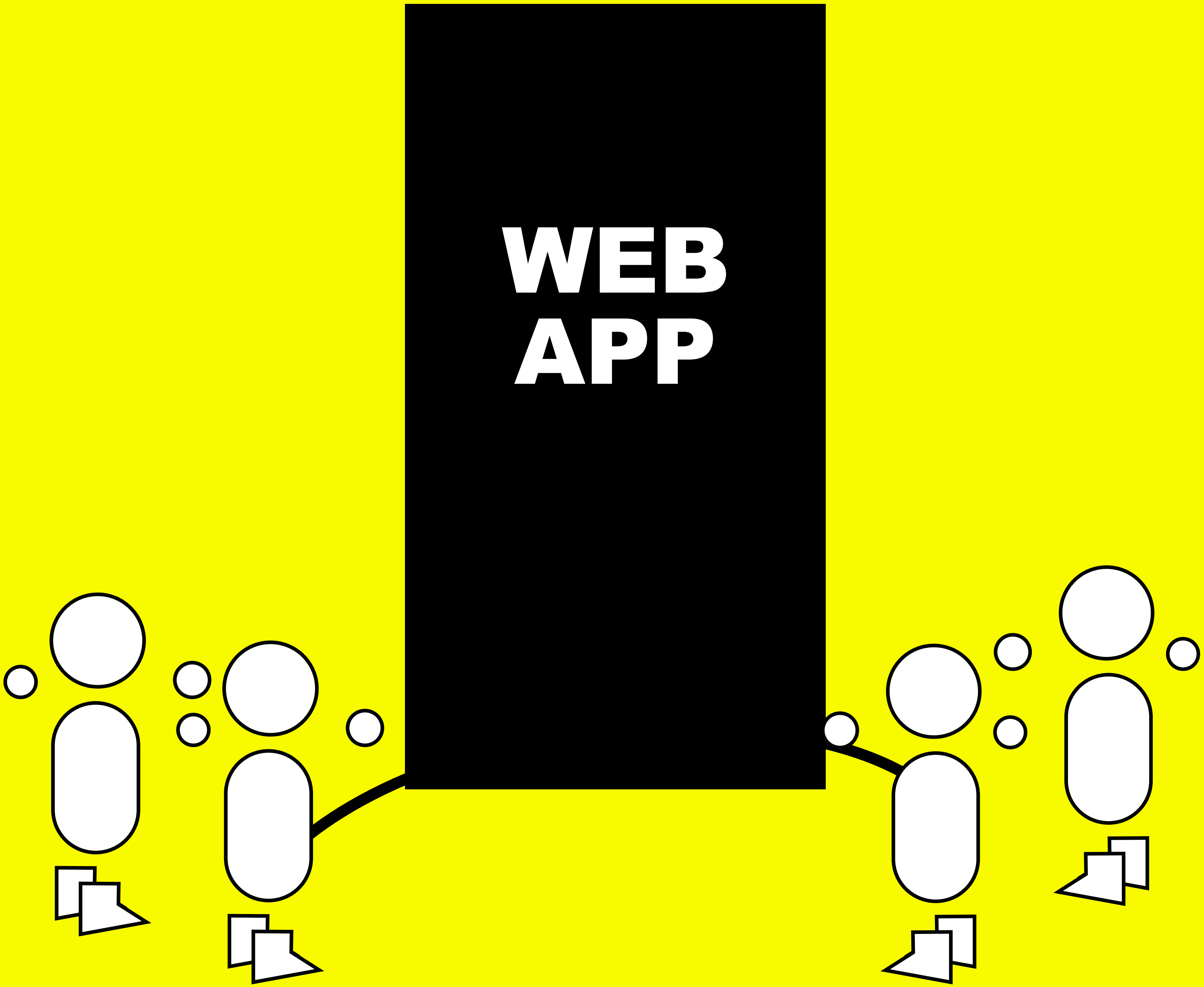
THIS TALK IS ABOUT OPEN WEB STANDARDS

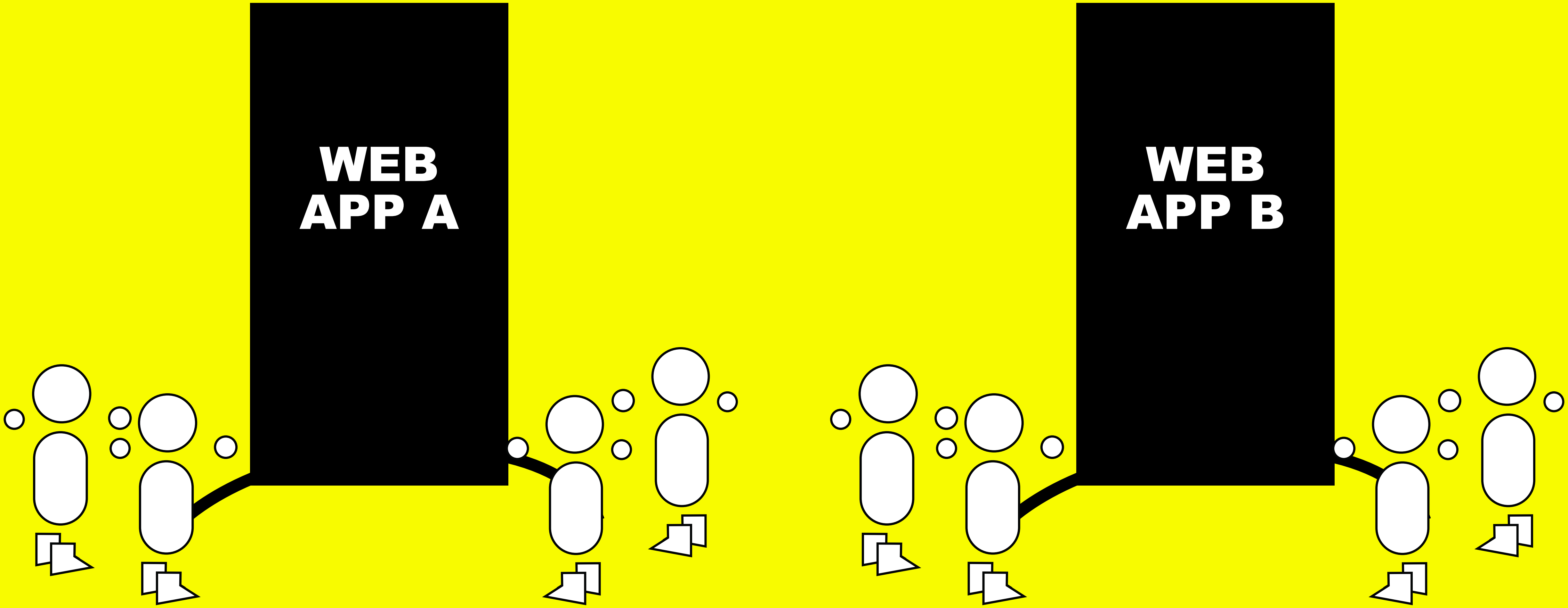
ACT I

IN WHICH THE PROBLEM IS DESCRIBED

IN THE BEGINNING, THERE WERE WEB APPS

WEB
APP

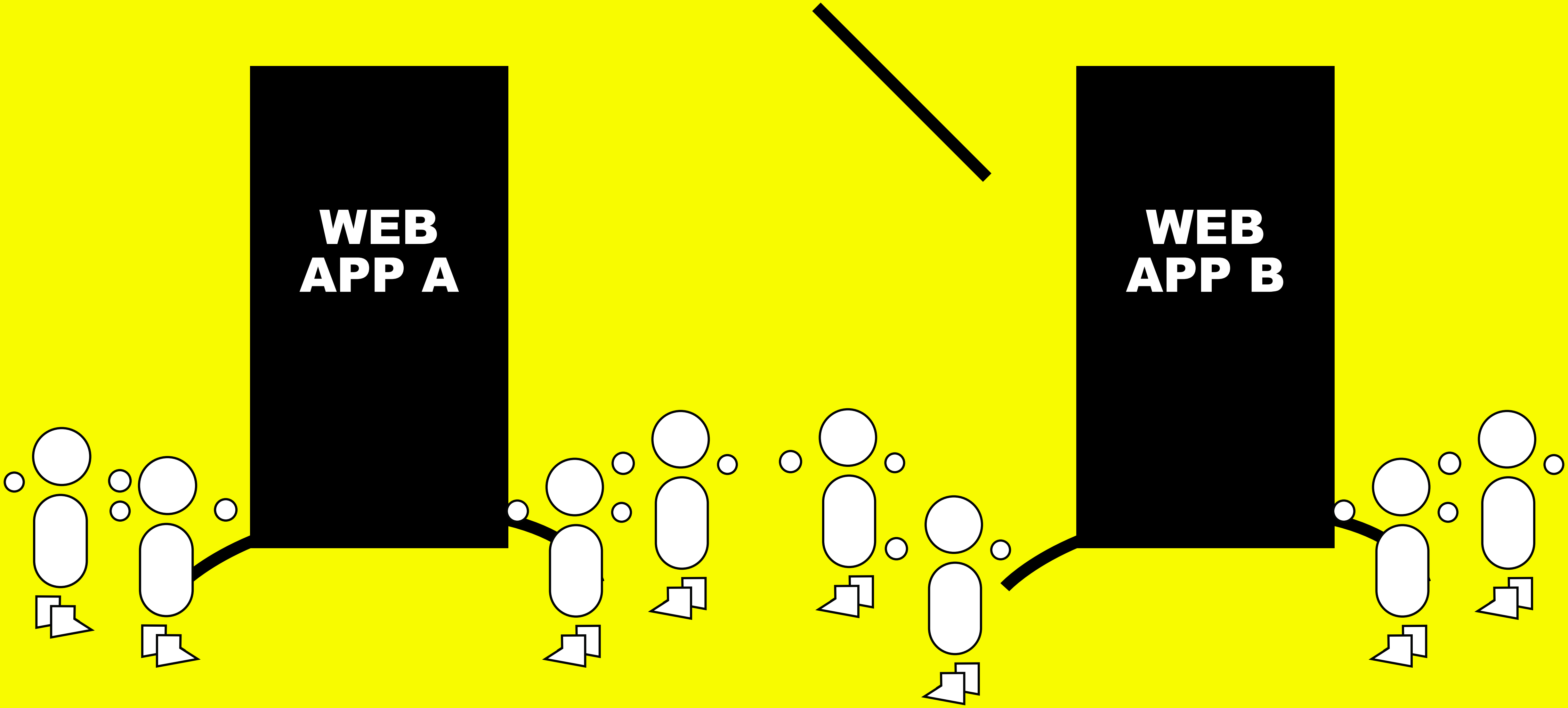

WEB
APP

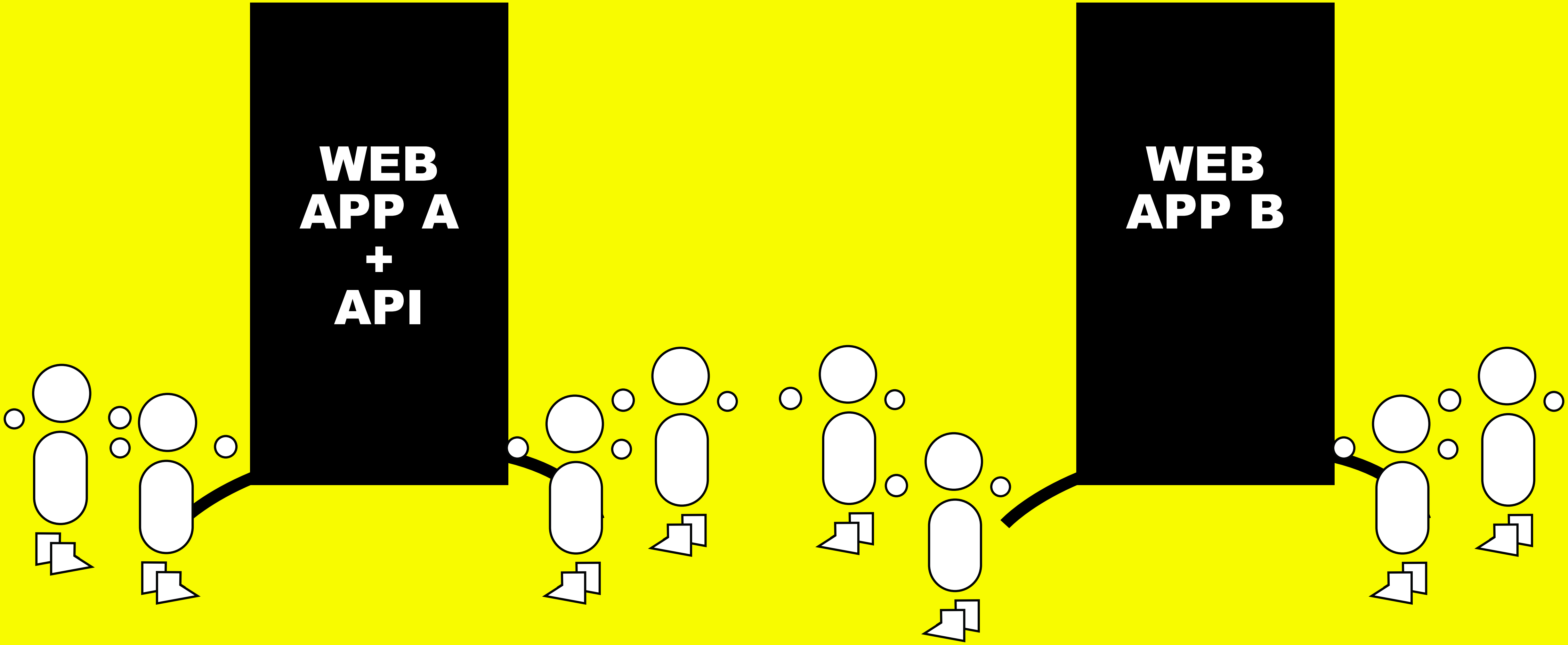


WEB
APP A

WEB
APP B

“HEY, MY USERS WANT TO ACCESS YOUR STUFF.”





