Использование ES6 и технологии AJAX в Rails

Знания

- Файлопровод
- Изменения в rails 5.1
- DOM и события
- Некоторые нововведения ES6
- Технология AJAX в jQuery
- Технология AJAX в Rails

Ресурсы (assets)

- **JS**
- CSS
- Изображения
- Шрифты
- …

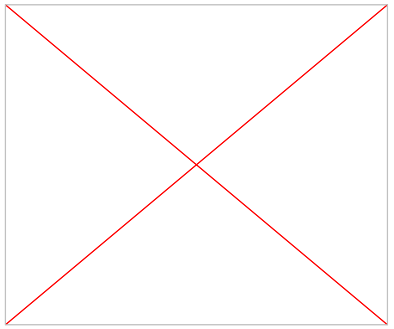
Файлопровод (assets pipeline)

- Объединение ресурсов
- Минимизация ресурсов
- Добавление препроцессоров и синтаксического сахара

Файлопровод (assets pipeline)

- Gem sprockets
- Gem sass-rails
- Gem coffee-rails
- Gem uglifier —
  - сжатие JS-файлов
  - Нужно окружение ExecJS

Манифест



- Подключение в layout-файле
  - javascript_include_tag 'custom'
  - stylesheet_link_tag 'application'
- Подключение JS/CSS файлов
  - //= require js-файл
  - //= require_tree

Умения

- Создать text.txt.erb
- Получить доступ к text.txt из браузера

Использовать asset_path, **image-url** в sass/css.erb

```
background-image: image-url('select_arrow.png')
```

- Подключить свой манифест

Знакомьтесь, JS!

- JavaScript — язык разработки пользовательского интерфейса веб-приложений

Популярность JS

- Серверная часть
  - Express.js — JavaScript
- Клиентская часть
  - Backbone.js
  - Angular.js
  - React.js
- СУБД — MongoDB
- Мобильные приложения — React Native

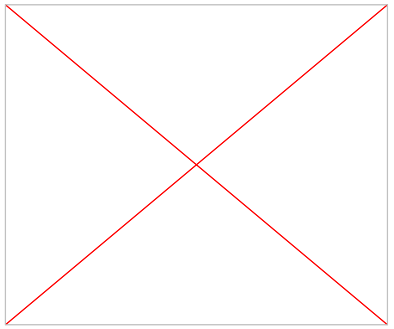
Изменения в Rails 5.1

- Управление JS-зависимостями с Yarn
- Система сборки проекта на JS Webpack
- Поддержка Babel.js и ES6

Yarn

- Bundler для JS
- Альтернатива NPM
- $ yarn add <имя пакета>
- Используются файлы package.json и yarn.lock

Webpacker



- Сборщик приложения для JS-проекта
- Подключение необходимых модулей
- Дополнение/замена Sprockets
- …

