Лекция 13
AJAX
Websockets

Нужно сделать чат

- Можно отправлять сообщения
- Можно получать сообщения

Пишем миграцию

```ruby
def up
  create_table :messages do |t|
    t.text :body
    t.references :user
    t.timestamps
  end
end

def down
  drop_table :messages
end
```

Пишем модель

```ruby
class Message < ApplicationRecord

  validates :body, presence: true
  belongs_to :user

end
```

Пишем контроллер для закрытых страниц

```
class  BaseAuthController  < ApplicationController
  before_action :get_user

  private

  def get_user
    @current_user = User.find_by_id(session[:user])
    redirect_to auth_path unless @current_user.present?
  end
end
```

Пишем контроллер для чата

```
class ChatController < BaseAuthController

  def index
    #do nothing, just render
  end

  def message
    #create new message and return to chat
    redirect_to :index
  end

end
```

Дописываем контроллер

```ruby
def index
  @messages = Message.all
end

def message
  message = Message.create(body: params.require(:body))
  message.user = @current_user
  message.save
  return render status: :bad_request unless message.valid?

  redirect_to chat_path
end
```

Добавляем роутинг

```
get 'chat', to: 'chat#index'
post 'chat', to: 'chat#message'
```

Bootstrap в шаблоне

```
<body>
<div class="row">
  <div class="col-12">
    <%= yield %>
  </div>
</div>
</body>
```