WEB
APP A
+
API

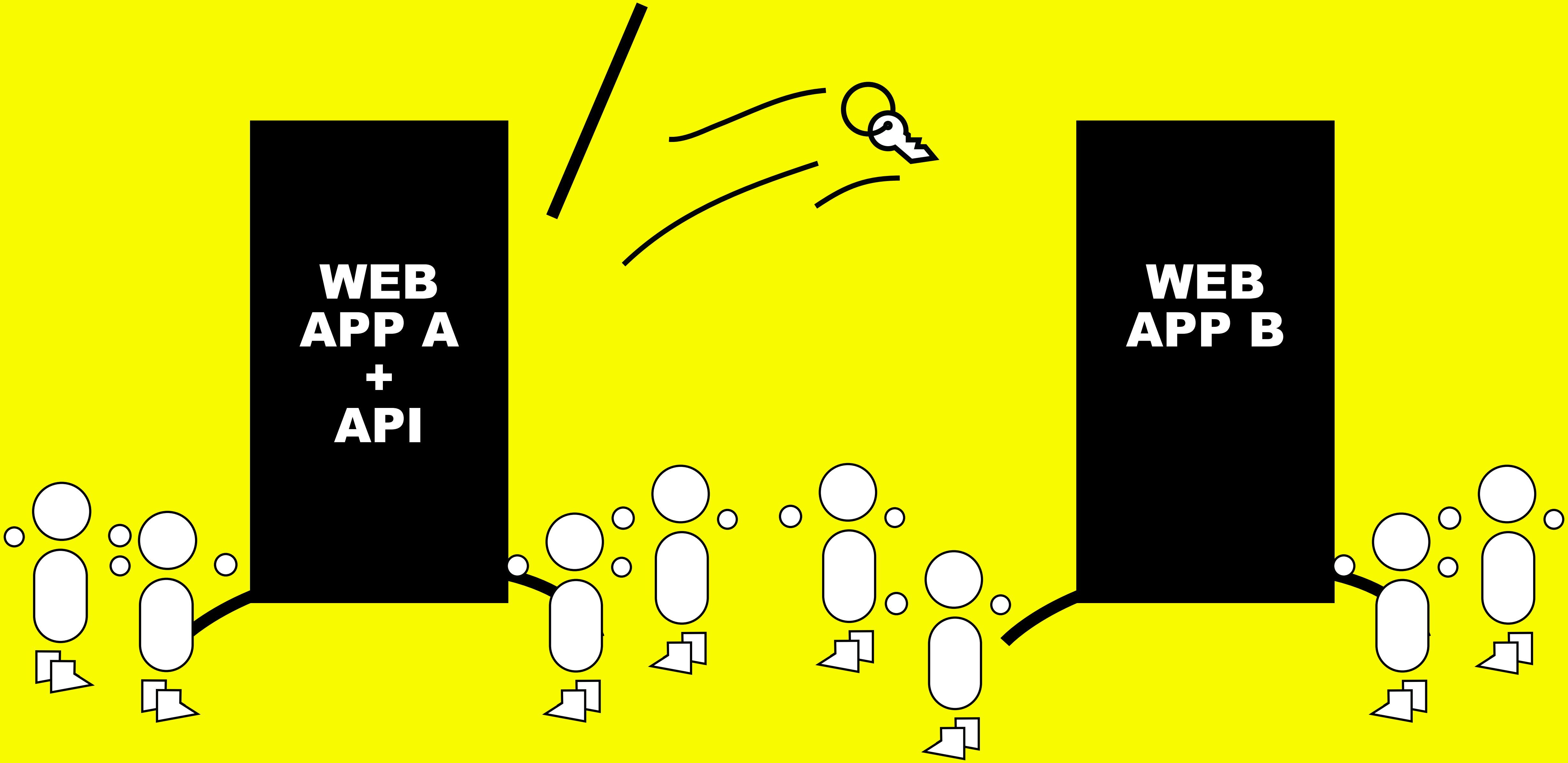
WEB
APP B

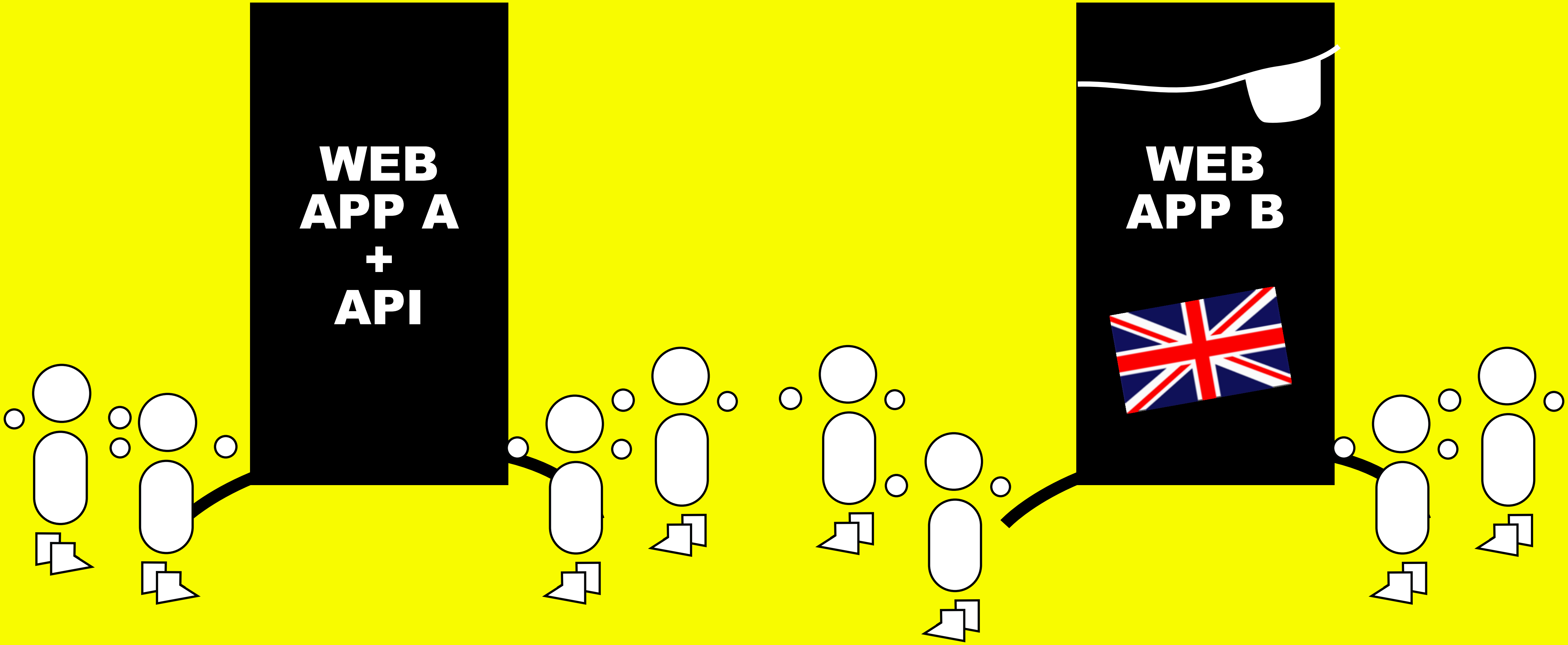
HTTP BASIC

http://user:password@...

Authorization: Basic dXNlcjpwYXNzd29yZA==

OK, HERE’S THE KEYS.




