Action Cable 101

real-time web experiences on Rails

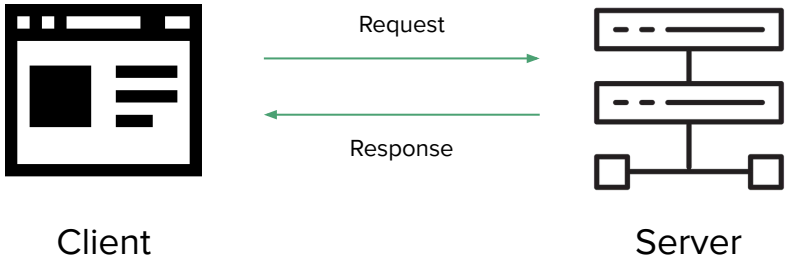
Topics

- Why use WebSockets?
- What is Action Cable?
- Demo implementation
- Gotchas, good-to-knows etc

whoami

- Solution architect, views presented here are my own :)
- Full-stack dev & general web technology enthusiast
- Serial prototypist
- Slack: @David Smith on Ruby Australia

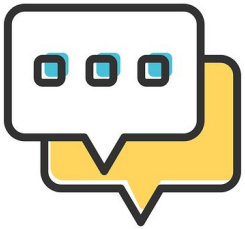
HTTP Request - Response cycle



- Initiated by client
- Stateless
- Point-in-time
- Heavy (relatively)

Well suited to “browsing”

Beyond document browsing



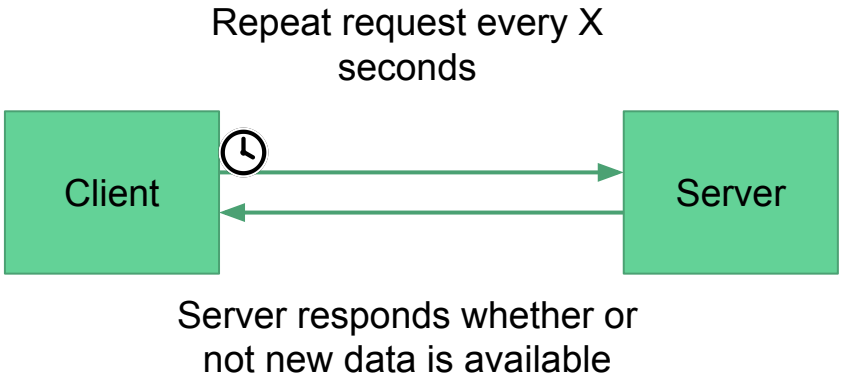
Dashboards, live events, stock prices

Chat & messaging

Alerts, notifications, recommendations

Collaboration - show user presence, changes to documents

“Real time” over HTTP: polling



“Real time” over HTTP: long-polling

Repeat request with long TTL

Server waits for new information to respond

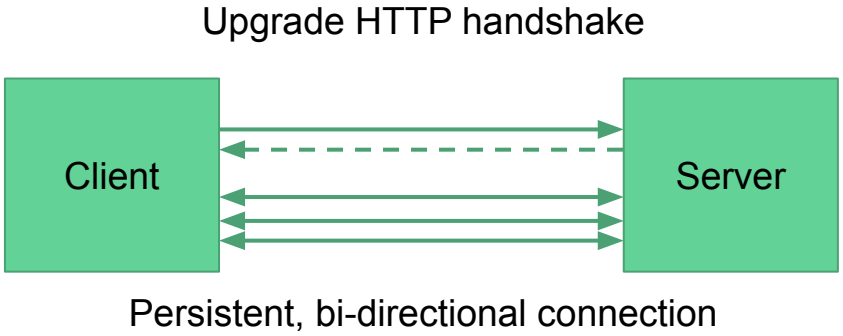
“Real time” over HTTP: Server Sent Events

