История про медленный SVG, node.js и немного Ruby

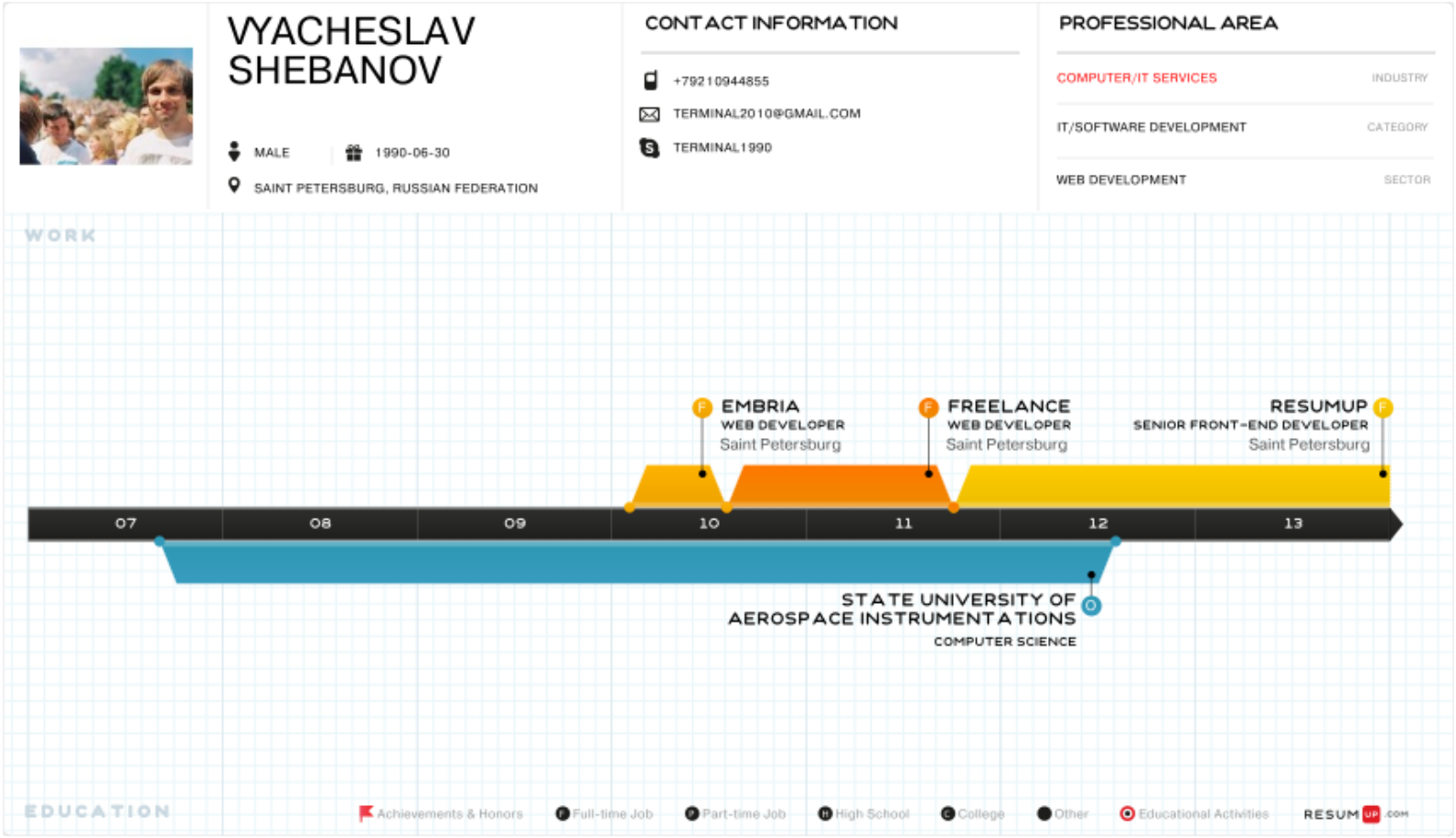
Вячеслав Шебанов,

RESUM UP

Немного о рассказчике

И еще немного

В чем проблема?



Чего мы хотим?

- Скорости
  - сложные резюме рисуются от 5 до 30 секунд
  - нас это не устраивает
- Использовать JavaScript
  - написано много кода, который не хочется выбрасывать
- Устойчивости

Когда мы этого хотим?

- Сейчас

Почему так долго?