Test, which is a new approach

12:00 PM | Aug 13

Apollo University, Delhi, India Test

12:00 PM | Aug 13

Apollo University, Delhi, India Test

12:00 PM | Aug 13

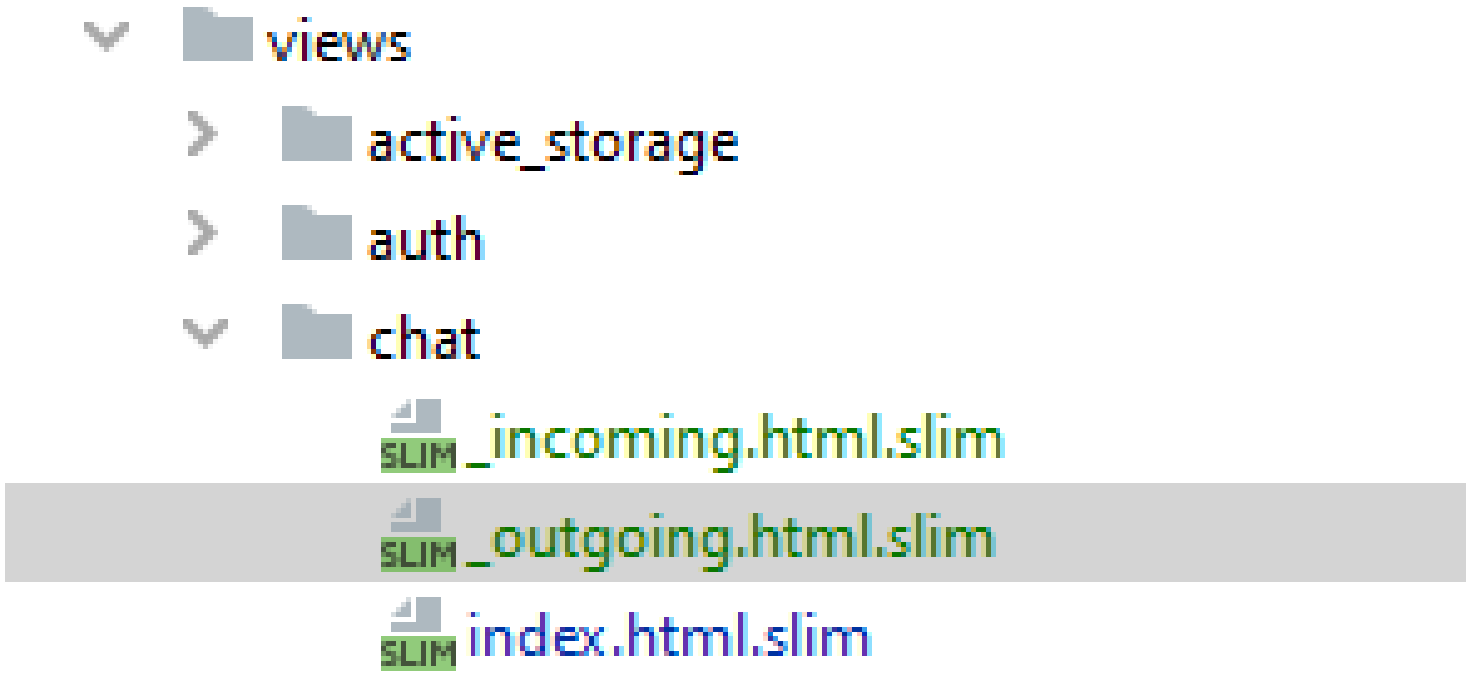
Apollo University, Delhi, India Test

12:00 PM | Aug 13

Введите сообщение

Есть 2 типа сообщений

- Нужно отображать список из объектов разного вида, но общей структуры
- Будем рендерить отдельно в зависимости от вида



Partial View

```
.media.w-50.mb-3.ml-auto
  .media-body
    .bg-primary.rounded.py-2.px-3.mb-2
      p.text-small.mb-0.text-white =message.body
    p.small.text-muted =l message.created_at
```

```
def direction(user)
  self.user == user ?  'outgoing' : 'incoming'
end
```

Верстаем чат

```
.row.justify-content-center
  .col-8.px-0
    .px-4.py-5.chat-box.bg-white
      -@messages.each do |m|
        = render partial: m.direction(@current_user),
                          object: m
```

ActionController::InvalidAuthenticityToken in ChatController#message

ActionController::InvalidAuthenticityToken

Extracted source (around line **#217**):

```
215
216           def handle_unverified_request
217             raise ActionController::InvalidAuthenticityToken
218           end
219         end
220       end
```

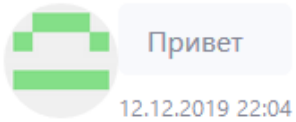
Токены в формах

```
<%= form_for @person do |f| %>
  <%= f.label :first_name %>:
  <%= f.text_field :first_name %><br />

  <%= f.label :last_name %>:
  <%= f.text_field :last_name %><br />

  <%= f.submit %>
<% end %>
```

```
<input name="authenticity_token" type="hidden" value="" />
```

Создаем сообщение

```
form.bg-light action='/chat' method='post'
  .input-group
    input name="authenticity_token" value= form_authenticity_token type="hidden"
    input.form-control.rounded-0.border-0.py-4.bg-light name="body" type="text"
    .input-group-append
      button#button-addon2.btn.btn-link type="submit"
        i.fa.fa-paper-plane
```



Много недостатков!

- Нет возможности получить новое сообщение
- Приходится обновлять страницу
- Рендерится много лишнего

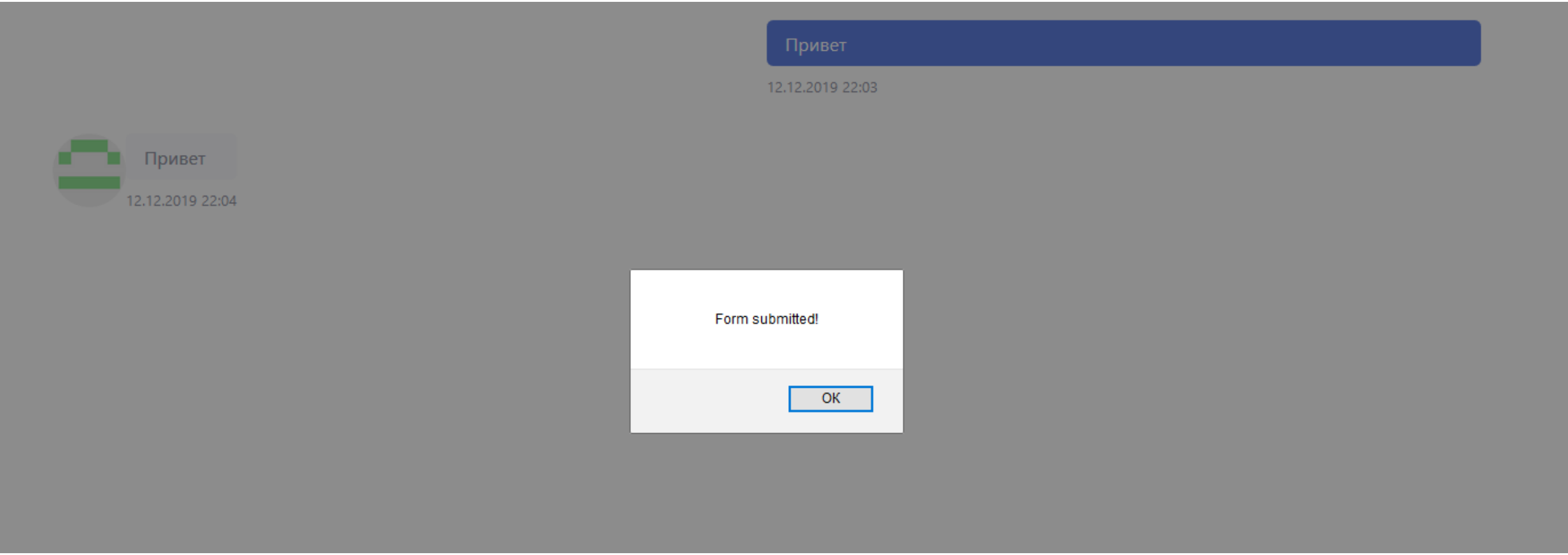
Будем рендерить на клиенте!