Умения

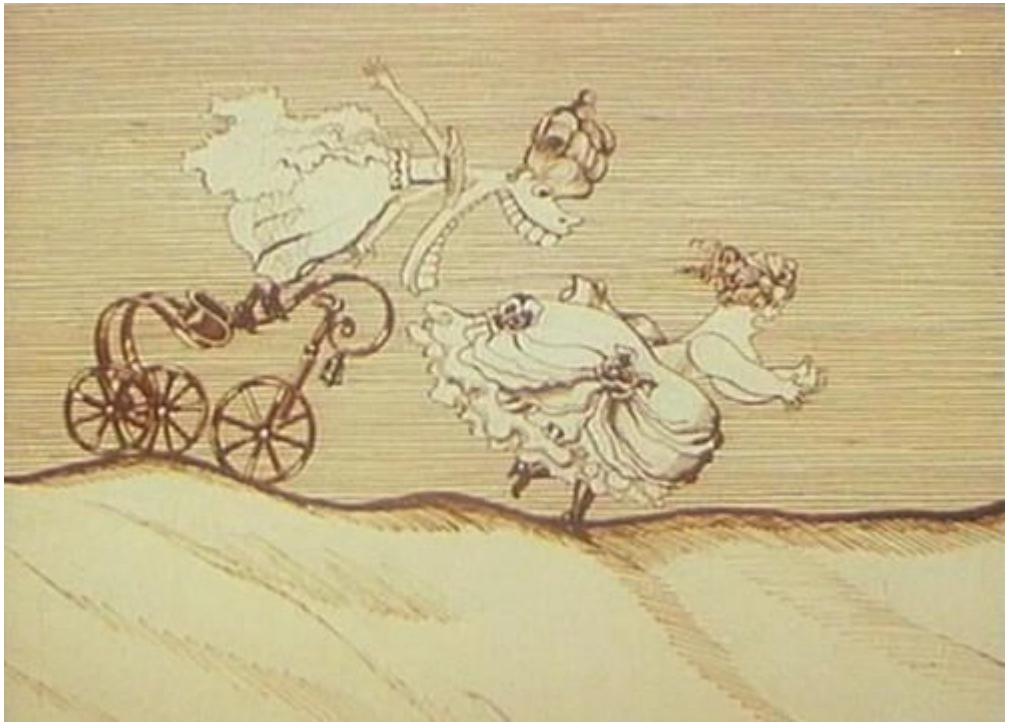
- Проверить Babel,
  - Let, ``
  - использовать fetch

