

eleks

Сбалансированное окружение для вашей продуктивности

Алексей Зеленюк, *Application Architect*

eleks.com





О чем доклад

- Хорошее окружение
- Хорошее JS окружение
- Мой первый Энтерпрайз
- Менее продвинутые проекты
- Пишем с нуля
- Что делать со сложностью?
- Бройлерплейты против велосипедов
- Теплый рельсовый CLI

Хорошее окружение:

1. Простое в понимании
2. Быстро отзывается на изменение или ошибку
3. Минимизирует рутинные операции
4. Хорошо расширяется
5. Переносимо на платформы

Хорошее JS окружение

1. Package Management

2. Bundling

3. Transpiling

4. Minification

5. Sourcemaps

6. Mocked API

7. Dev Webserver

8. Component Libraries

9. Code Style

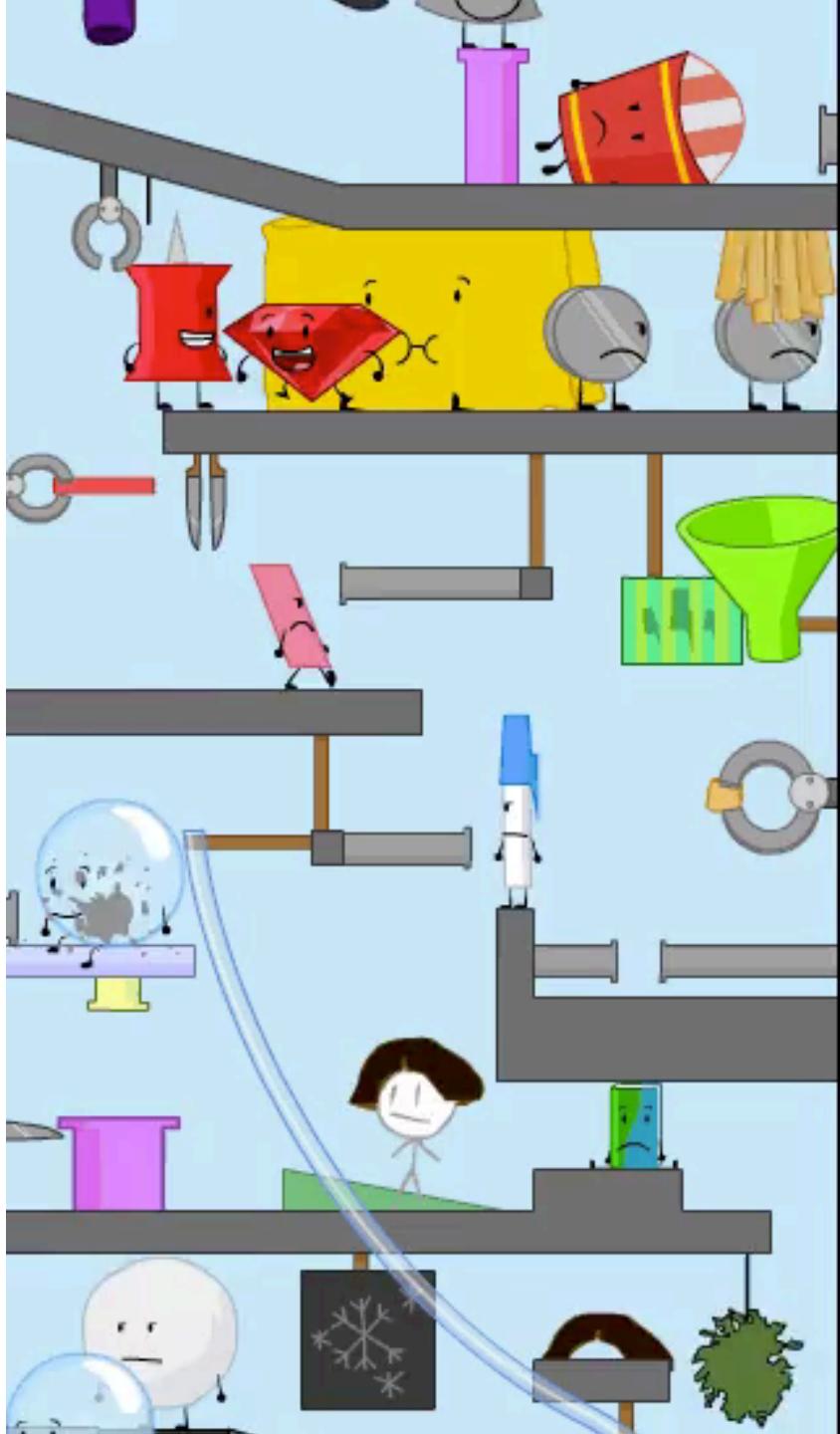
10. Юнит тесты

11. E2E тесты

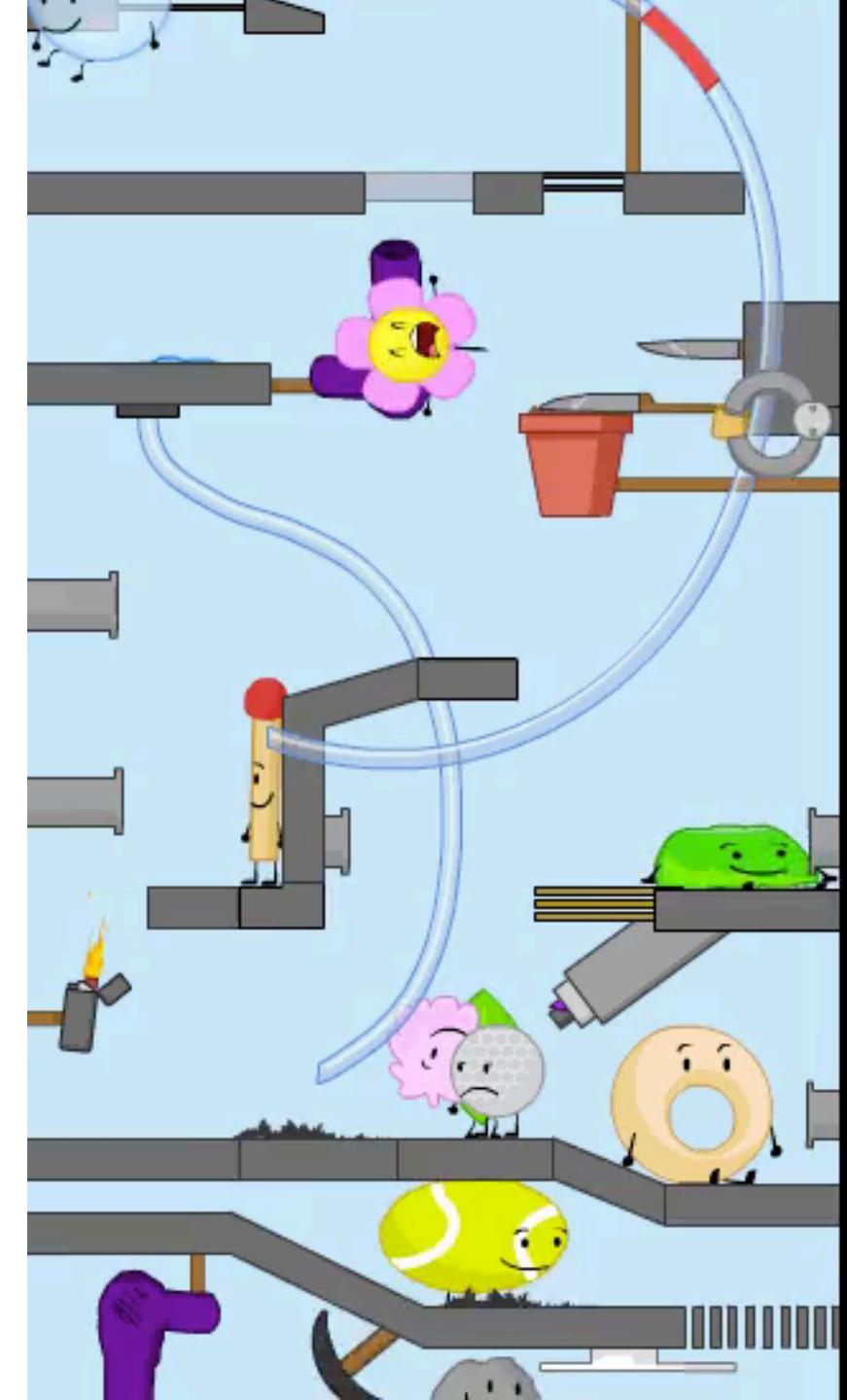
12. API тесты

В придачу

1. Build automation
2. Production deploy
3. Continuous Integration
4. Code-review процесс и инструменты
5. Tests coverage
6. SDLC специфика, и прочий Agile