- Нужно перестать отправлять форму
- Нужно поймать событие отправки и сделать что-то свое
- Нужно разобраться с токеном
- Нужно добавить полученные данные на страницу

JS

```
$(document).ready(function() {
    $('#message-form').submit( function (e) {
        alert('Form submitted!');
        e.preventDefault(e);
        return false;
    });
});
```

Форма больше не работает!



Отладка JS


```
18          success: function(data) {
19              appendIncomingMessage($(".chat-box"), data)}
20      });
21  }
22  $(document).ready(function() {
23      $('#message-form').submit( function (e) {
24          alert(e);
25          e.preventDefault(e);
26          return false;
27      });
28  });
29
```

Достаем данные из формы

```
$('#message-form').submit( function (e) {
    const data = $('#message-form').serializeArray().reduce(function(obj, item) {
        obj[item.name] = item.value;
        return obj;
    }, {});
    console.log(data);
    e.preventDefault(e);
    return false;
});
```

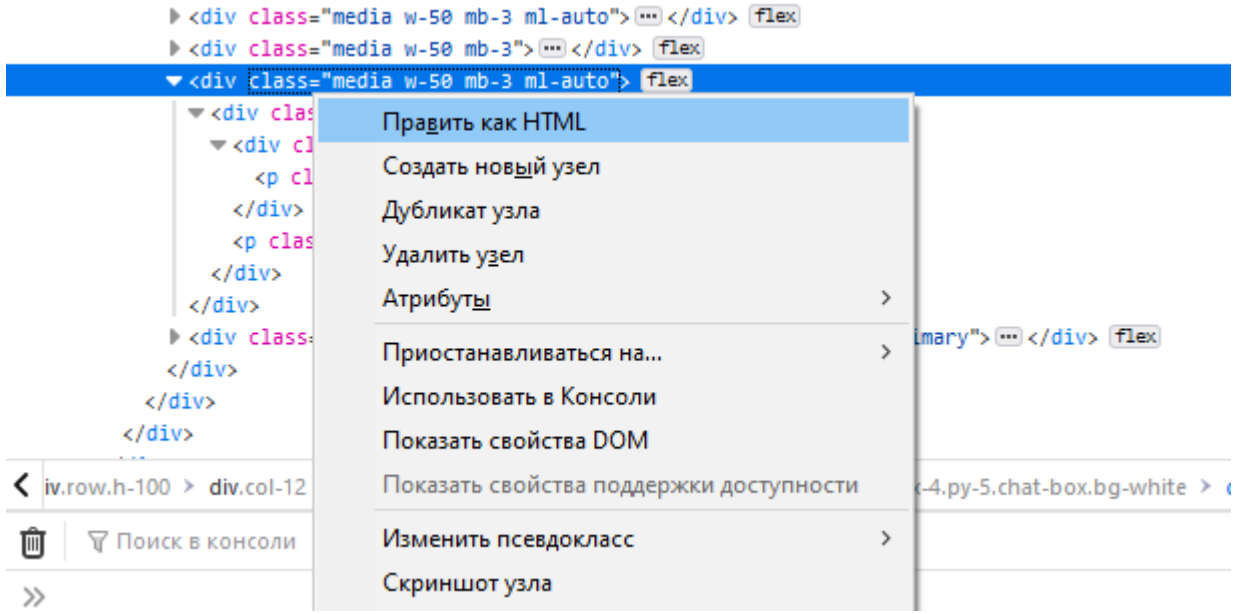
Отправляем запрос

```
function getAuthToken() {
    return $('meta[name=csrf-token]').attr('content');
}
function sendMessage(data) {
    $.ajax({
        type: "POST",
        url: "/chat",
        data:{'body': data.body,
                'authenticity_token': getAuthToken() },
        dataType: "script",
        success: function(data) {
            appendIncomingMessage($(".chat-box"), data)}
    });
}
```