WEB
APP A
+
API
WEB
APP B

Gmail
by Google
BETA
Bank of America
WEB
APP A
+
API
WEB
APP B

FUBAR

FAILED USER BAR FOR AUTHORIZATION ROBUSTNESS

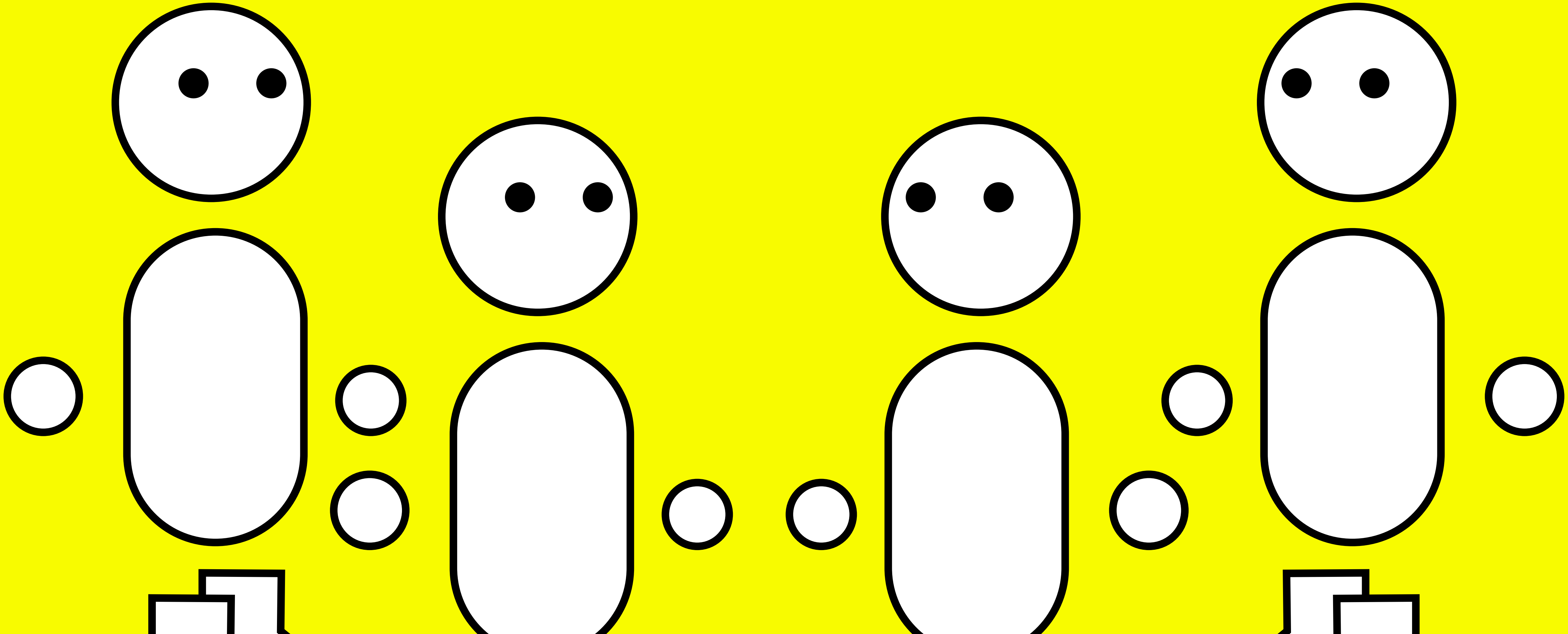
COUGH

THIS IS A PROBLEM

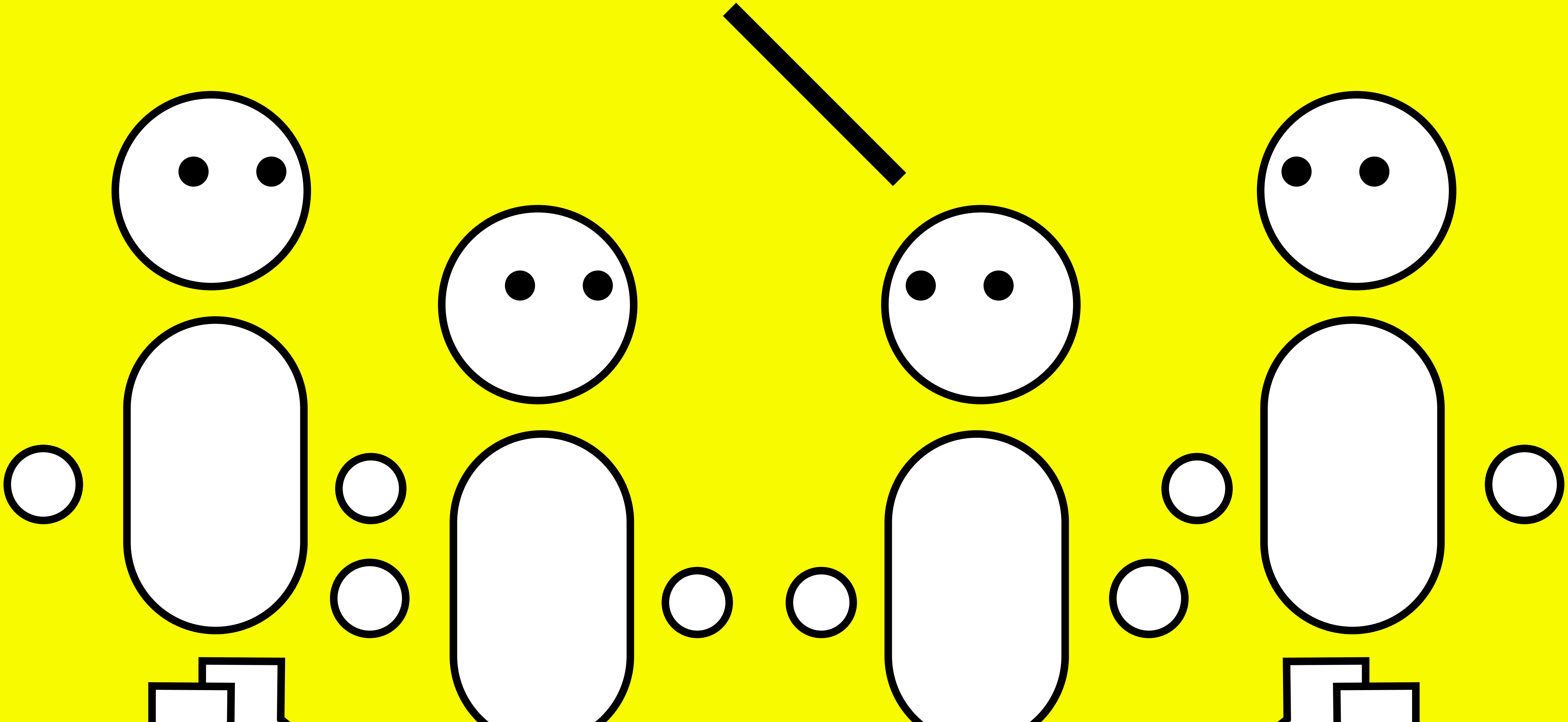
ACT 2

IN WHICH A NEW WAY IS CREATED

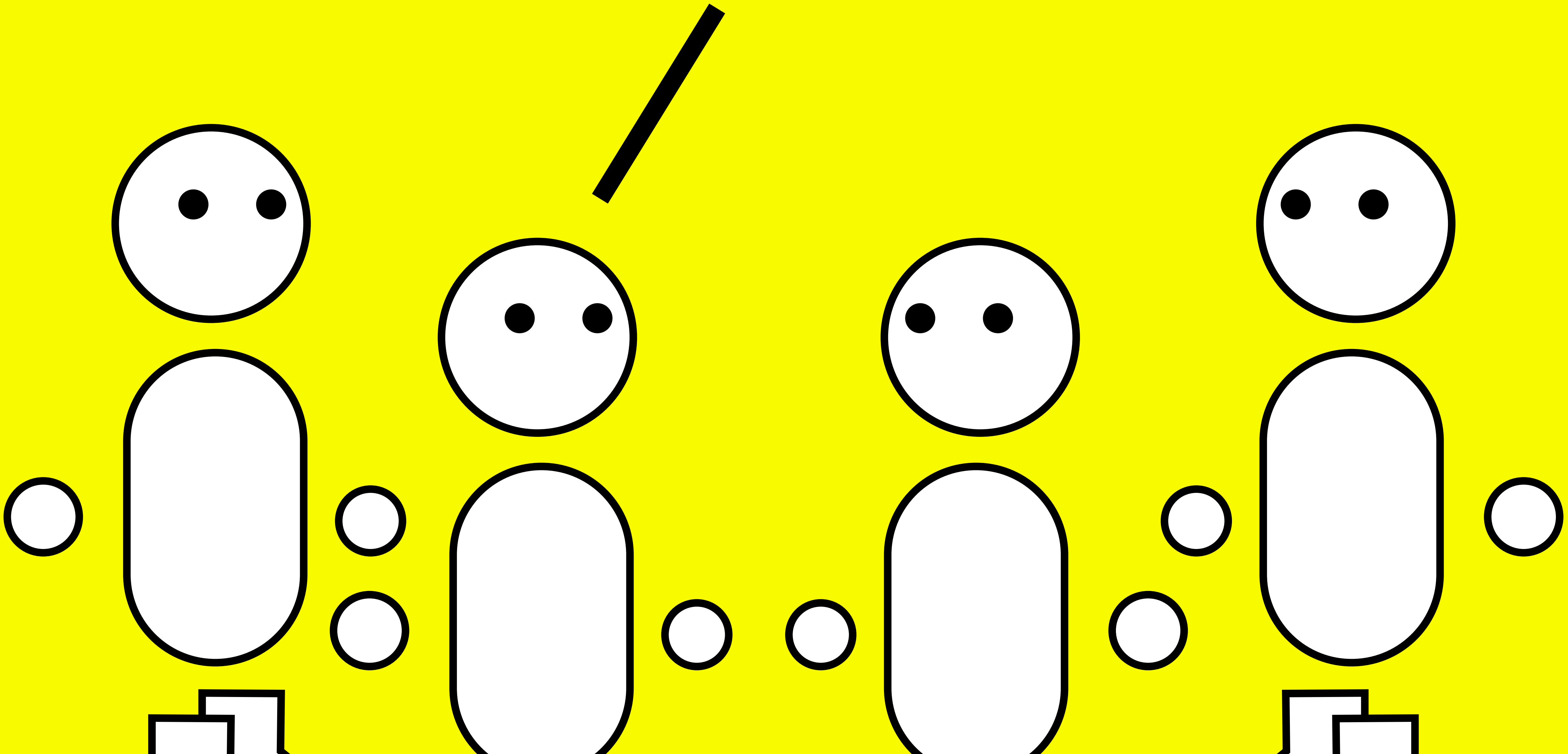
CHRIS MESSINA
BLAINE COOK
LARRY HALFF
DAVID RECORDON



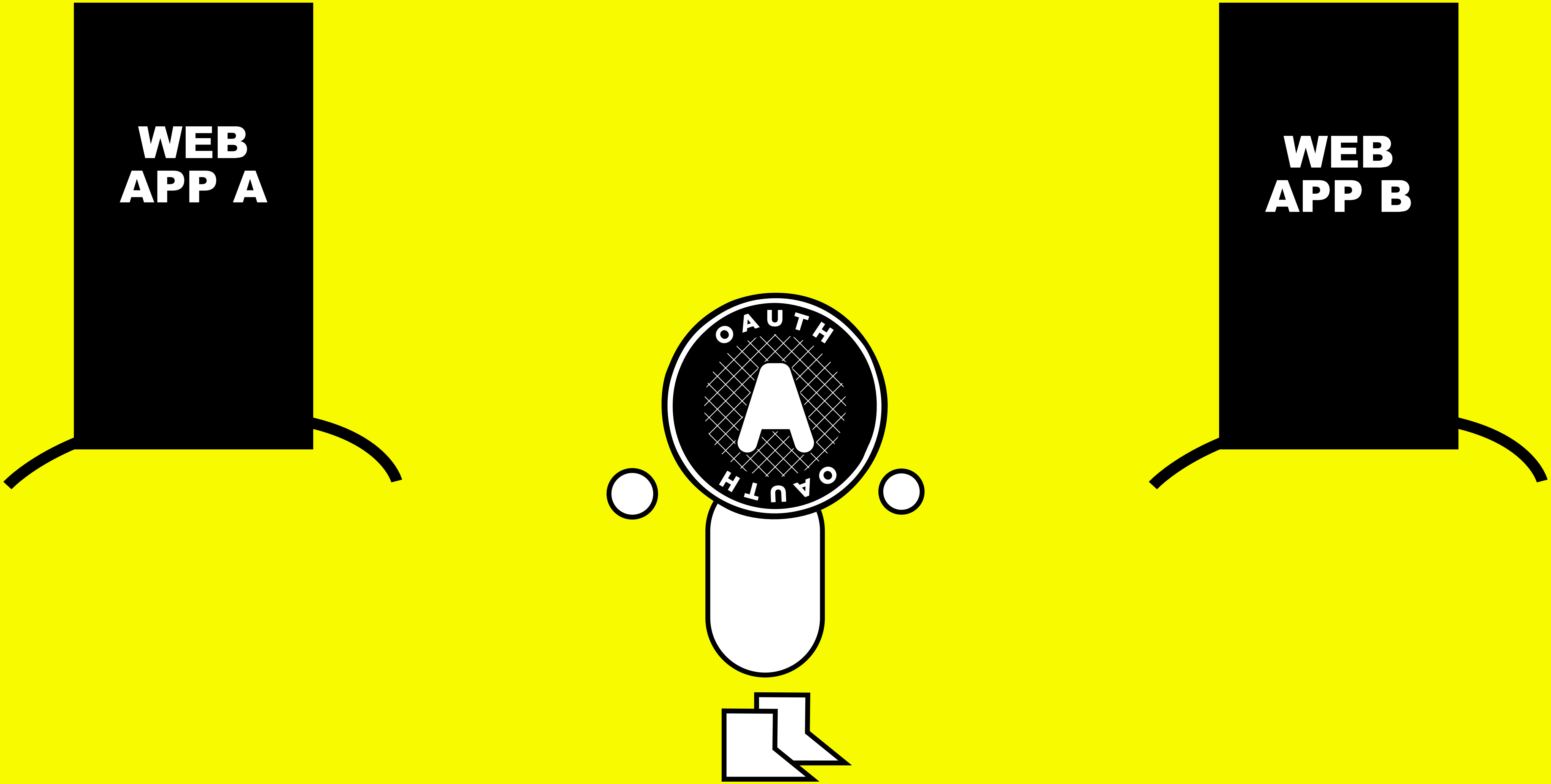
“HEY, WOULDN’T IT BE GREAT TO HAVE AN OPEN AUTHORIZATION STANDARD”

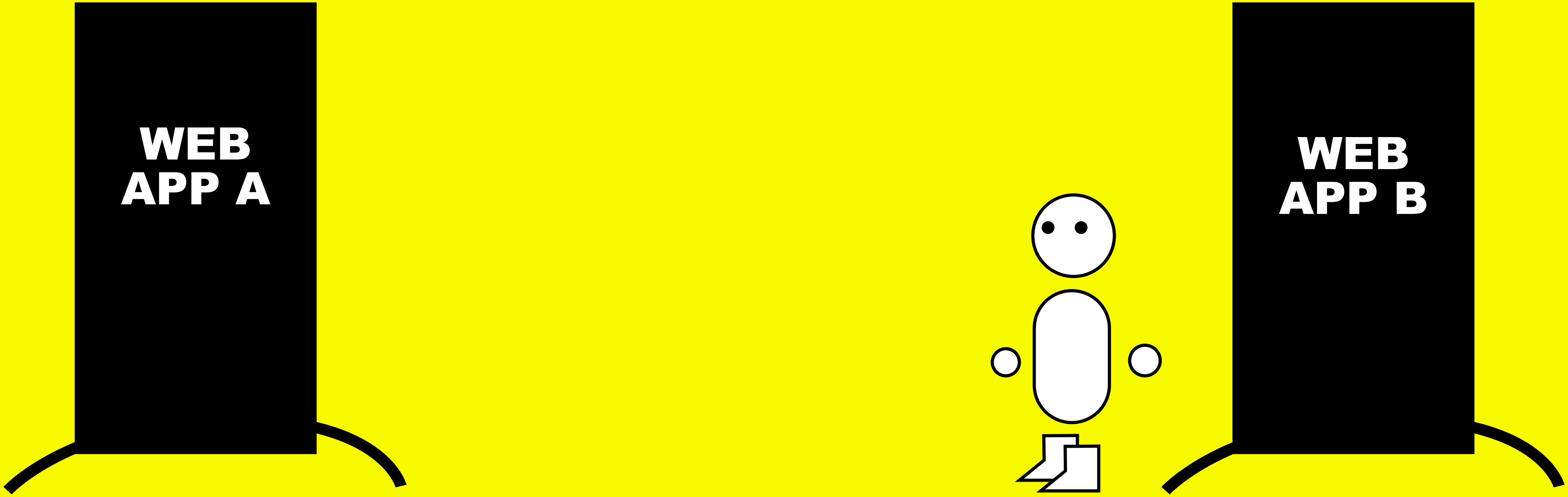


“TOTALLY, LET’S MAKE ONE AND CALL IT OAUTH.”