JS-минимум

- Синтаксис
- Объектная модель
- DOM и события

DOM и события

- А это — весёлая императрица,
- Которая часто кусает певицу

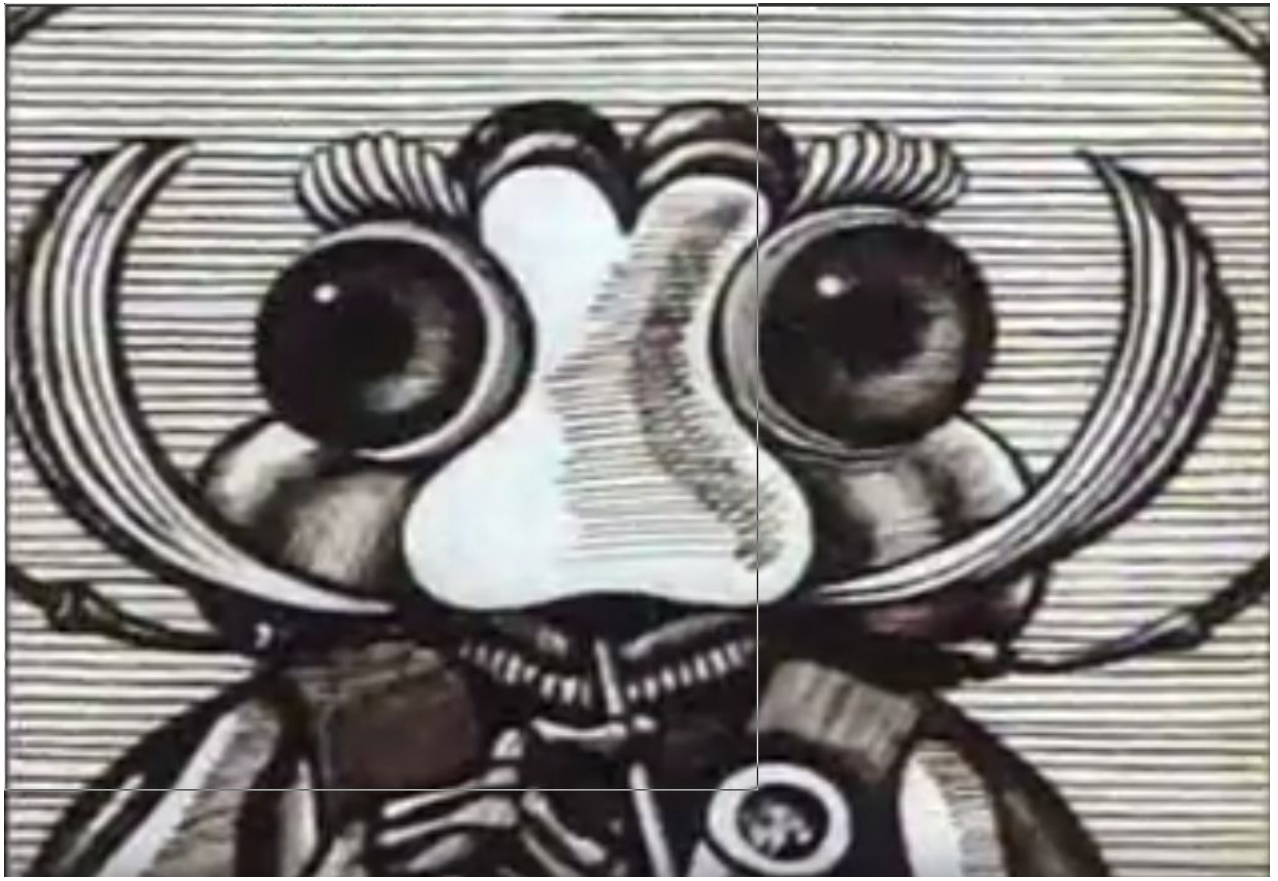


DOM и события

- Которая в тёмном чулане хранится

DOM и события

- В доме, который построил Жук



Создание-чтение-изменение-удаление в JS

- Создание: createElement, затем insertBefore (elem, beforeElem) или appendChild
- Чтение: см. слайд выше (getElement[s]By{Id | TagName | … }
- Изменение: elem.style.innerHTML=»Новое содержимое»
- Удаление: <parent>.removeChild(elem)

События в JS

- element.addEventListener(«event_name», handler);

Модель DOM в jQuery

- Создать:
- $(«selector»).append(« Test »)
- Считать: $(«selector»).html()
- Изменить: $(«selector»).html(«Другой текст»)
- Удалить: $(«selector»).remove()

Умения

- Вывести слайдер
- Подсчитывать количество введённых символов
- …

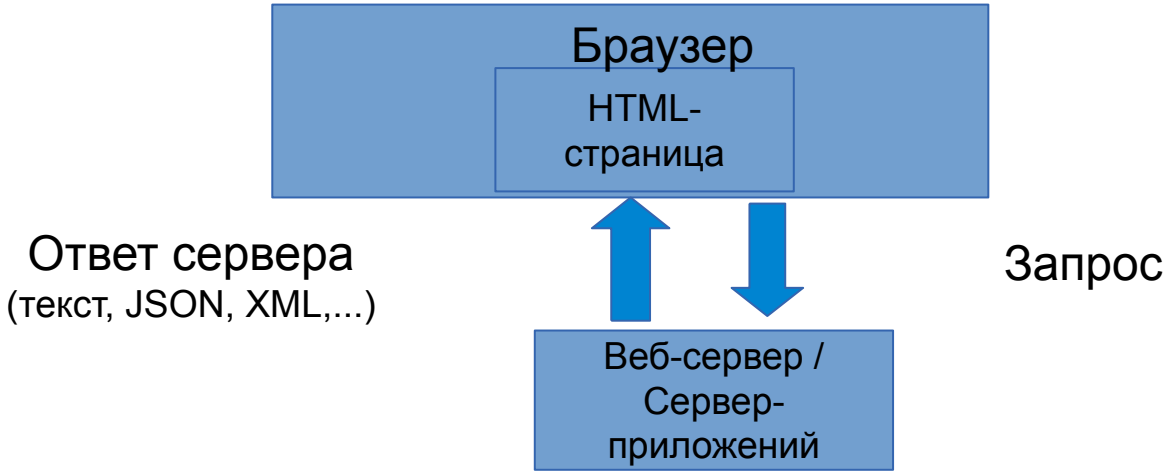
Вывести слайдер


```
window.addEventListener('load', main);
const PREFIX = "/assets/slider/";
let slides = ['slide1.jpg', 'slide2.jpg', 'slide3.jpg', 'slide4.jpg'];
let currentSlide = 0;
function main() {
  document.querySelector('#slide-right').addEventListener('click',
moveRight);
}
function moveRight(event) {
  event.preventDefault();
  currentSlide = ++currentSlide % slides.length;
  document.querySelector('#slide').setAttribute('src',
`${PREFIX}${slides[currentSlide]}`);
}
```