Отвечаем по формату

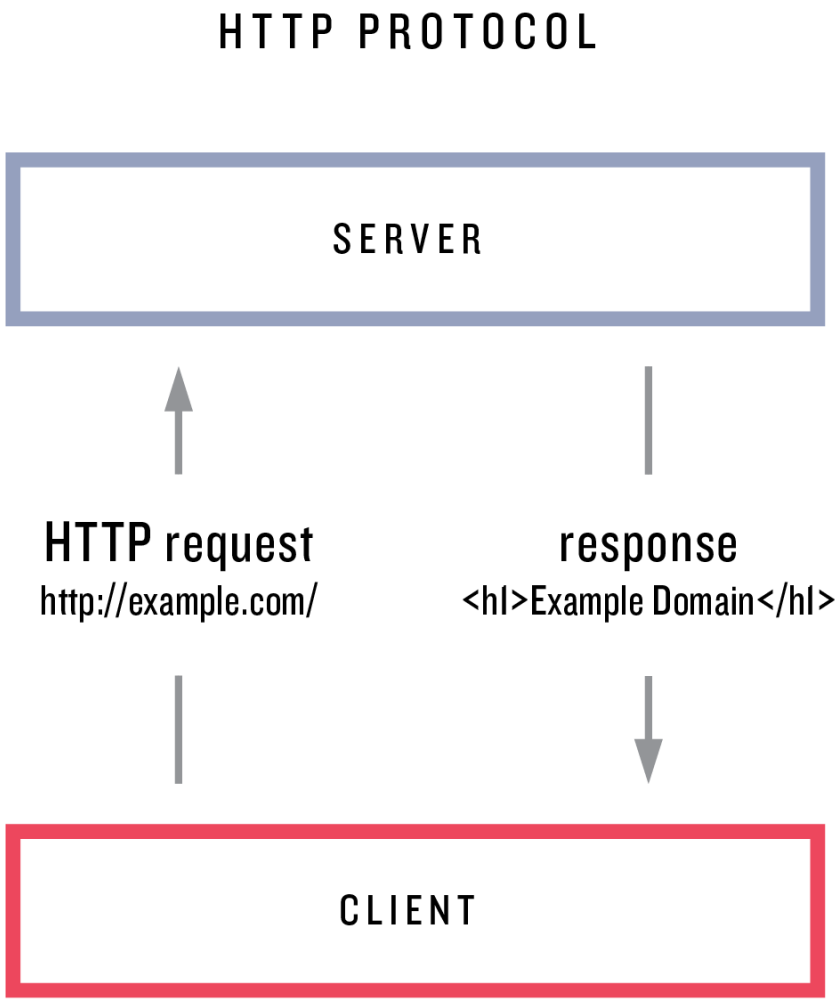
```
def message
  message = Message.create(body: params.require(:body))
  message.user = @current_user
  message.save

  return render status: :bad_request unless message.valid?

  respond_to do |format|
    format.html redirect_to chat_path
    format.json { render json: {
        body: message.body,
        avatar_url: message.user.avatar_url,
        created_at: message.created_at
    } }
  end
end
```