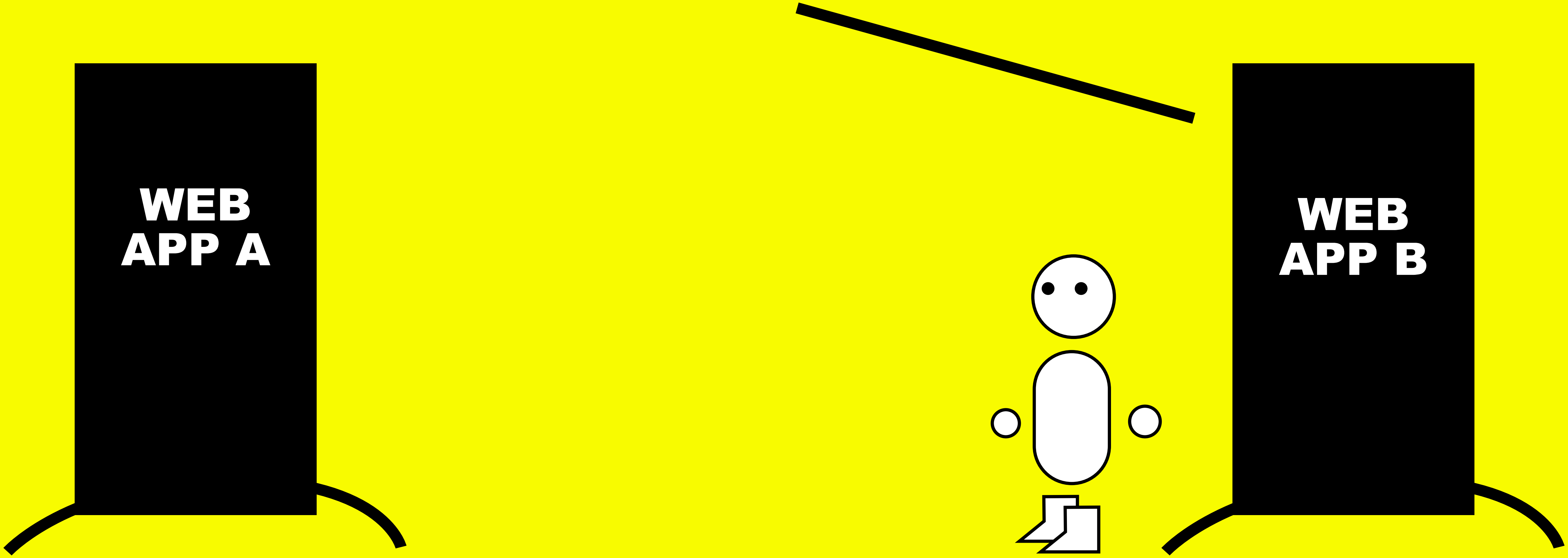


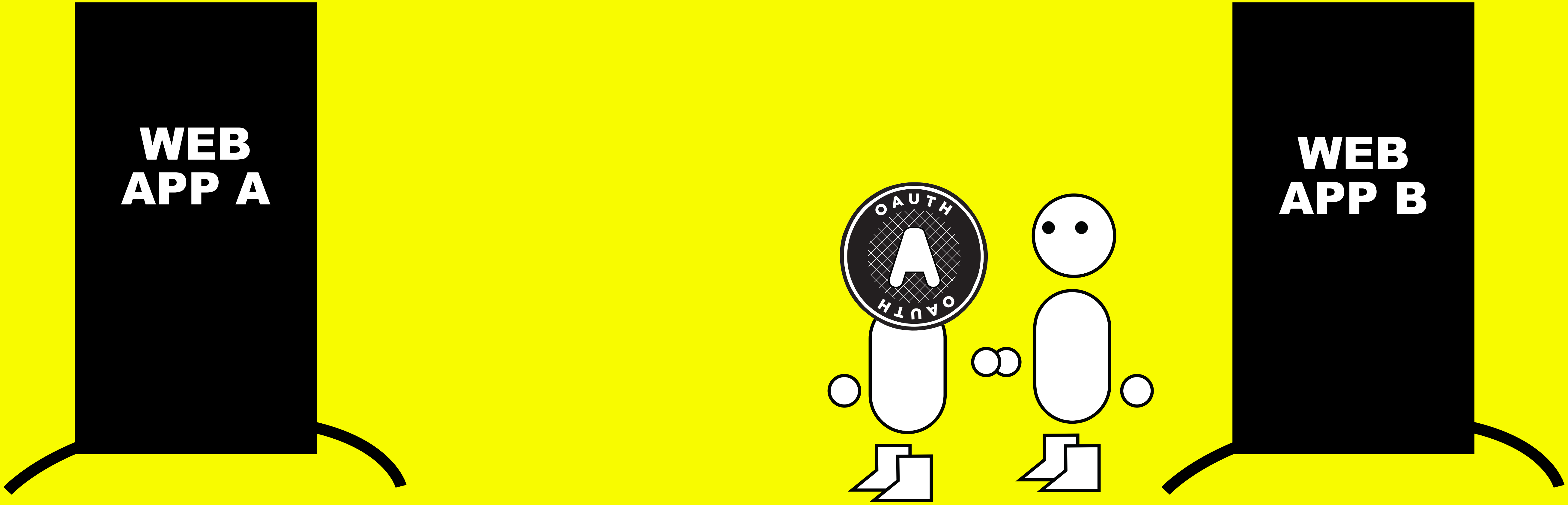
FOOTAGE
MISSING

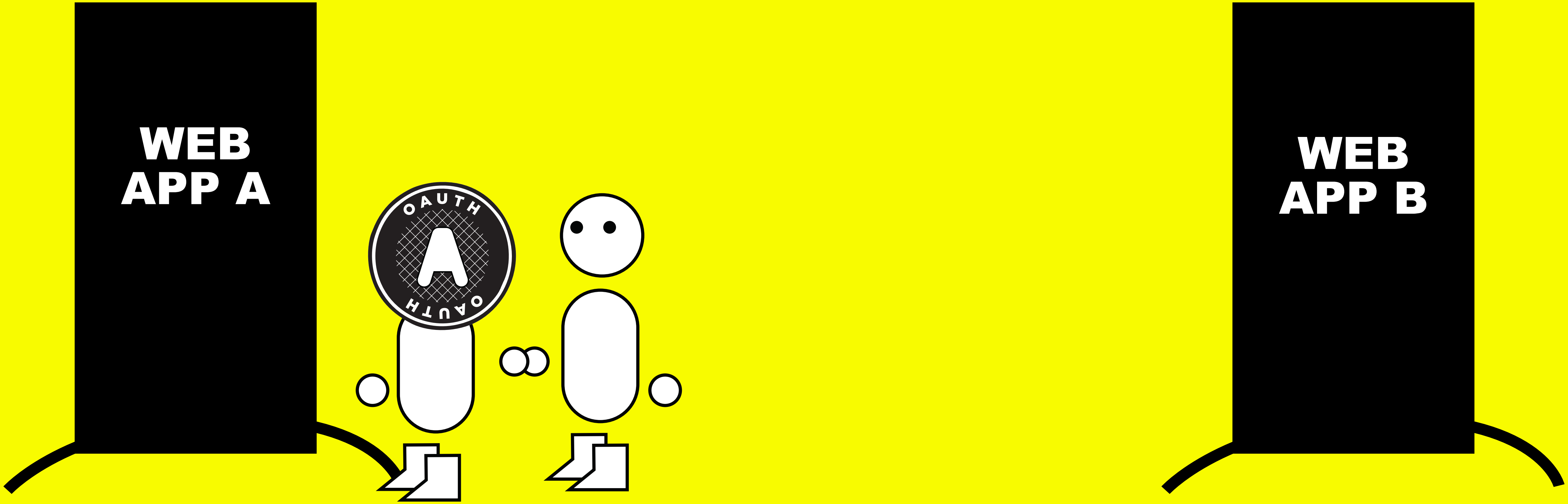

WEB APP A
WEB APP B
OAUTH
OAUTH

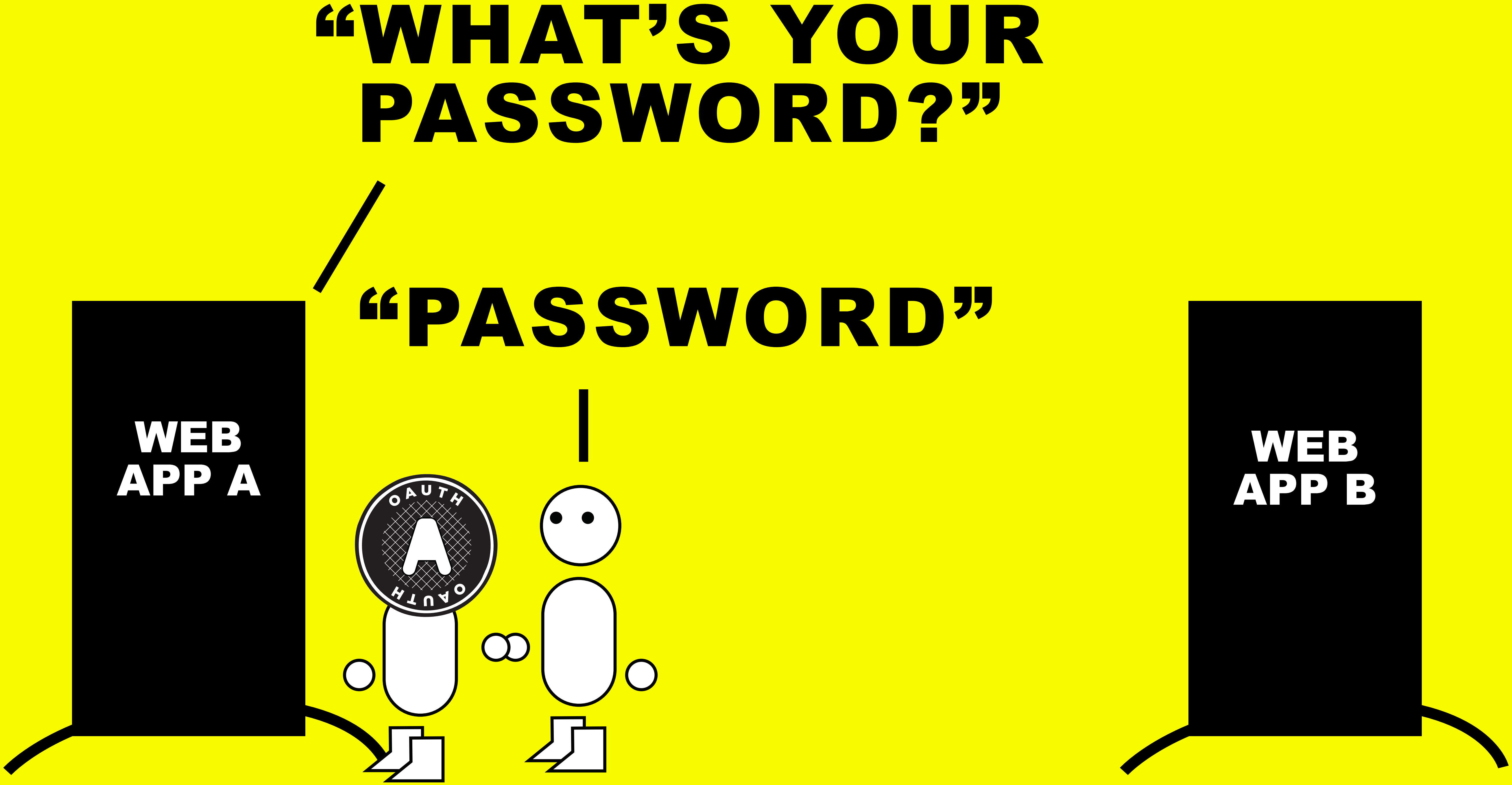

WEB
APP A
WEB
APP B

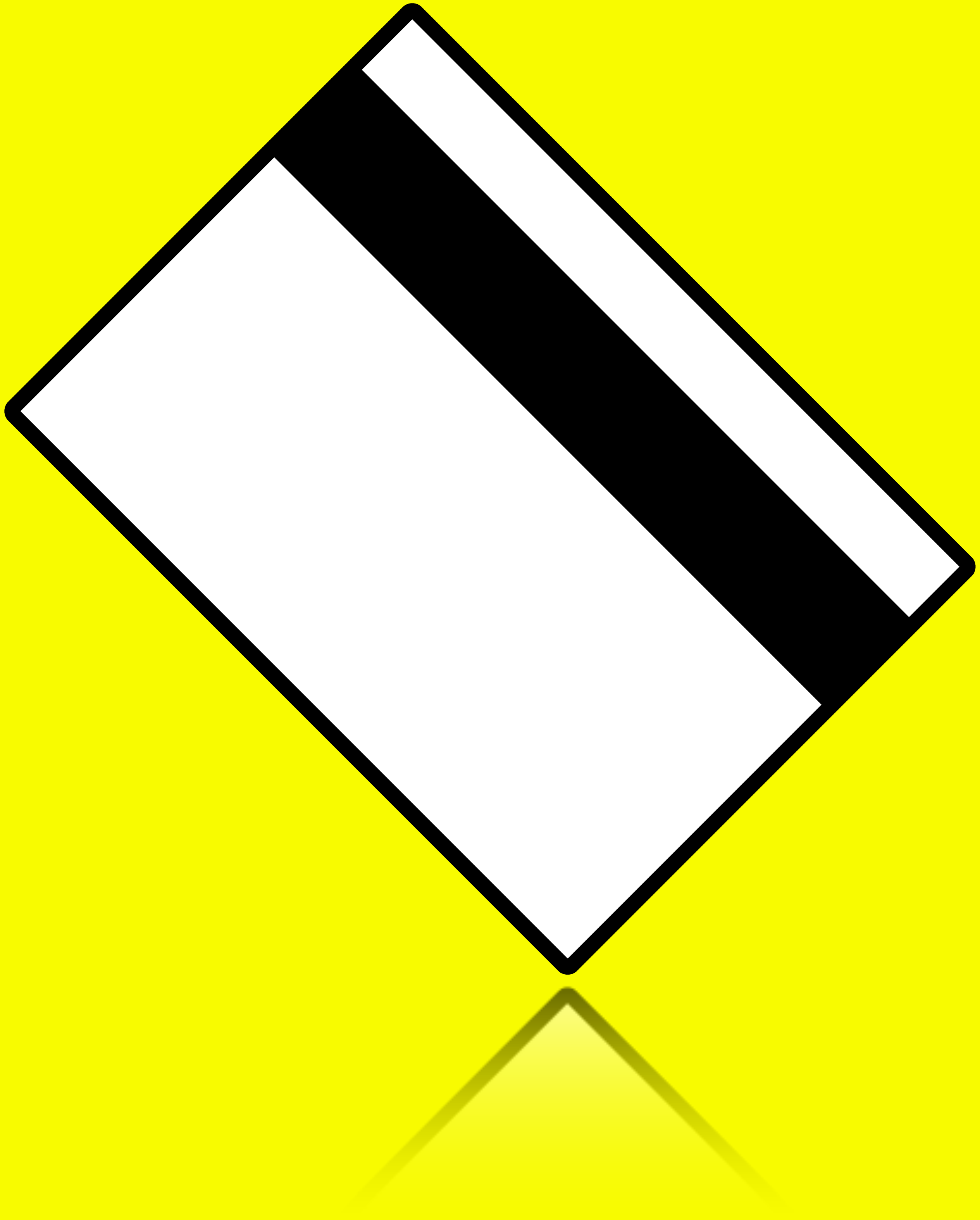
“HEY, MY USER WANTS TO ACCESS YOUR STUFF.”

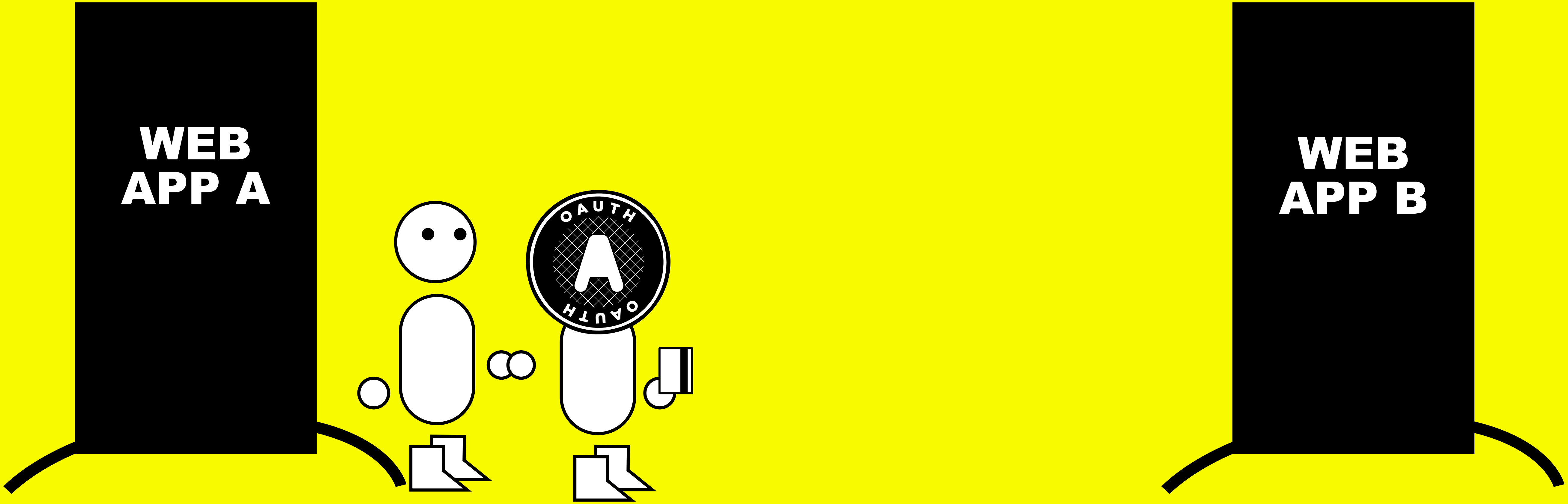


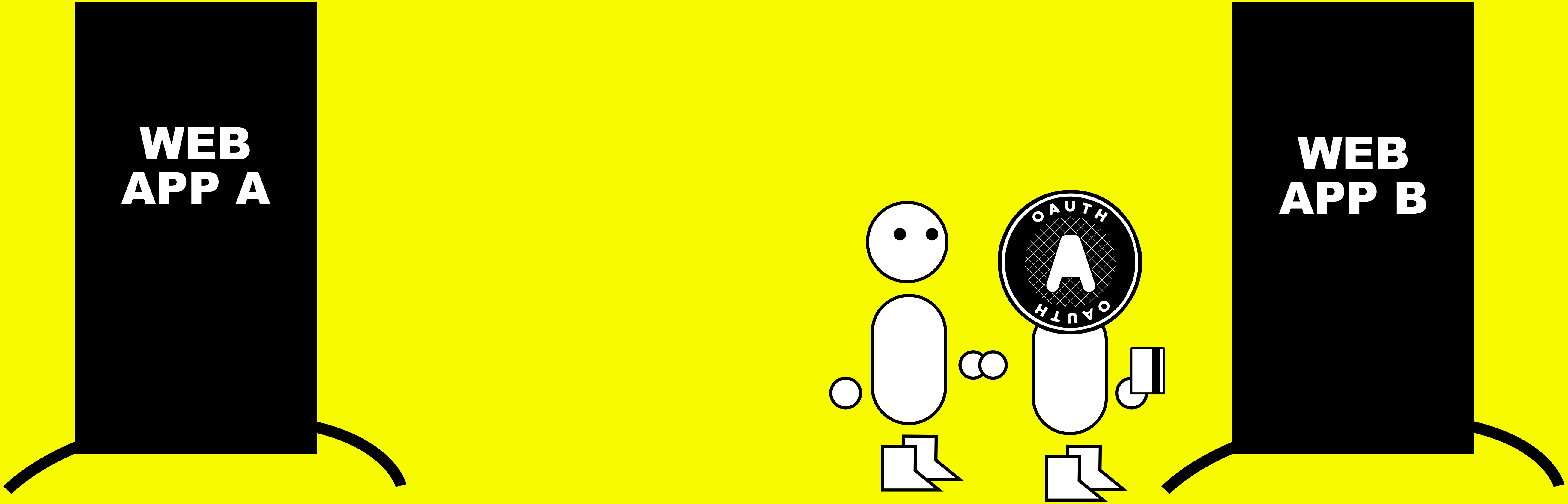

WEB
APP A
OAUTH
A
OAUTH
WEB
APP B


WEB
APP A
OAUTH
OAUTH
WEB
APP B


“WHAT’S YOUR PASSWORD?”
“PASSWORD”
WEB APP A
OAUTH
A
OAUTH
WEB APP B

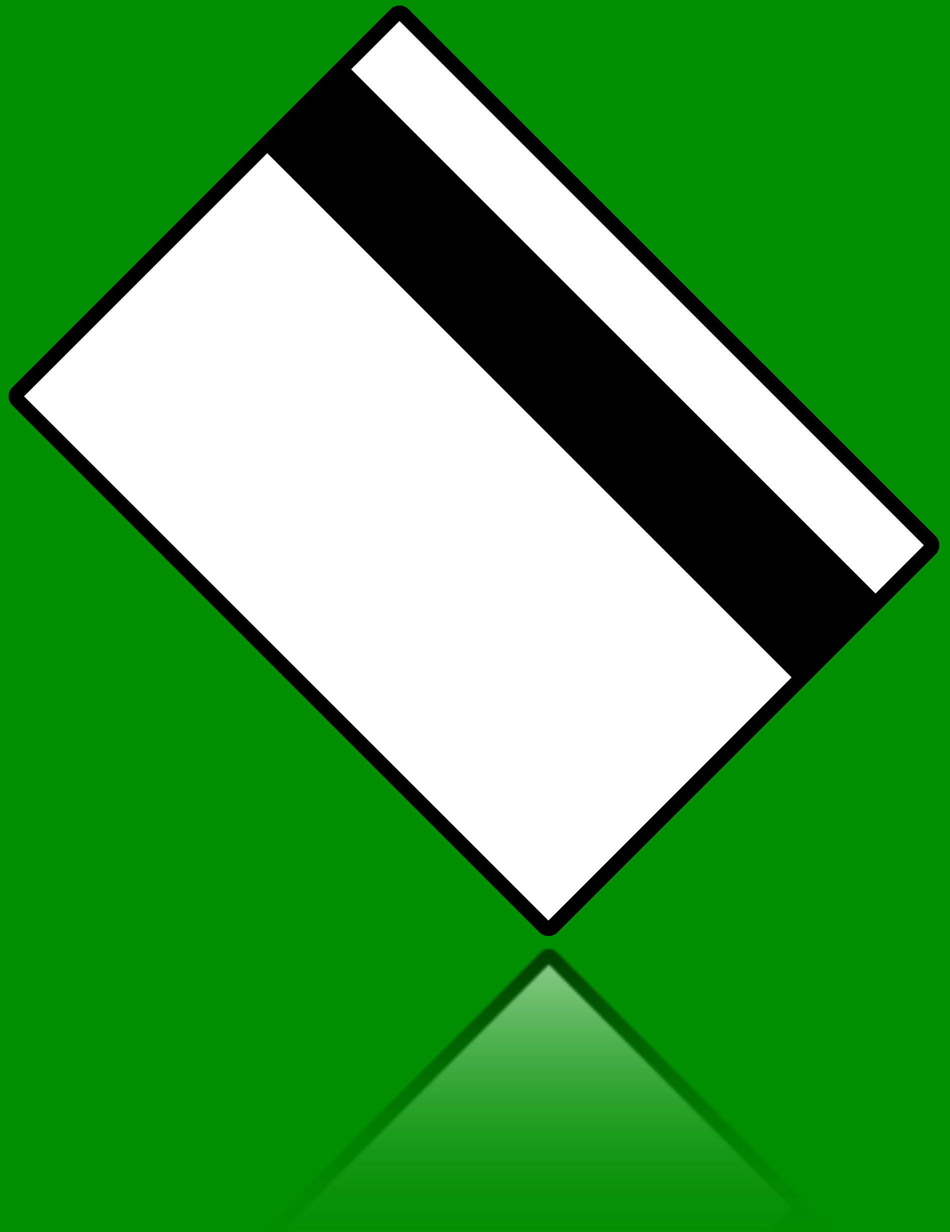



WEB
APP A
OAUTH
A
OAUTH
WEB
APP B

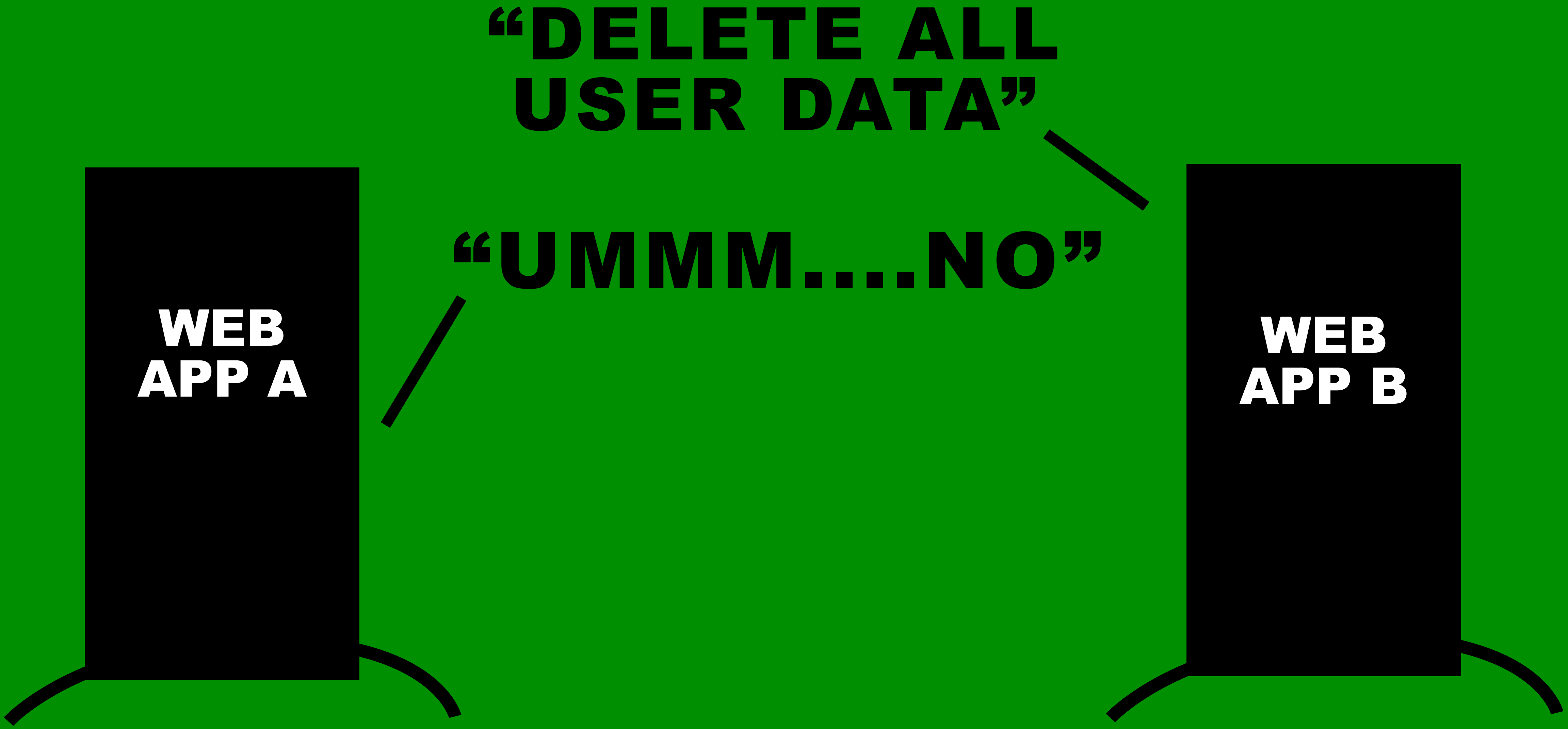

WEB APP A
OAUTH
A
OAUTH
WEB APP B

ADVANTAGES

1. SECURE



2. RESTRICTABLE



"DELETE ALL USER DATA"