- Слабые места Raphael: множественные операции с DOM
- Операции с DOM очень дорогие
- Реализация скорее всего обусловлена тем, что не все браузеры поддерживают inline-SVG

1. Cache

- Берем MongoDB
- Рисуем SVG на клиенте
- Через ajax отправляем на сервер в mongo
- На следующий заход: если браузер поддерживает inline-SVG берем кэш из mongo
- Иначе рисуем через Raphael

1. Cache

- Плюсы
  - из кэша рисуется где-то секунду
- Минусы
  - не все браузеры поддерживают inline-SVG
  - 1 секунда это долго

2. Server side

- получаем от начальства установку: рисовать надо еще и на сервере
- сделать надо быстро

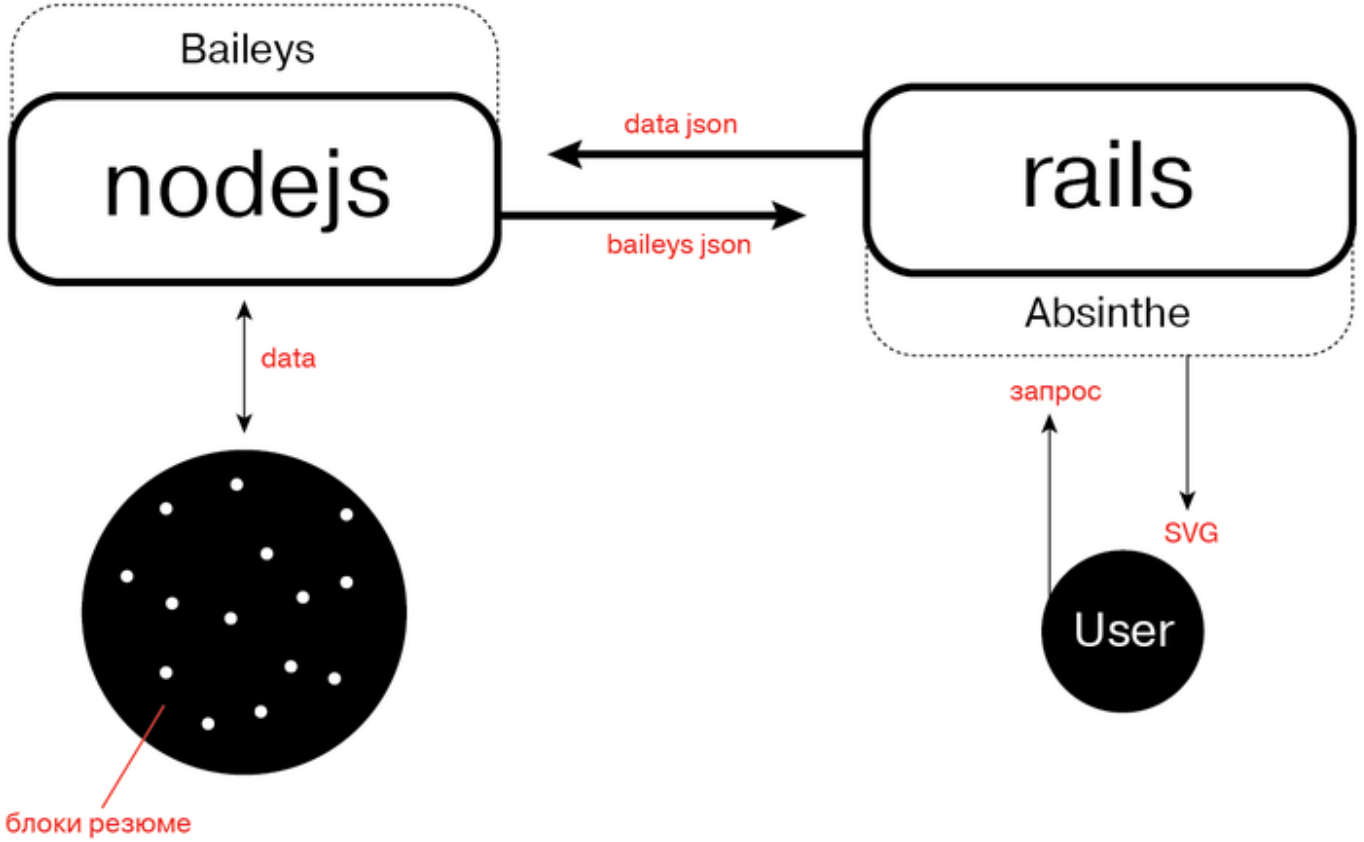
2. Server side

- Решаем в лоб, пробуем запустить все то же самое на сервере (execjs)
- Oops! DOM?
- Решение есть! (jsdom)
- На самом деле нет (jquery + raphael + jsdom = memory leaks)
- nodejs + express js! (опять нет)

3. По-настоящему node.js

- Выкидываем Raphael
- Пишем свою библиотеку, которая умеет строить абстрактную модель векторного изображения (Baileys)
- Пишем конвертер этой модели в SVG (Absinthe)