Приклеиваем данные



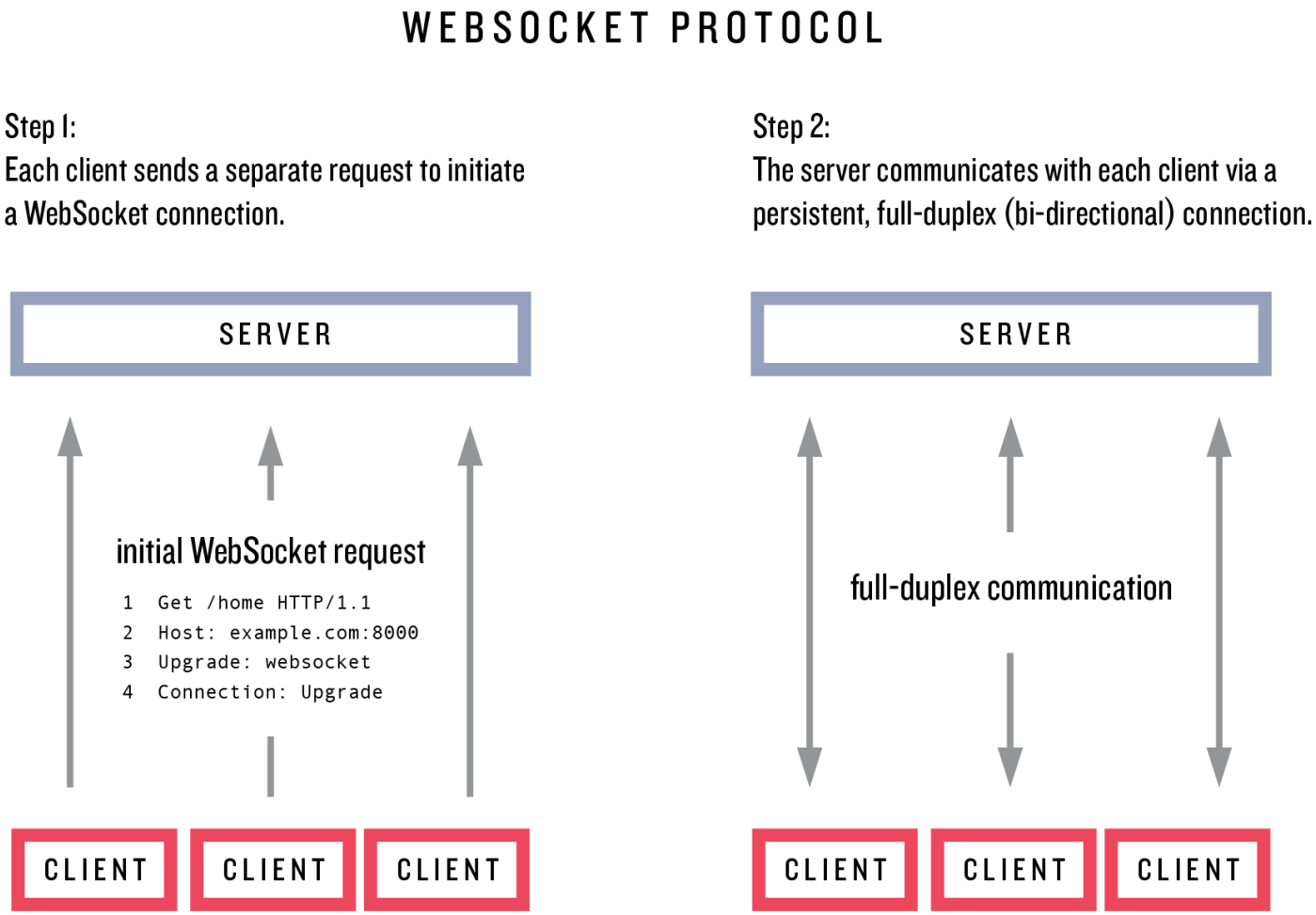

```
function appendIncomingMessage(element, message) {
    element.append("<div class='col-4'></div>")
}
```

Что было?


HTTP PROTOCOL
SERVER
HTTP request
http://example.com/
response
<h1>Example Domain</h1>
CLIENT


PRESS-TO-TALK
SWITCH NOT
DEPRESSED WHILE
RECEIVING

Что нужно?


WEBSOCKET PROTOCOL
Step I:
Each client sends a separate request to initiate a WebSocket connection.
SERVER
initial WebSocket request
1 Get /home HTTP/1.1
2 Host: example.com:8000
3 Upgrade: websocket
4 Connection: Upgrade
CLIENT
CLIENT
CLIENT
Step 2:
The server communicates with each client via a persistent, full-duplex (bi-directional) connection.
SERVER
full-duplex communication
CLIENT
CLIENT
CLIENT

Polling

- Давайте опрашивать сервер с каким-то периодом!