"UMMM....NO"

WEB APP A

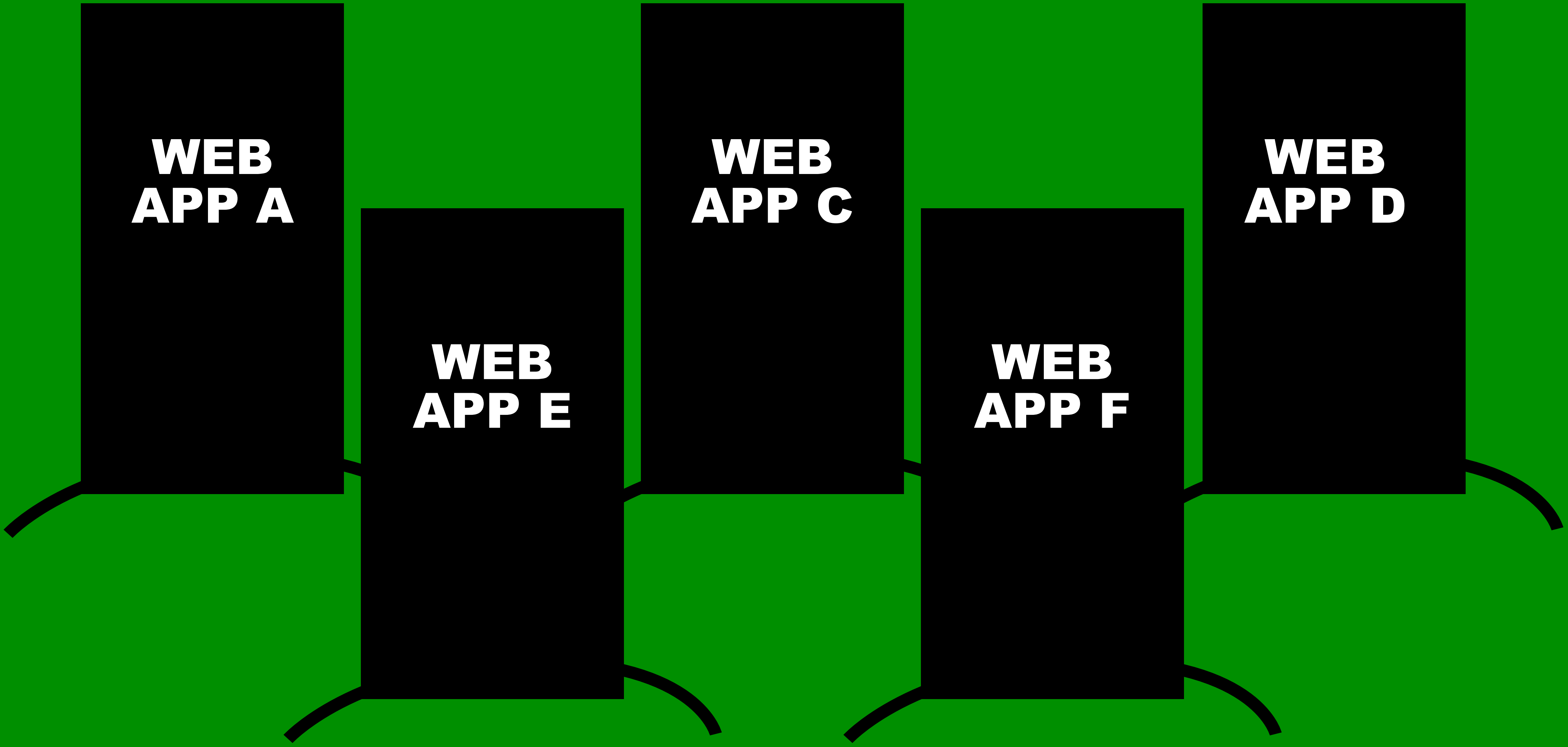
WEB APP B

3. REVOCABLE

YOINK

WEB APP B

3. STANDARD



WEB APP A

WEB APP E

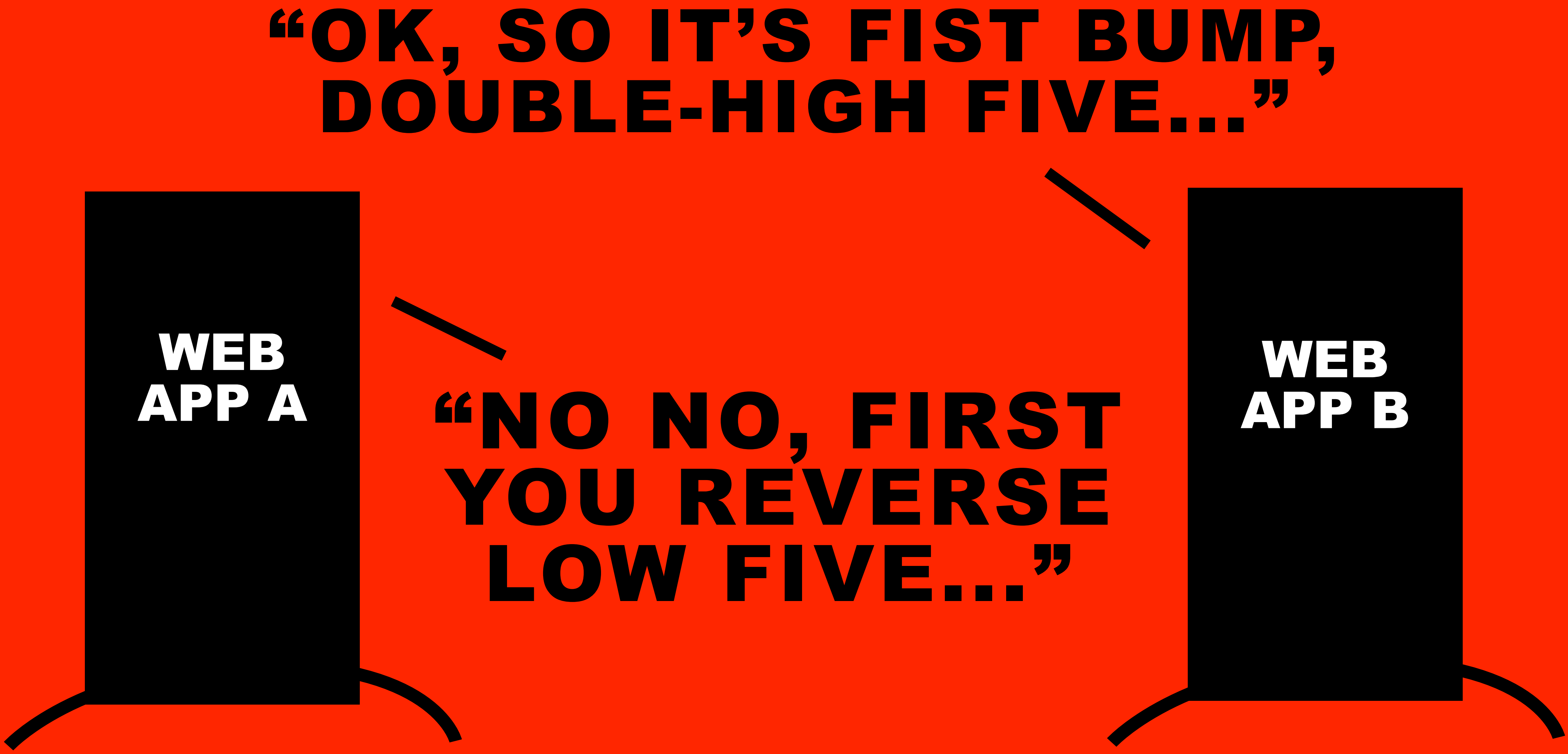
WEB APP C

WEB APP F

WEB APP D

NOT QUITE PERFECT

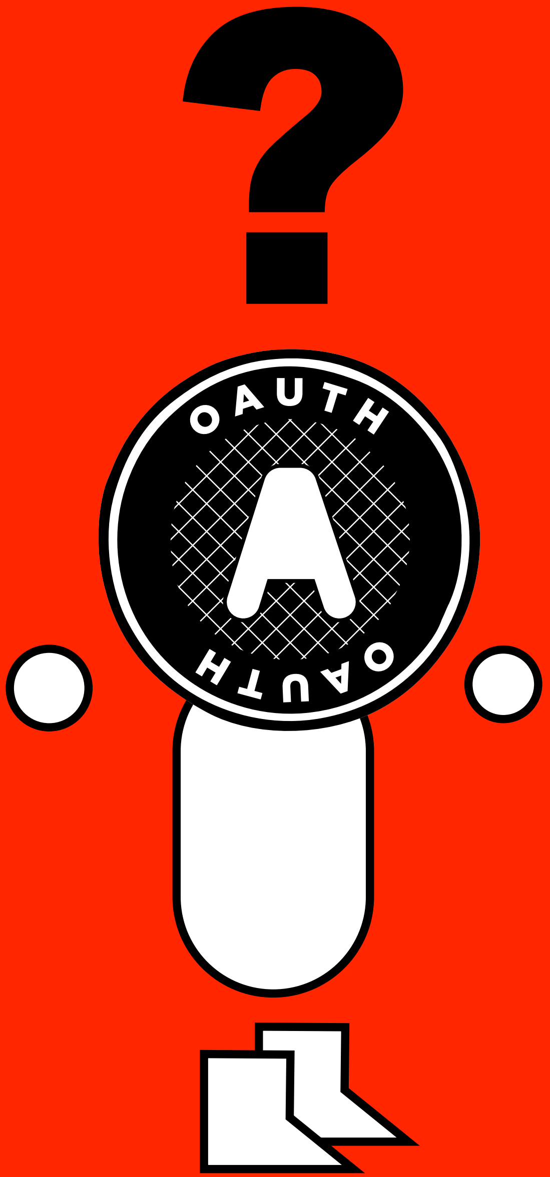
1. COMPLICATED

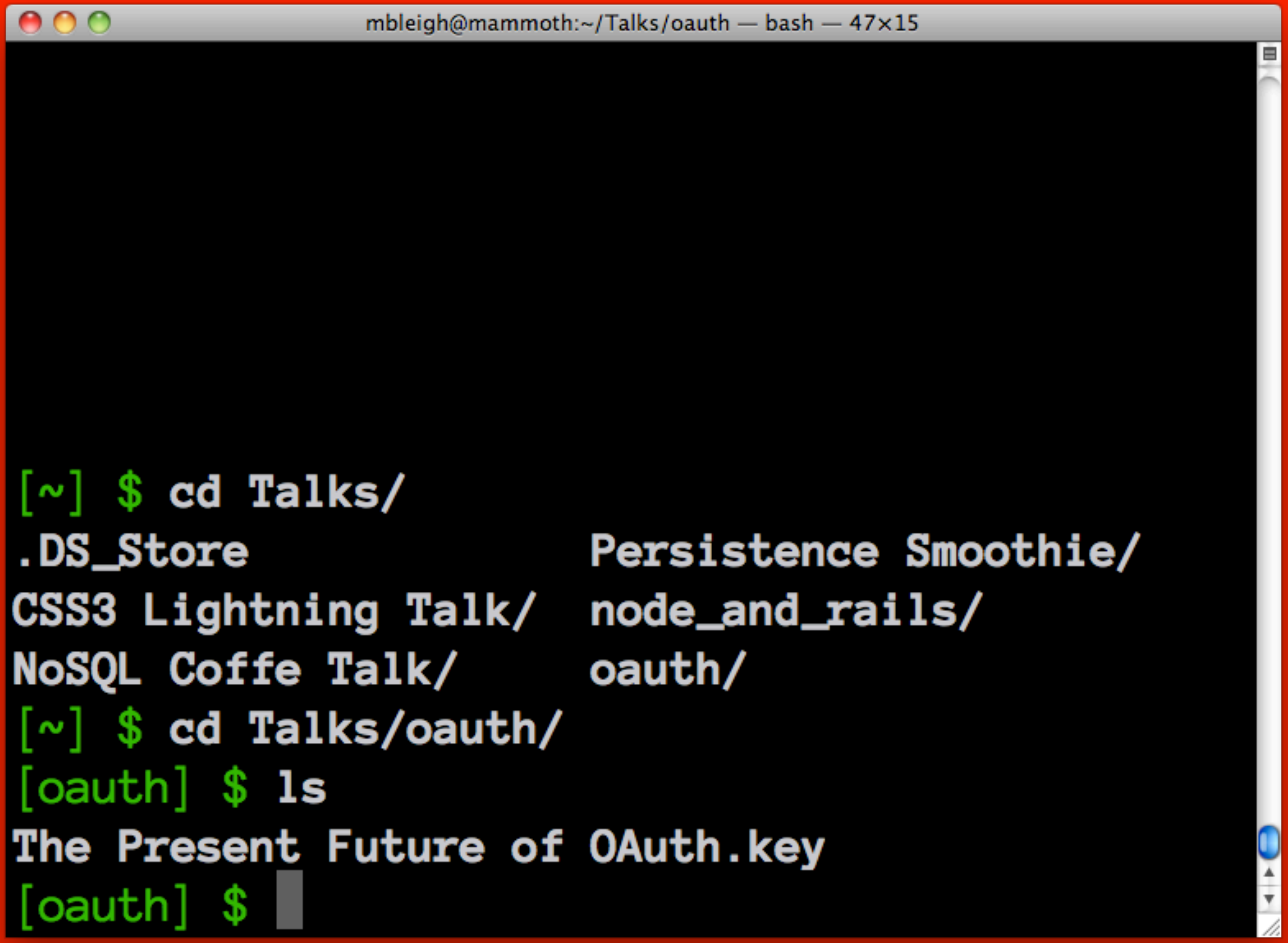


"OK, SO IT'S FIST BUMP, DOUBLE-HIGH FIVE..."

"NO NO, FIRST YOU REVERSE LOW FIVE..."

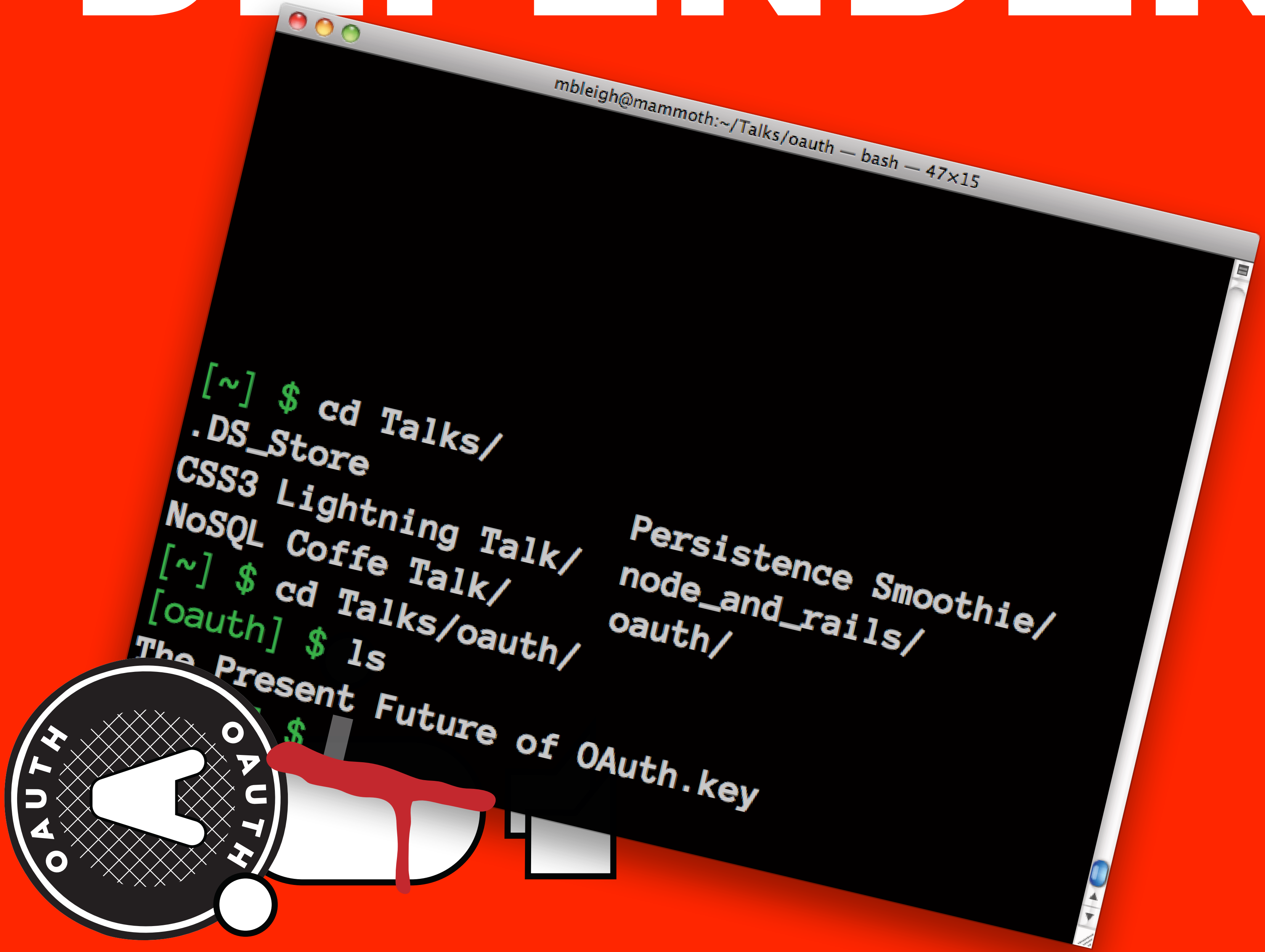
2. BROWSER-DEPENDENT





```
[~] $ cd Talks/
.DS_Store                  Persistence Smoothie/
CSS3 Lightning Talk/  node_and_rails/
NoSQL Coffe Talk/       oauth/
[~] $ cd Talks/oauth/
[oauth] $ ls
The Present Future of OAuth.key
[oauth] $
```