Open a connection to the server

Server sends continuous events as available

https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events

“Real time” over HTTP: WebSockets



https://datatracker.ietf.org/doc/html/rfc6455

“Real time” over HTTP: WebSockets

```
// Create WebSocket connection.
const socket = new WebSocket('ws://localhost:8080');

// Connection opened
socket.addEventListener('open', function (event) {
    socket.send('Hello Server!');
});

// Listen for messages
socket.addEventListener('message', function (event) {
    console.log('Message from server ', event.data);
});
```

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+
Sec-WebSocket-Protocol: chat
```

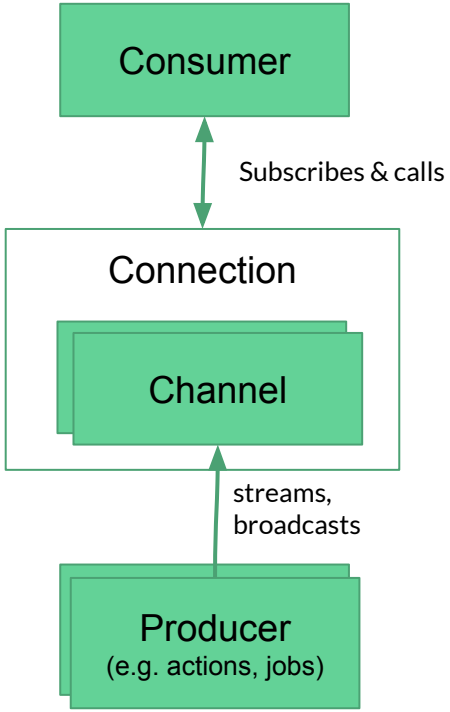
https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

What is Action Cable?

Not just websockets!

Pubsub abstractions in Rails:

- Connections
- Channels
- Subscriptions
- Streams and broadcasts



Action Cable: Connection

```
# app/channels/application_cable/connection.rb

module ApplicationCable
 class Connection < ActionCable::Connection::Base
   identified_by :current_user

   def connect
     self.current_user = find_verified_user
   end

   private

   def find_verified_user
     # FIXME: use real auth in production!!
     cookies.signed[:user_id] ||
       reject_unauthorized_connection
   end
 end
end
```

Action Cable: Channel

```
# app/channels/notifications_channel.rb

class NotificationsChannel < ApplicationCable::Channel
 def subscribed
   stream_from 'all_notifications'
   ActionCable.server.broadcast('all_notifications', {
     user: current_user,
     message: "has joined"
   })
 end

 def unsubscribed
   # Any cleanup needed when channel is unsubscribed
 end

 def typing
   ActionCable.server.broadcast('all_notifications', {
     user: current_user,
     message: "is typing"
   })
 end
end
```

Action Cable: subscription

app/javascript/notifications_channel.js

```
consumer.subscriptions.create("NotificationsChannel", {
 connected() {
   // Called when the subscription is ready for use on the server
   this.typing = this.typing.bind(this);
   document.addEventListener('keyup', this.typing)
 },

 disconnected() {
   // Called when the subscription has been terminated by the server
 },

 received(data) { // receive a message
   MyNotificationApp.notificationReceived(data);
 },

 typing() { // Invokes #typing on notifications_channel.rb
   this.perform('typing');
 }
});
```

Action Cable: `stream_for`

```ruby
class CommentsChannel < ApplicationCable::Channel
 def subscribed
   post = Post.find(params[:id])
   stream_for post
 end
end

# From a controller or background job
CommentsChannel.broadcast_to(@post, @comment)
```

https://guides.rubyonrails.org/action_cable_overview.html#example-2-receiving-new-web-notifications

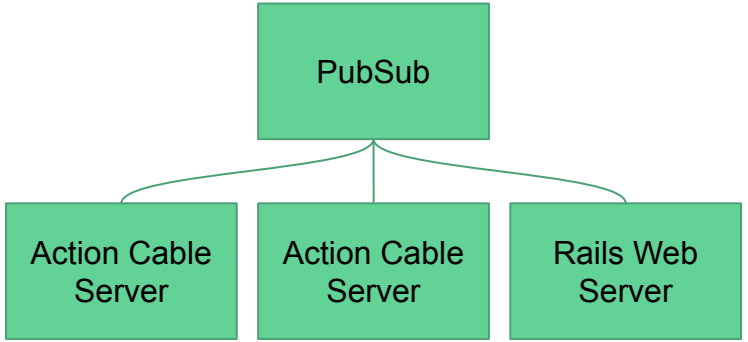
Action Cable: Adapter