Polling

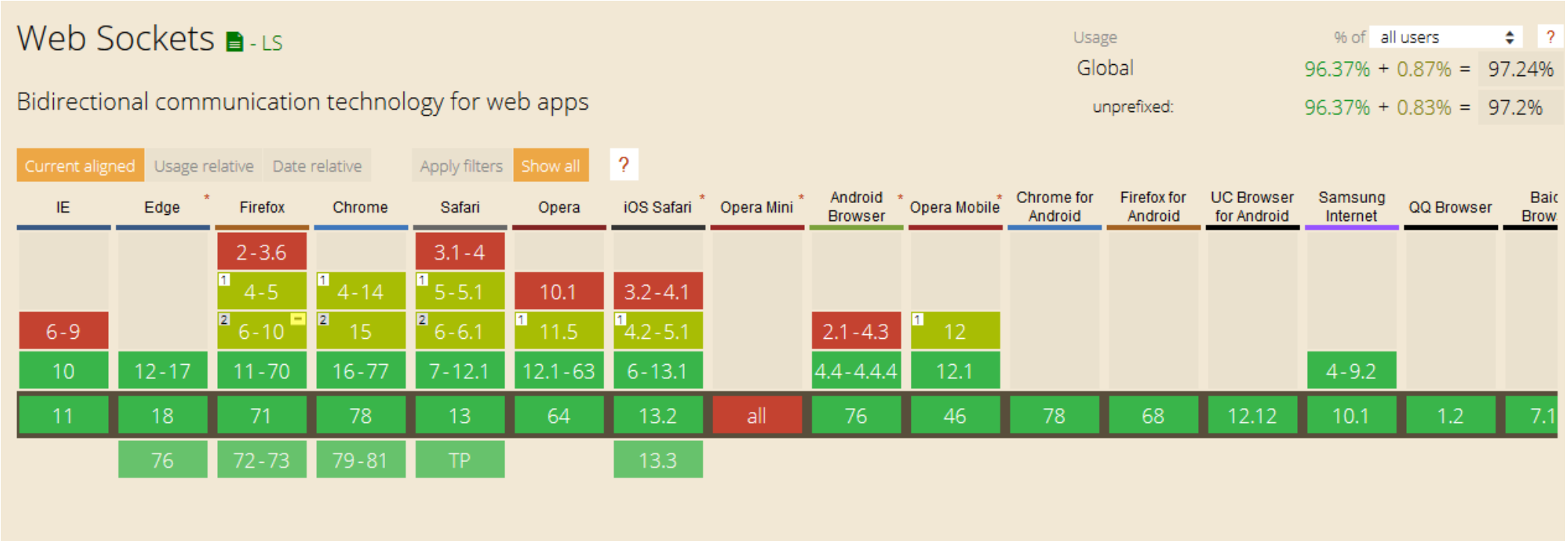
- ~~Давайте опрашивать сервер с каким-то периодом!~~

Нагрузка в моменты, когда ничего не происходит

Ужасные перспективы масштабирования

WebSockets

- **WebSocket** — протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

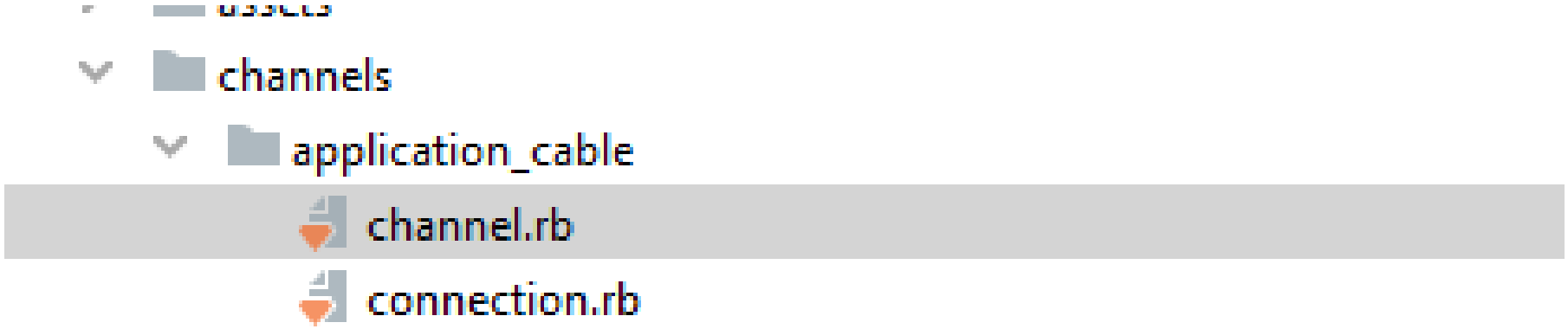


Publish-subscribe

- Рассылка сообщений в канал без указания конкретных получателей
- Сообщения фильтруются либо по теме либо по содержимому

Action Cable

```
module ApplicationCable
  class Channel < ActionCable::Channel::Base
  end
end
```