2. BROWSER-DEPENDENT

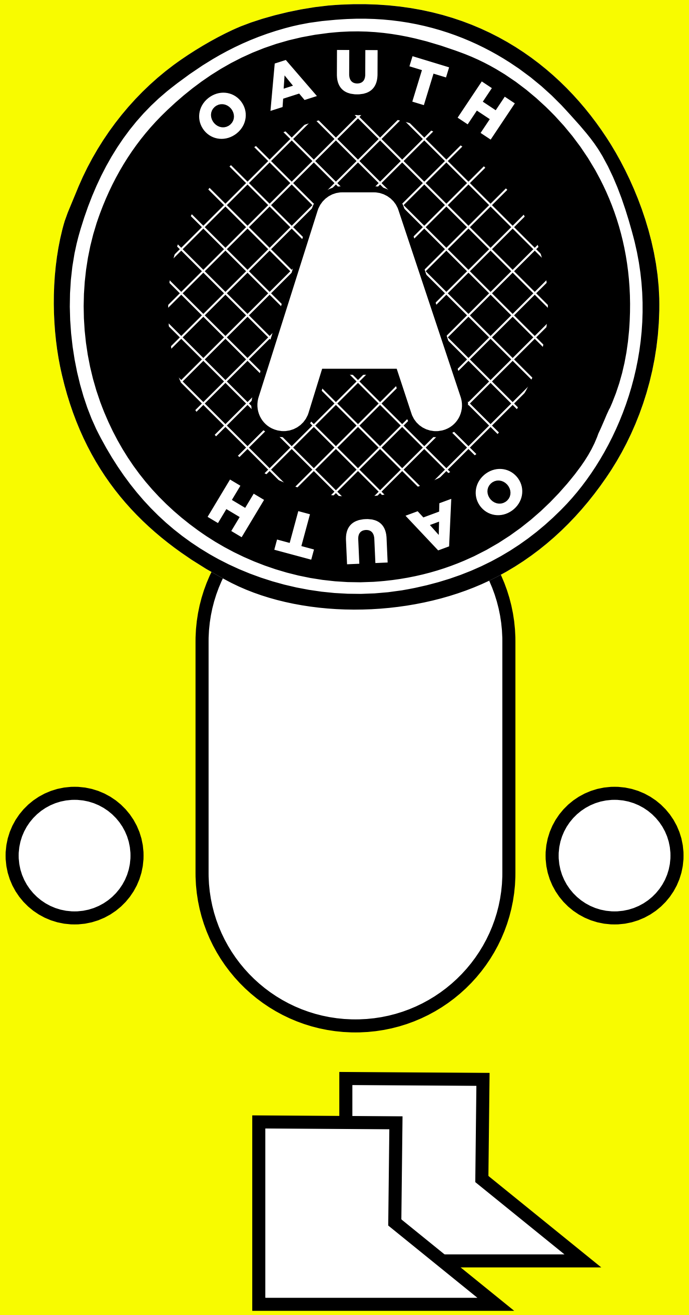


```
[~] $ cd Talks/
.DS_Store                   Persistence Smoothie/
CSS3 Lightning Talk/        node_and_rails/
NoSQL Coffe Talk/           oauth/
[~] $ cd Talks/oauth/
[oauth] $ ls
The Present Future of OAuth.key
$
```