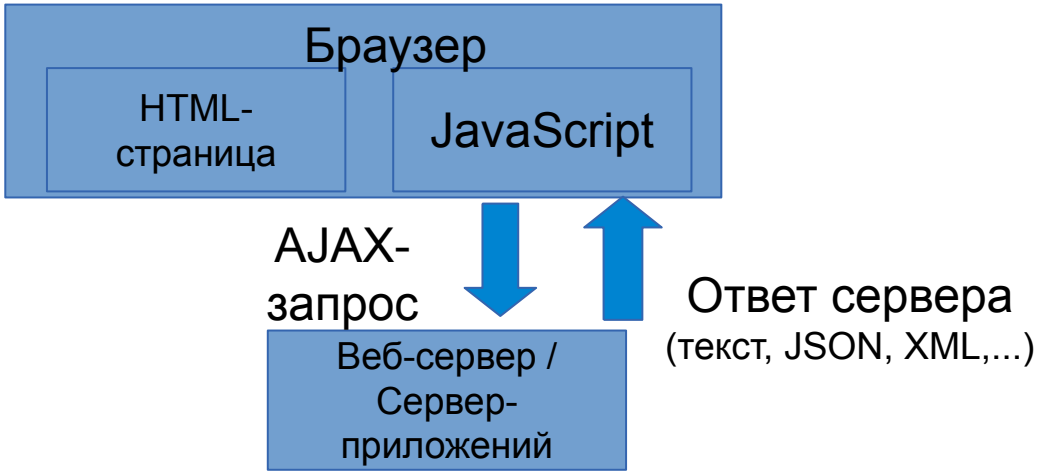
Подсчитывать кол-во введённых символов

```
$(function() {
    const LIMIT = 10;
    $('#expedition_description').on('keyup', function(e) {
        if (!$(this).siblings('.invalid-feedback').length)
            $(this).after('<div class="invalid-feedback"></div>');
        if ($(this).val().trim().length > LIMIT) {
            $(this).siblings('.invalid-feedback').html(`Превышено кол-во символов на
${$(this).val().trim().length - LIMIT }`).show();
        } else {
            $(this).siblings('.invalid-feedback').hide();
        }
    });
});
```

Общение с сервером: форма



Общение с сервером: AJAX



Реализация AJAX в JS

- Создаём объект XMLHttpRequest
- Открываем соединение open()
- Обрабатываем событие получения запроса — событие onreadystatechange
- Посылаем данные по открытому соединению send()

Реализация AJAX в JS: GET-запрос

```
1.let xhr = new XMLHttpRequest();
2.xhr.open('GET', "/competences.html", true);

3.xhr.addEventListener('readystatechange',
function(event) {

    if(xhr.readyState === XMLHttpRequest.DONE &&
xhr.status === 200)
        console.log(xhr.responseText);
});
4.xhr.send(null);
```

Реализация AJAX в jQuery

- Объект jqXHR
- Метод $.ajax
  - Url
  - Data — хеш данных
  - Type: метод HTTP-запроса
  - DataType — формат ожидаемого ответа
- Обработка ответа
  - Success: function(data) — всё хорошо
  - Error(jqXHR jqXHR, String textStatus, String errorThrown)

Версии JS

- ES9 (планируется в 2018)
- ES8 (2017 год)
  - Async/await functions, ...
- ES7 (2016)
  - Оператор возведения в степень 2**3
- ES6 (2015)
  - Промисы
  - Классы
  - ...
- ES5 (2009)
- ...