Делаем канал для чата

```
$ rails g channel chat
      invoke  test_unit
      create    test/channels/chat_channel_test.rb
      create  app/channels/chat_channel.rb
   identical  app/javascript/channels/index.js
   identical  app/javascript/channels/consumer.js
      create  app/javascript/channels/chat_channel.js
```

События на сервере

```
class ChatChannel < ApplicationCable::Channel
  def subscribed
    # stream_from "some_channel"
  end

  def unsubscribed
    # Any cleanup needed when channel is unsubscribed
  end
end
```

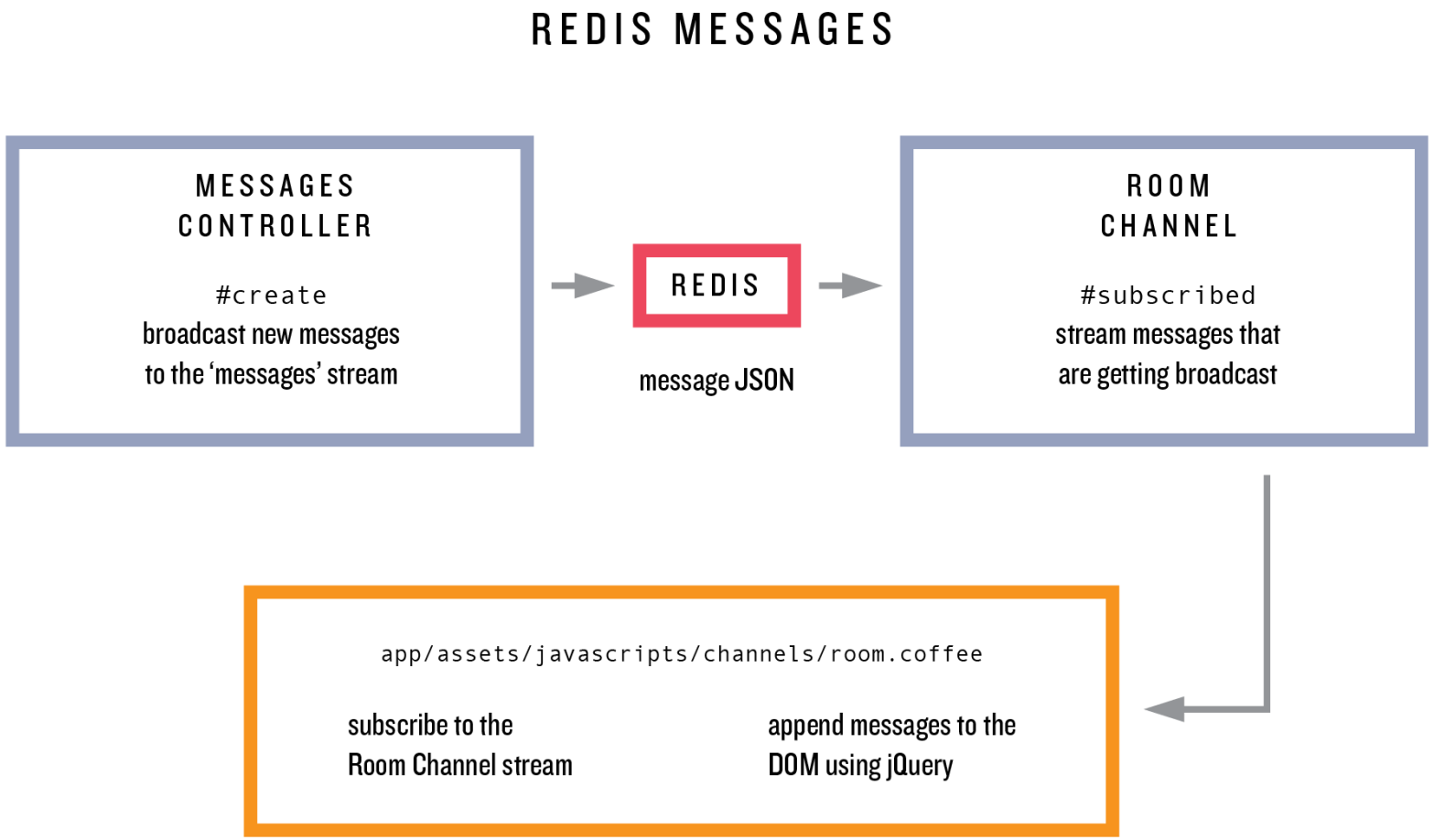
События на клиенте

```
import consumer from "./consumer"

consumer.subscriptions.create("ChatChannel", {
  connected() {
  },

  disconnected() {
  },

  received(data) {
  }
});
```