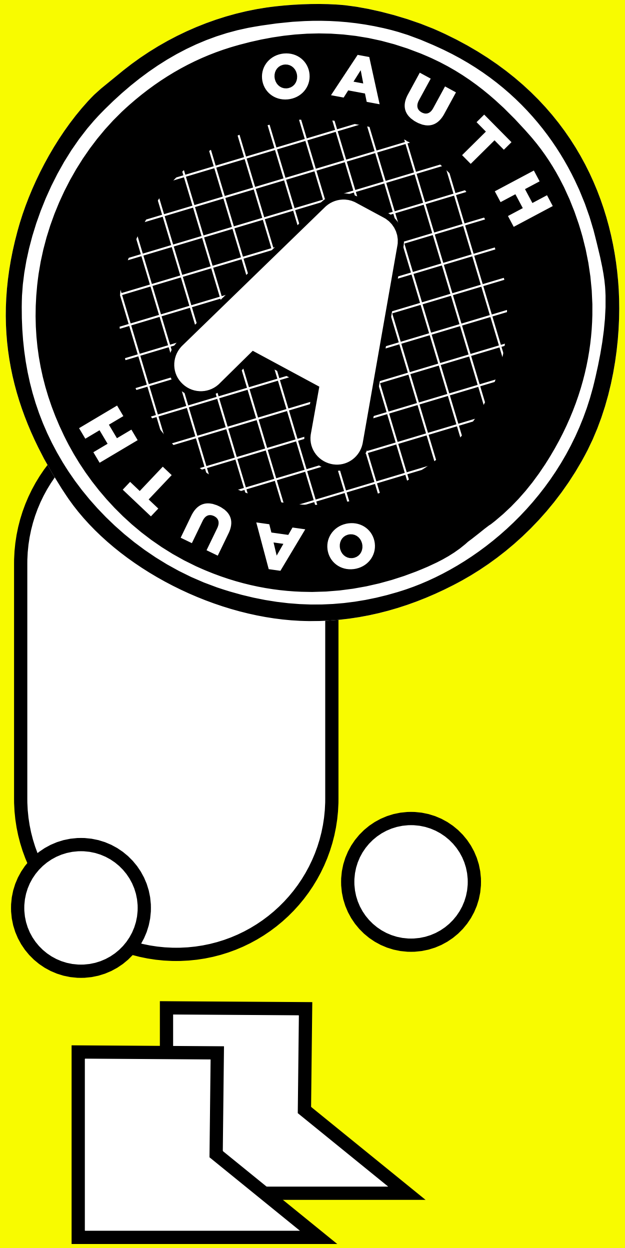
WE CAN DO BETTER

ACT 3

IN WHICH WE LEARN FROM OUR MISTAKES



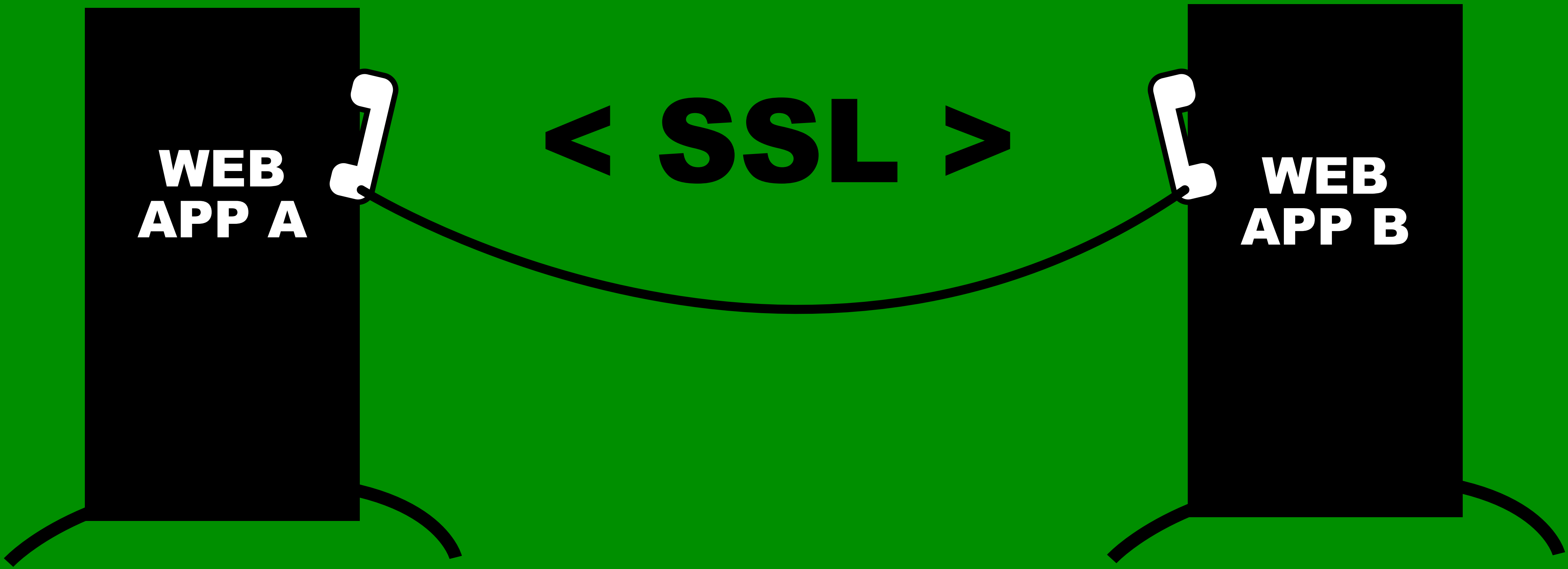
OAUTH
A
OAUTH


OAUTH
OAUTH

OAUTH 2.0

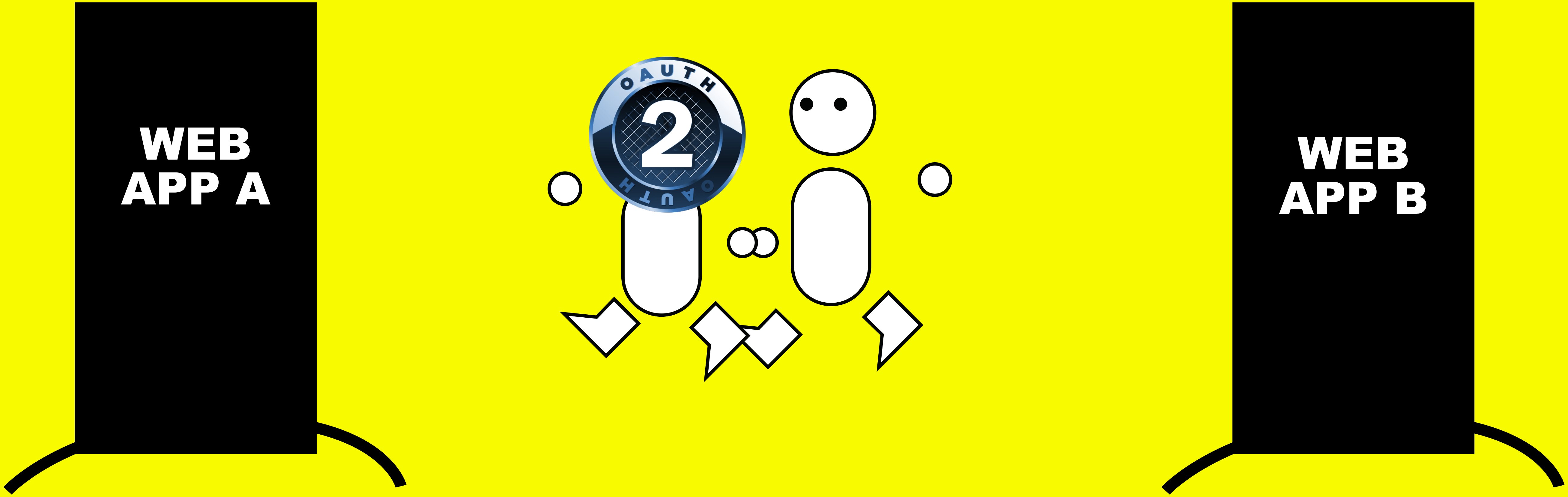
IMPROVEMENTS

1. SIMPLER



2. FLOWS

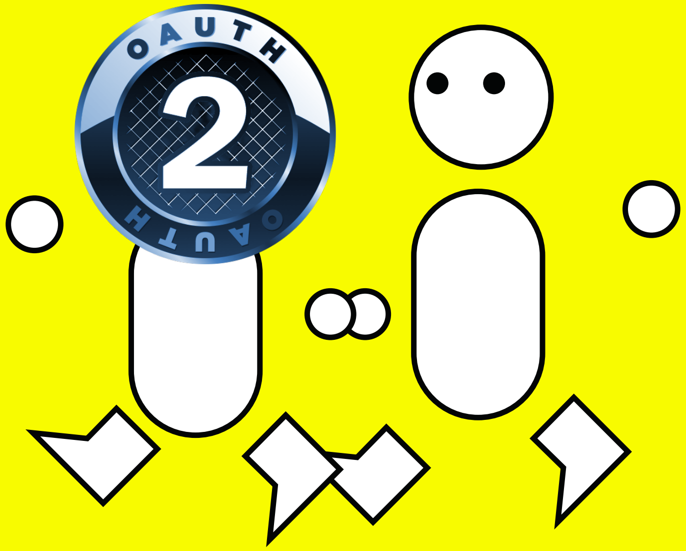
WEB SERVER



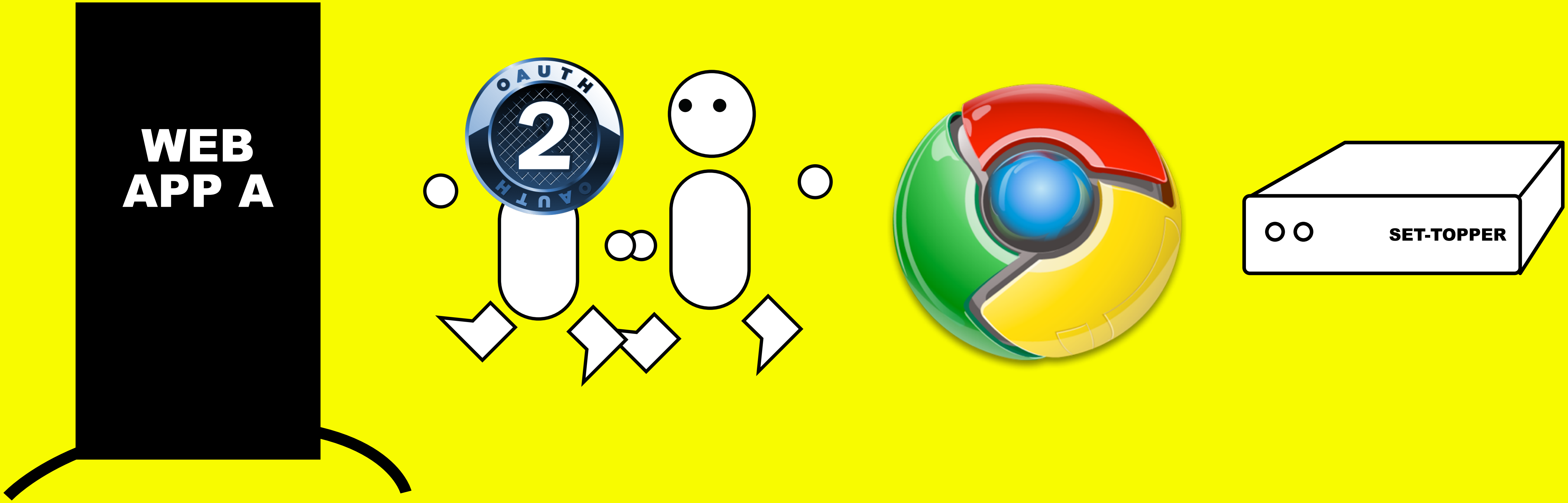
WEB APP A

WEB APP B

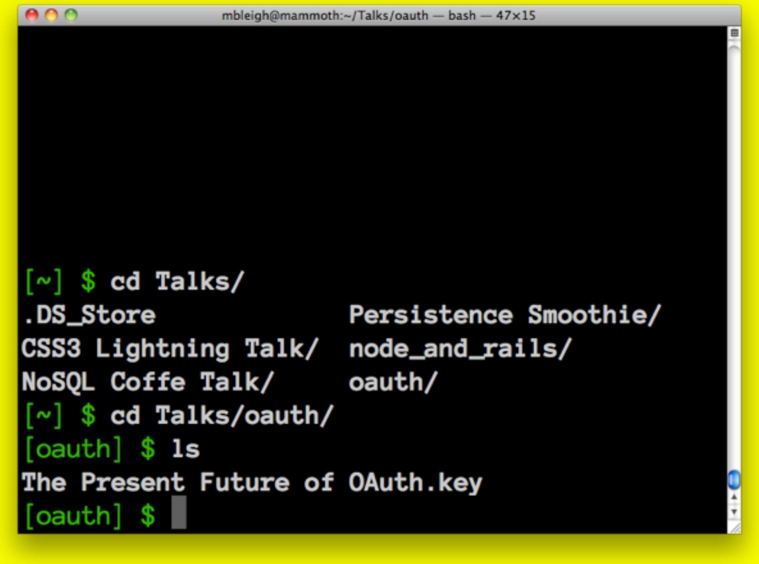
USER-AGENT



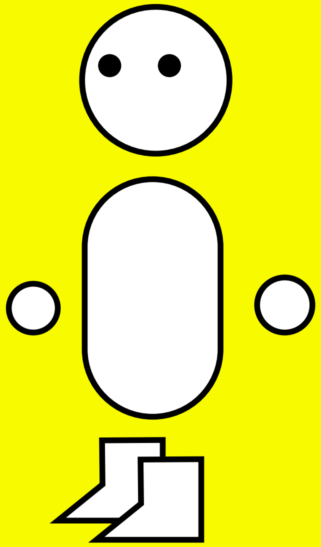
DEVICE



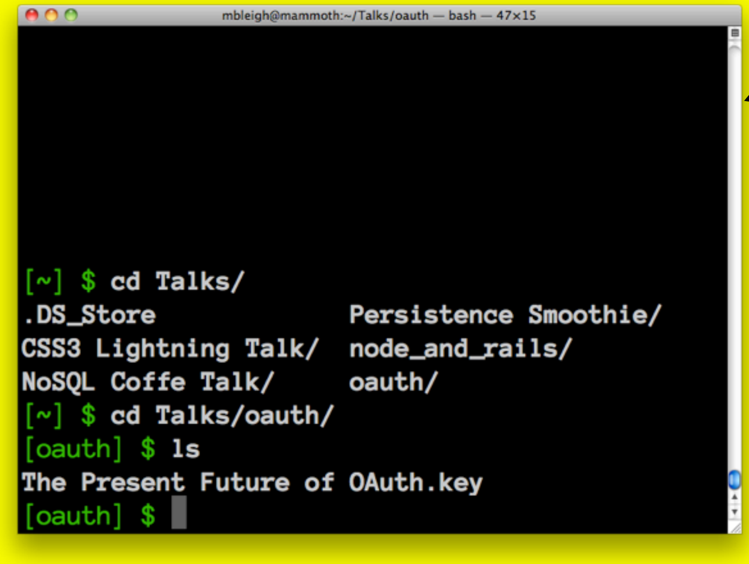
PASSWORD



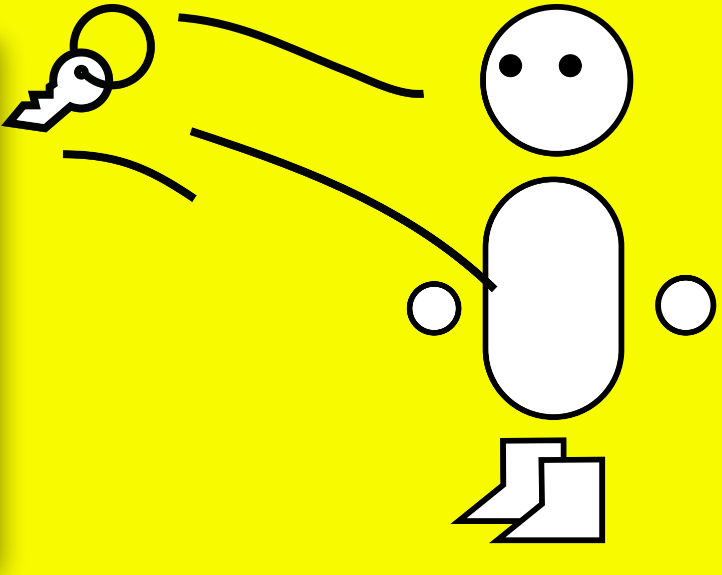
```
[~] $ cd Talks/
.DS_Store             Persistence Smoothie/
CSS3 Lightning Talk/  node_and_rails/
NoSQL Coffe Talk/     oauth/
[~] $ cd Talks/oauth/
[oauth] $ ls
The Present Future of OAuth.key
[oauth] $ 
```



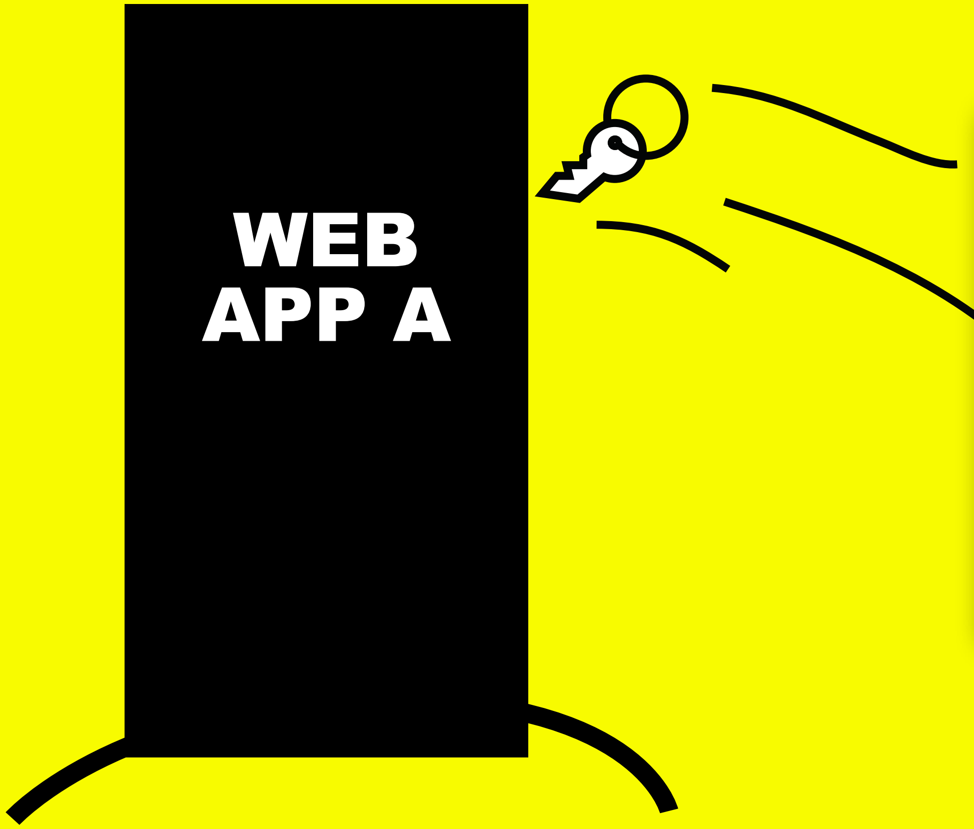
PASSWORD

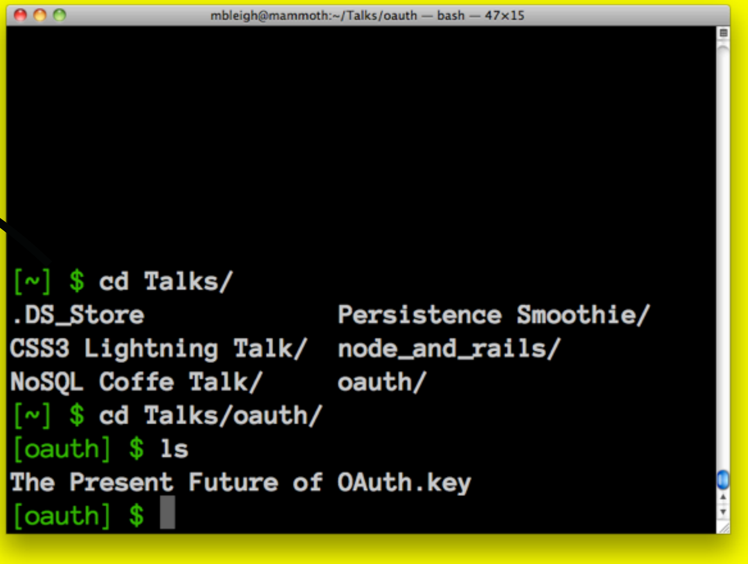


```
[~] $ cd Talks/
.DS_Store             Persistence Smoothie/
CSS3 Lightning Talk/  node_and_rails/
NoSQL Coffe Talk/     oauth/
[~] $ cd Talks/oauth/
[oauth] $ ls
The Present Future of OAuth.key
[oauth] $
```

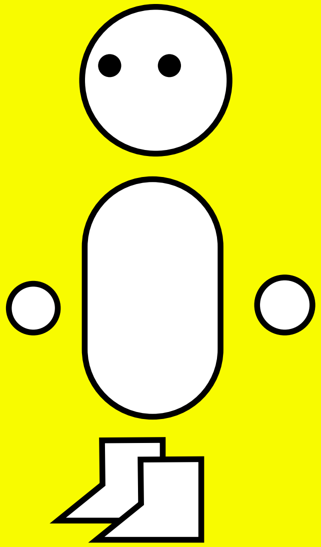


PASSWORD

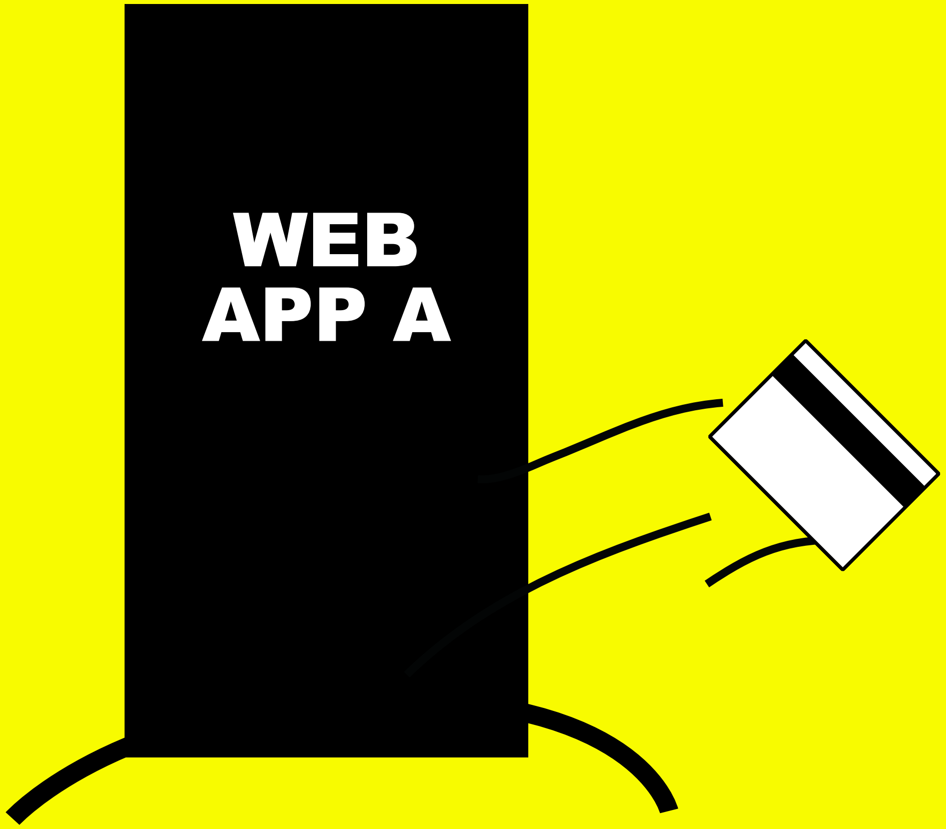


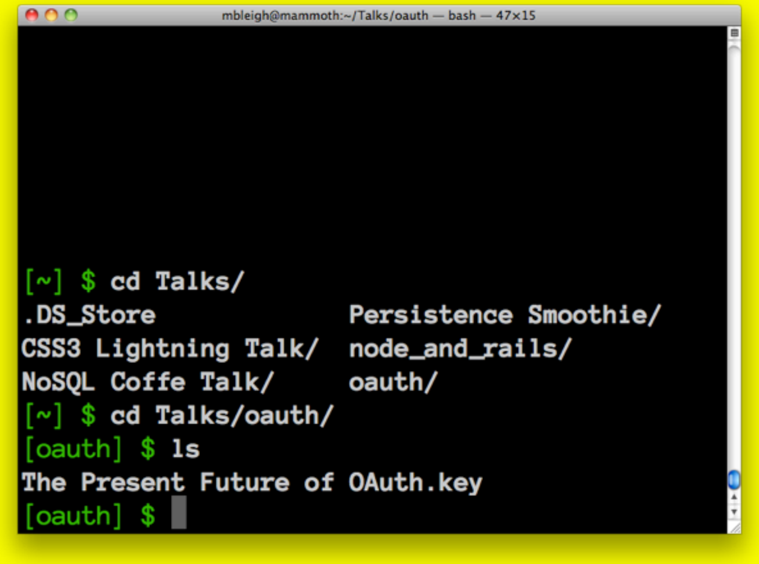


```
[~] $ cd Talks/
.DS_Store              Persistence Smoothie/
CSS3 Lightning Talk/  node_and_rails/
NoSQL Coffe Talk/      oauth/
[~] $ cd Talks/oauth/
[oauth] $ ls
The Present Future of OAuth.key
[oauth] $ 
```

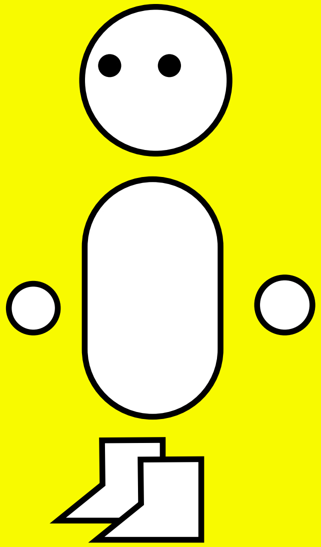


PASSWORD

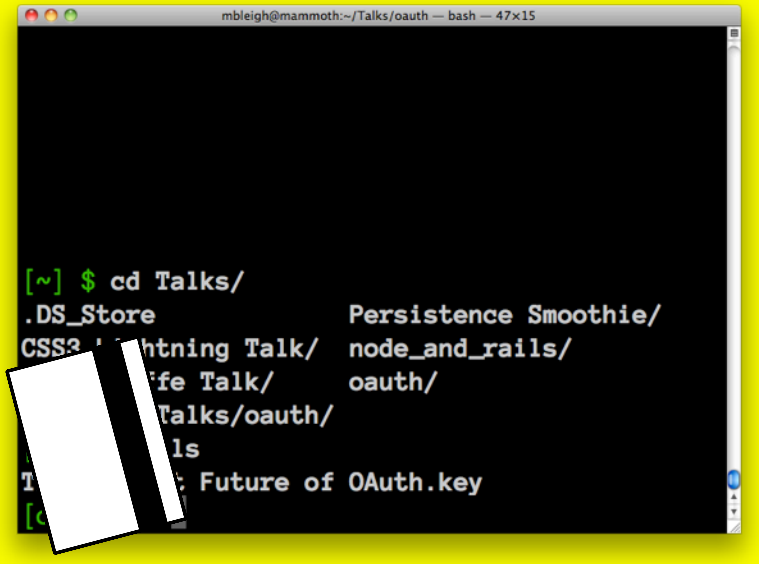




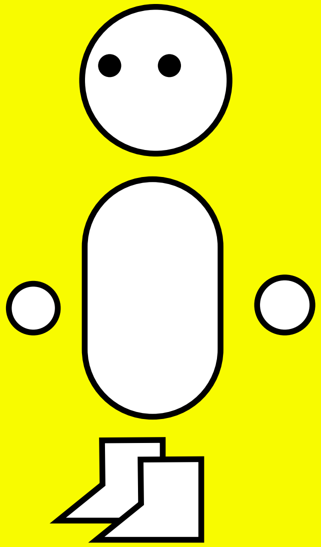
```
[~] $ cd Talks/
.DS_Store            Persistence Smoothie/
CSS3 Lightning Talk/  node_and_rails/
NoSQL Coffe Talk/     oauth/
[~] $ cd Talks/oauth/
[oauth] $ ls
The Present Future of OAuth.key
[oauth] $
```