Нововведения ES6

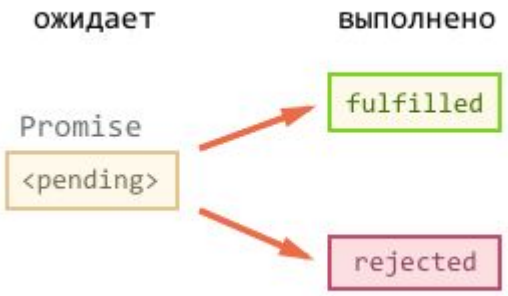
- стрелочные функции
- деструктуризация
- promise
- fetch

Стрелочные функции

- Реализация части функционального подхода
  - Больше декларативности, чем императивности
  - Что надо сделать, а не как
- Другой синтаксис для функциональных выражений и анонимных функций
- …

Промис

- Объект, который хранит своё состояние
- Позволяет реализовывать асинхронный код в «плоском» виде (в виде цепочек вызовов)
- Больше декларативности
- new Promise((resolve, **reject**) => {...})
  - .then((data) => {…})
  - **.catch((err) => {…})**



Промис — бросание монетки, случай

«Провидение не алгебра.
Ум человеческий, по простонародному выражению, не пророк,
а угадчик, он видит общий ход вещей
и может выводить из оного глубокие предположения,
часто оправданные временем,
но невозможно ему предвидеть случая —
мощного мгновенного орудия Провидения».

Promise

```
let coin = new Promise((resolve, reject) => {
    let res = Math.round((Math.random()*100));
    if (res > 50)
        reject(`Сдаём сессию`);
    else
        resolve(`Летим в космос`);
});
coin
    .then((result) => console.log(result))
    .catch((err) => console.error(err))
;
```

Функция fetch

- XHR-вызовы
- Построен на Promise
- Возвращает Response

Выводим вакансии

```
const URL = "https://api.hh.ru/vacancies?text=rails";
let ul = document.createElement('ul');
document.body.appendChild(ul);
fetch(URL)
    .then((response) => {
        return response.json();
    })
    .then((result) => result.items.forEach((item) => {
        let li = document.createElement('li');
        li.innerHTML = `<a href="${item.alternate_url}">${item.name}</a>`;
        ul.appendChild(li);
    }))
    .catch((err) => console.error(`ошибка запроса ${err}`));
;
```