Отправляем сообщения

```
def message
  message = Message.create(body: params.require(:body))
  message.user = @current_user
  message.save

  return render status: :bad_request, :ok, json: {}
                                        unless message.valid?
  ActionCable.server.broadcast 'chat_channel',
                               content: message.body,
                               user_id: message.user.id,
                               avatar_url:…
  render status: :ok, json: {}
end
```

Redis


REDIS MESSAGES
MESSAGES CONTROLLER
#create
broadcast new messages to the 'messages' stream
REDIS
message JSON
ROOM CHANNEL
#subscribed
stream messages that are getting broadcast
app/assets/javascripts/channels/room.coffee
subscribe to the Room Channel stream
append messages to the DOM using jQuery

Конфигурация

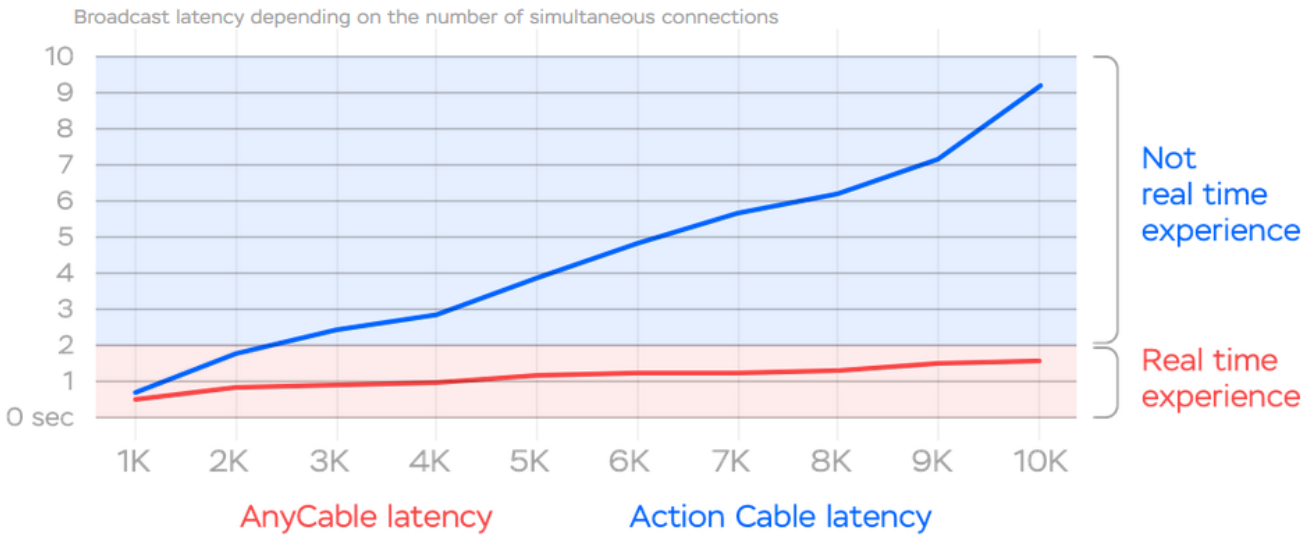
```
development:
  adapter: async

test:
  adapter: test

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

Клиентский рендеринг

```
function appendMessage(element, data) {
  element.append("<div class=\"media w-50 mb-3\"><img alt=\"avatar\" class=\"pic rounded-circle \" src=“+
      data.avatar_url+
      "><div class=\"media-bodyml-3\"><div class=\"bg-light rounded py-2 px-3 mb-2\">"+
      "<p class=\"text-small mb-0 text-muted\">"+data.body+"</p>" +
      '</div><p class=\"small text-muted\">'+data.created_at+"</p></div></div>\n");
    window.scrollTo(0,document.body.scrollHeight);
}
```

AnyCable


Handling 20K connections
AnyCable
798 MB
Action Cable
3,5 GB


Broadcast latency depending on the number of simultaneous connections
10
9
8
7
6
5
4
3
2
1
0 sec
1K
2K
3K
4K
5K
6K
7K
8K
9K
10K
Not real time experience
Real time experience
AnyCable latency
Action Cable latency

Ссылки

- https://iridakos.com/programming/2019/04/04/creating-chat-application-rails-websockets
- http://rusrails.ru/action-cable-overview
- https://anycable.io/