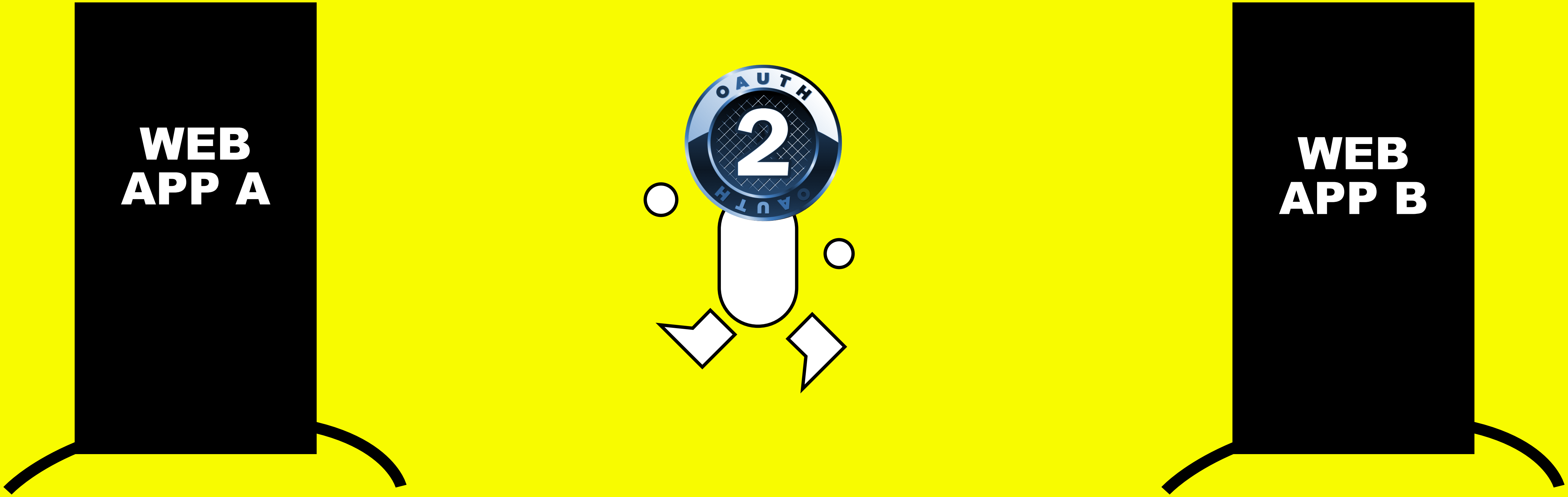
PASSWORD



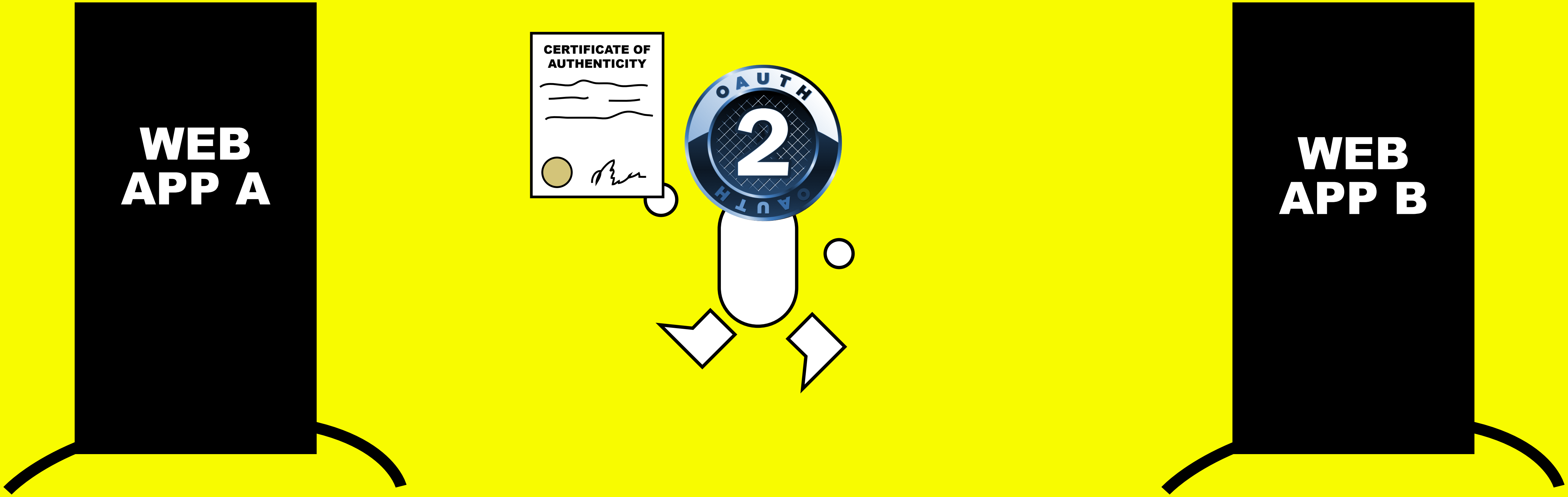
```
[~] $ cd Talks/
.DS_Store                 Persistence Smoothie/
CSS3      htning Talk/    node_and_rails/
           fe Talk/       oauth/
          Talks/oauth/
            ls
T           : Future of OAuth.key
[c
```



CLIENT CREDENTIALS



ASSERTION

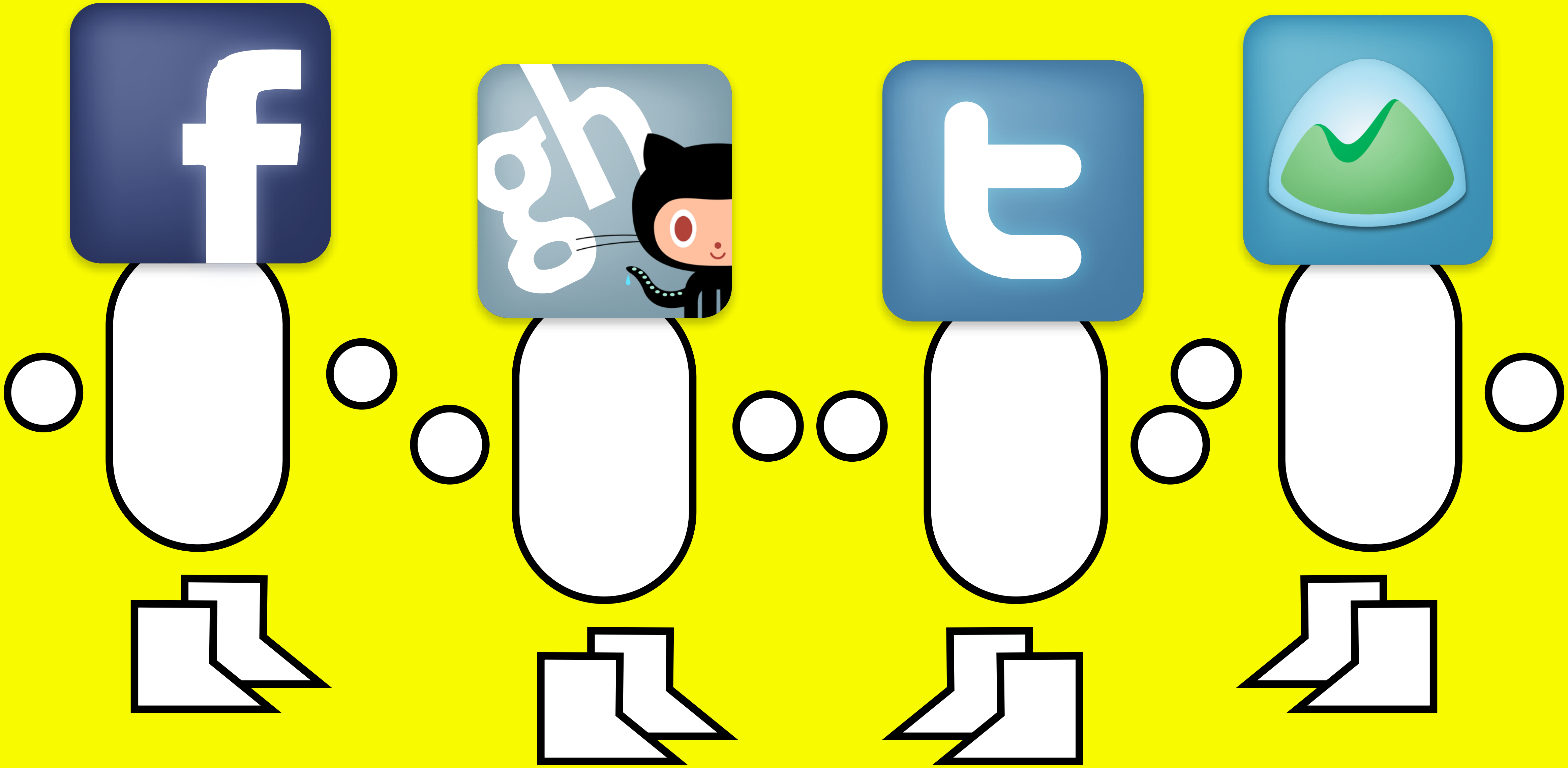

WEB APP A
CERTIFICATE OF AUTHENTICITY
OAUTH
2
OAUTH
WEB APP B

FLEX-
IBILITY

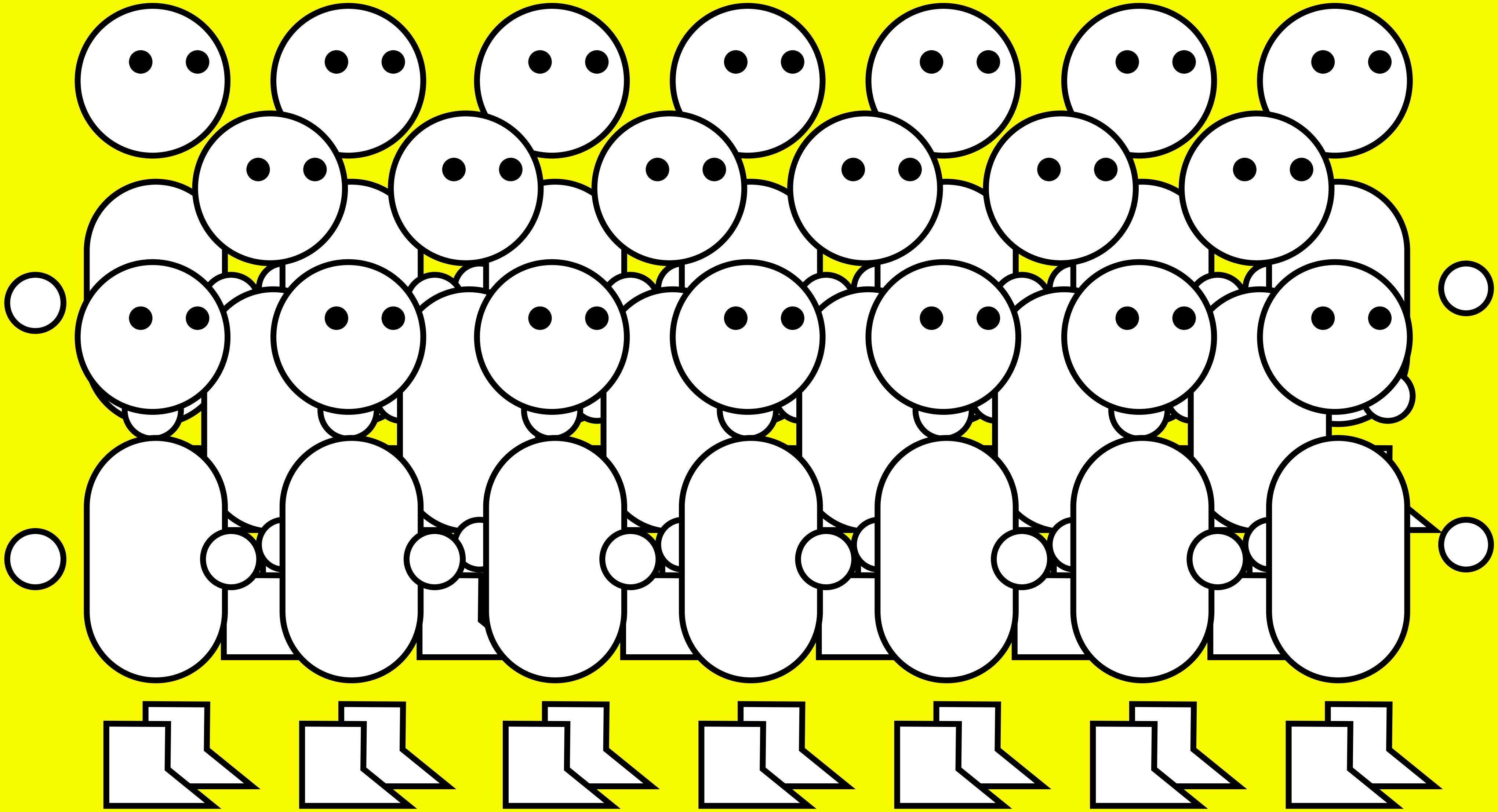
ACT 4

IN WHICH WE GET DOWN TO BUSINESS

WHO'S DOING IT RIGHT NOW?



WHO WILL BE DOING IT SOON?



WHO WILL BE DOING IT SOON?

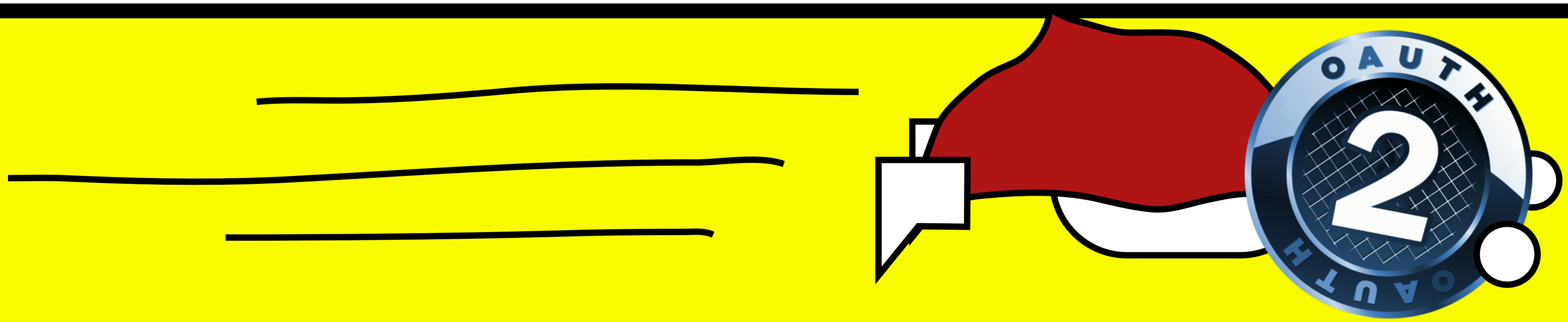
YOU

CONSUMING OAUTH 2.0

```
# in Gemfile
gem 'oauth2'

$ rails g controller oauth

# in routes.rb
resource :oauth, :controller => 'oauth' do
  get :start
  get :callback
end
```



```ruby
class OauthController < ApplicationController
  def start
    redirect_to client.web_server.authorize_url(
      :redirect_uri => callback_oauth_url(:format => 'json'),
      :scope => 'user'
    )
  end

  def callback
    access_token = client.web_server.get_access_token(
      params[:code], :redirect_uri => callback_oauth_url(:format => 'json')
    )

    # you should store the access token info now.
    render :json => access_token.get('/api/v2/json/user/show')
  end

  protected

  def client
    @client ||= OAuth2::Client.new(
      '296e901b0e6ab74db167', '625fe65c7f74ee4a015d121efb011a45776d510d',
      :site => 'https://github.com',
      :authorize_path => '/login/oauth/authorize',
      :access_token_path => '/login/oauth/access_token'
    )
  end
end
```

PROVIDING OAUTH 2.0

READ THE SPEC

http://bit.ly/oauth2-spec

NO SERIOUSLY,

READ THE

SPEC

http://bit.ly/oauth2-spec