Готовим сани летом: передаем привет React

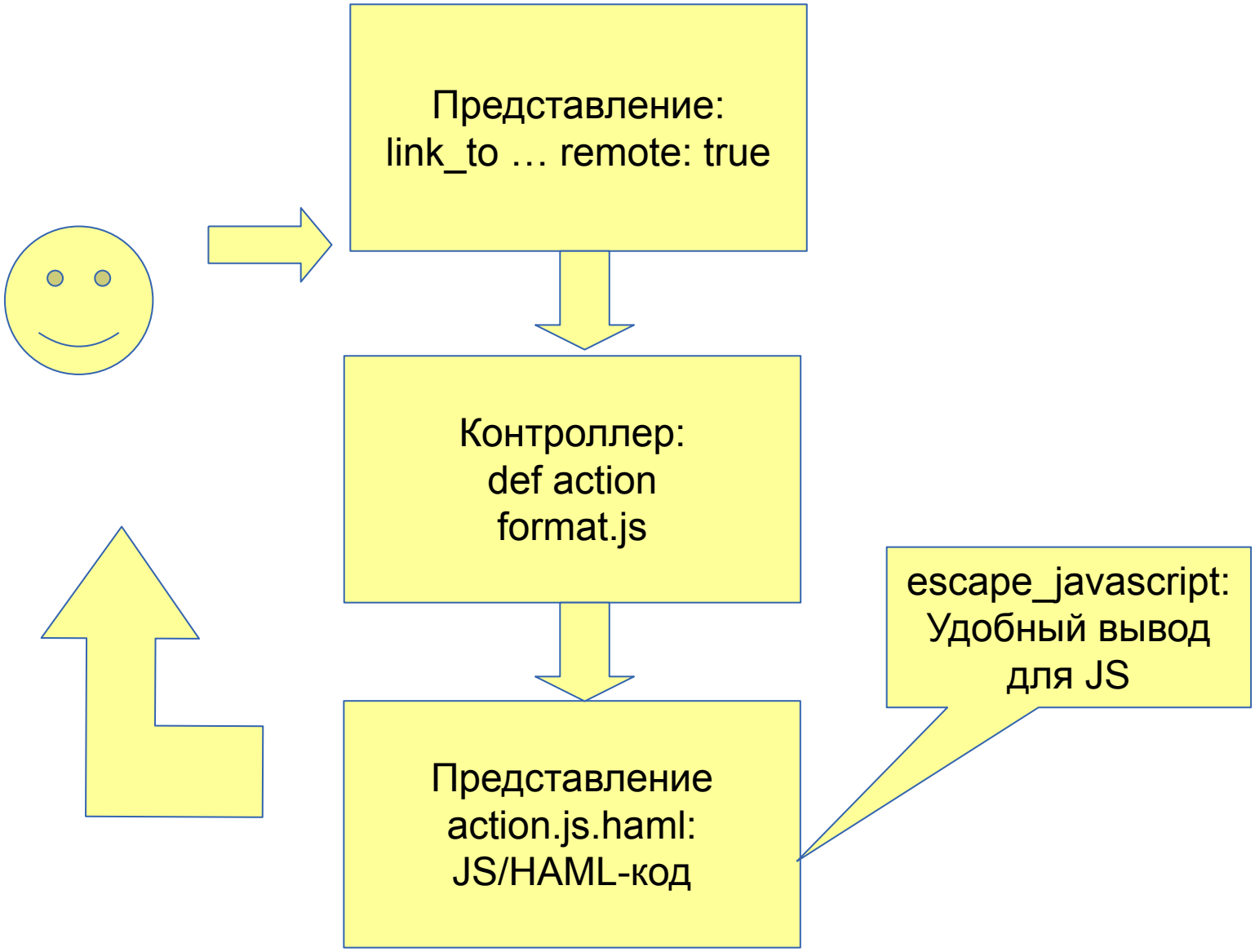
- установить:
  - gem webpacker
  - yarn
- Запустить:
  - rails webpacker:install
  - rails webpacker:install:react
  - spring binstub
- Добавить в application.html.haml:

```
= javascript_pack_tag 'application'
= javascript_pack_tag 'hello_react'
```

Удалить ресурс с помощью AJAX

```
$('[data-method="delete"]').click(function(event) {
   event.preventDefault();
   event.stopPropagation();
let expeditionId =
$(event.target).parents('tr').attr('data-id');
   $.ajax({
      url: `/expeditions/${expeditionId}`,
     data:
{ authenticity_token:
$('[name="csrf-token"]')[0].content },
      type: 'delete',
      dataType: 'text',
      success: function(data) {
$(event.target).parents('tr').fadeOut().remove();
      }
   })
});
```

AJAX в Rails


Представление:
link_to … remote: true
Контроллер:
def action
format.js
Представление
action.js.haml:
JS/HAML-код
escape_javascript:
Удобный вывод
для JS

Удалить ресурс средствами Rails

```
# index.html.haml

%tr{id: dom_id(expedition)}
    %td= link_to 'Удалить', expedition_path(expedition),
method: :delete, data: { confirm: 'Вы уверены?' },
remote: true

# контроллер

format.js

# destroy.js.haml

:plain
  $("##{dom_id(@expedition)}").remove();
```

Создать новую экспедицию

Вывести форму по AJAX-вызову

Реализовать создание по AJAX-вызову

Форма создания (new.js.haml)

```
:plain
  var partial = "#{escape_javascript(render partial: 'form')}";
  $('table.table').before(partial);
```

create.js.haml

```
:plain
  $('form.simple_form').remove();
                $('#expeditions').prepend('#{escape_javascript(render(partial:
'expedition', object: @expedition))}');
```

Заглянем под капот

Как формируется AJAX-запрос

```
$('[data-method="delete"]').click(function(event) {

// ...
   $.ajax({
      url: `/competences/${competenceId}`,
     data:
{ authenticity_token:
$('[name="csrf-token"]')[0].content },
      type: 'delete',
      dataType: 'script', # получаем и выполняем JS
   })
});

# контроллер

format.js

# destroy.js.erb

$(`tr[data-id="<%=@competence.id%>"]`).fade
Out().remove()
```