```
# config/cable.yml

development:
  adapter: async

test:
  adapter: test

production:
  adapter: redis
  url: <%= ENV.fetch("REDIS_URL") { "redis://localhost:6379/1" } %>
  channel_prefix: rails_events_production
```



Action Cable: Tests

https://api.rubyonrails.org/classes/ActionCable/Channel/TestCase.html

https://relishapp.com/rspec/rspec-rails/docs/channel-specs/channel-spec

https://guides.rubyonrails.org/testing.html#testing-action-cable

Given a file named "spec/channels/chat_channel_spec.rb" with:

```ruby
require "rails_helper"

RSpec.describe ChatChannel, :type => :channel do
  it "successfully subscribes" do
    subscribe room_id: 42

    perform :echo, foo: 'bar'
    expect(transmissions.last).to eq('foo' => 'bar')
  end
end
```

When I run `rspec spec/channels/chat_channel_spec.rb`

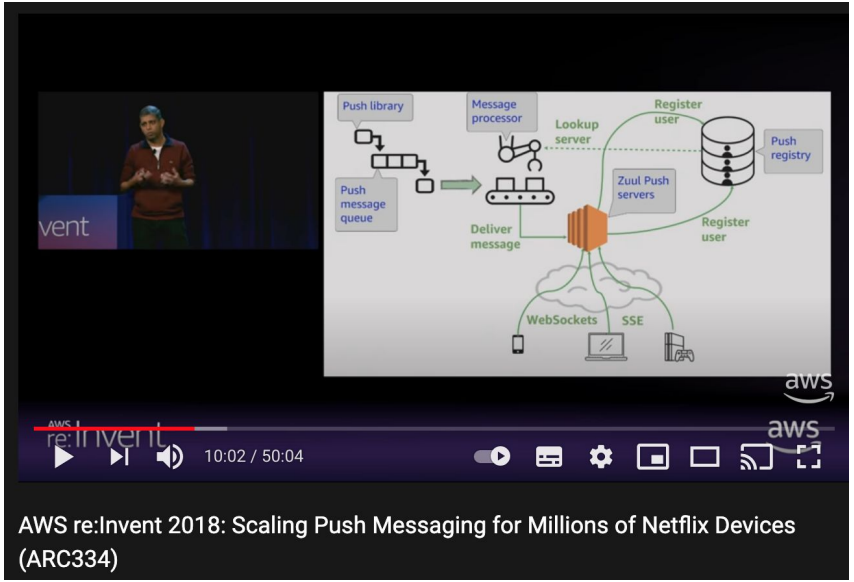
Then the example should pass

Managing Persistent Connections

`Connection` and `Channel` instance is long-lived

Think about deployments

Check reverse proxy and load balancer settings



https://www.youtube.com/watch?v=IdR6N9B-S1E

Action Cable: good to know

+ Uses your existing auth, domain logic
+ Automatic reconnection
+ Flexible server model
+ Redis pubsub adapter included

- Messages aren't durable, ordered or guaranteed
- Not versioned
- Client is JS-specific

References & further reading

https://api.rubyonrails.org/classes/ActionCable/Channel/Base.html

https://medium.com/@dhh/rails-5-action-cable-demo-8bba4ccfc55e

https://www.learnenough.com/action-cable-tutorial

Action Cable = integrated WebSockets & pubsub for real-time Rails

Questions, comments, complaints: Slack @David Smith on Ruby Australia

Fork github.com/djs070/action-cable-demo

Thank you!