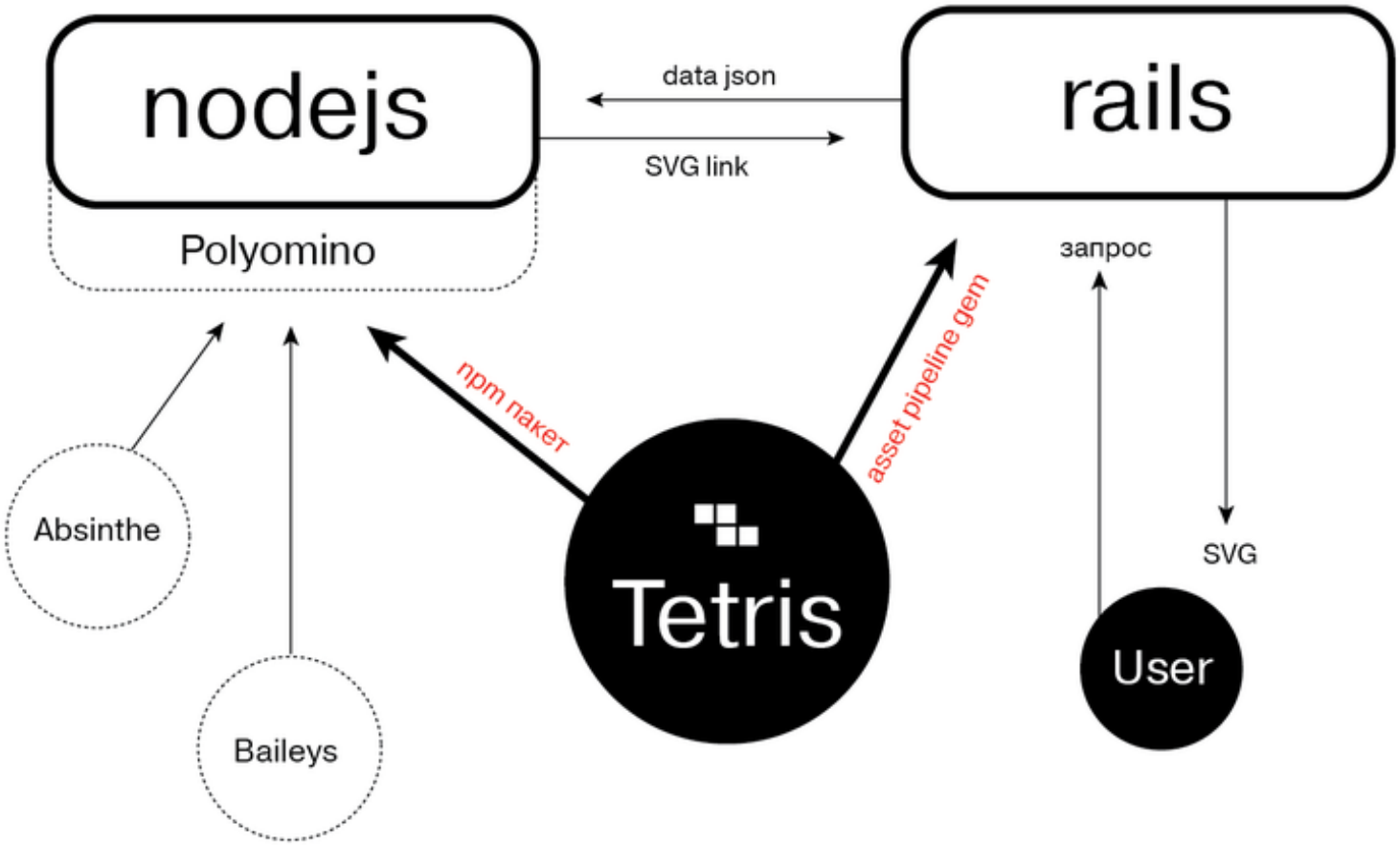
3. По-настоящему node.js



3. По-настоящему node.js

- Что получилось?
  - отрисовка занимает 500мс
  - можем рисовать SVG на сервере

4. Дальше больше



4. Дальше больше

- Получилось
  - отрисовка 100мс
  - весь код отрисовки блоков написан на js
  - если сервер падает, отрабатывает fallback, что гарантирует устойчивость
  - Profit!

5. И еще чуть-чуть

- Добавим Sidekiq
- Добавим Danthes
- В итоге разгружаем веб-сервер

Конец

Контакты

- Вячеслав Шебанов
- Twitter: @thought_sync
- Github: Termina1
- E-mail: vs.shebanov@gmail.com