Как создаётся объект XHR (rails-ujs)

```
createXHR = function(options, done) {
  var xhr;
  xhr = new XMLHttpRequest();
  xhr.open(options.type, options.url, true);
  xhr.setRequestHeader('Accept', options.accept);
  if (typeof options.data === 'string') {
                              xhr.setRequestHeader('Content-Type',
'application/x-www-form-urlencoded; charset=UTF-8');
  }
//…
  xhr.withCredentials = !!options.withCredentials;
  xhr.onreadystatechange = function() {
    if (xhr.readyState === XMLHttpRequest.DONE) {
      return done(xhr);
    }
  };
  return xhr;
};
```


ТУДА
100м
СЮДА
200м
ВЕЧНАЯ НЕОПРЕДЕЛЕННОСТЬ!
КАК ЖИТЬ?
DEMOTIVATORS.RU

Неопределённости

- Зачем вообще нужен AJAX?
- Почему при AJAX-запросе может возникнуть ActionController::InvalidAuthenticityToken ?
  - Нужно явно указать в запросе
  - data: { authenticity_token: $('[name="csrf-token"]')[0].content },
  - В form_with authenticity_token: true
- Почему может не сработать event.preventDefault() для обработчика события onclick на ссылку «Удалить» (компетенцию)? Нужно вызвать stopPropagation()

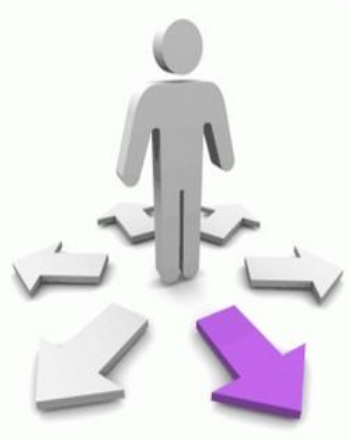
Результат


Это же так просто!
Я это сделаю!
Я могу это сделать!
Я попытаюсь...
Как мне это сделать?
Я хочу это сделать!
Я не могу это сделать (не умею, не получится)
Я не буду этого делать.

Результат

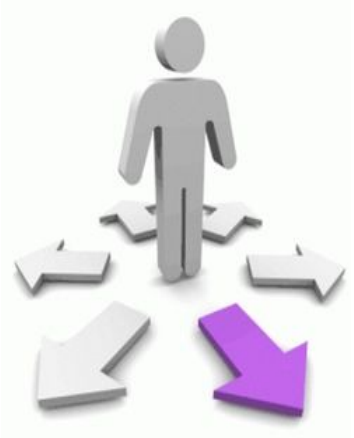
- Познакомились с файлопроводом
- Научились использовать ES6 в Rails
- Сделали наше веб-приложение более интерактивным с помощью AJAX

Весьма желательно



- Jquery DOM — чувствуйте себя в DOM как дома
- Jquery события
- Jquery AJAX
- Метки файлопровода
- …

Формирование необходимого кругозора веб-разработчика



- Использование ресурсов (assets) в development и production
- POST-запрос с помощью XHR
- ES6 ООП
- CoffeeScript
- Turbolinks
-
- …

- jQuery

```
$(function() {
  return console.log("DOM готов");
});
```

```
$(".button").on("click", function() {
  return console.log("нажми меня!");
});
```

```
$(".button").on("click",
function(event) {
  console.log("меня нажали!");
  return event.preventDefault();
});
```

- CoffeeScript

```
$ ->
  console.log("DOM is ready")
```

```
$(".button").on "click", ->
  console.log("нажми меня!")
```

```
$(".button").on "click", (event) ->
  console.log("меня нажали!")
  event.preventDefault()
```

Список источников

- Основное
- Assets
- AJAX в Rails
- Api.jquery.com
- …

- Дополнительное
- CoffeeScript и jQuery
- XHR
- Learn.javascript.ru