Мой первый *JS* Энтерпрайз



Чтобы собрать

- Python (древний)
- Ruby gems
- Rake tasks (много)
- SH scripts (много)
- Backbone (0.9)
- RequireJS
- Юнит тесты (много)
- E2E тесты (много)
- NPM registry
- Компонент либы (свои)
- CI cluster
- На апдейт ~ 10 минут
- На деплой – ~ 1 час
- Жесткое код-ревью

Как-то так...



Когда попадаешь в
менее продвинутый
проект





Когда начинаешь проект с нуля



MAKE GIFS AT GIFOUP.COM

Что в итоге?



- Программа обросла библиотеками
- Билд обрастает скриптами
- Проект новыми требованиями
- И новыми костылями
- Много тяжелых тестов
- Сложность растет
- Рефакторить нет времени
- Лепим сверху, сзади и по бокам
- Производительность упала
- А давайте перепишем с нуля?

This time you have definitely chosen the right libraries and build tools

Начать с чистого листа нетрудно.
Трудно изменить почерк.

П. Коэльо



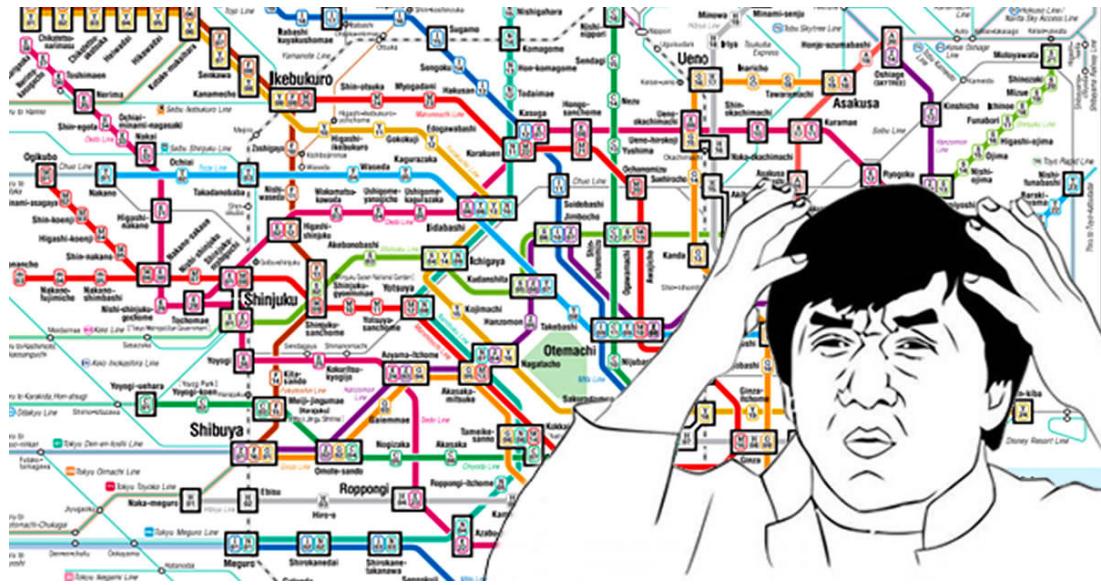
Real World

Rewriting Your Front
End Every Six Weeks

O RLY?

@ThePracticalDev

Избыточная сложность. Что к ней приводит?



1. Решение несуществующих проблем
2. Лишние зависимости для решения простых задач
3. Слишком умные модули
4. Непродуманные решения и лишние абстракции
5. Рефакторинг – преждевременный или запоздалый

Велосипеды

Преимущества и недостатки



- Выбираем только нужное
- Больше понимания
- Меньше лишней сложности
- Проще изменить
- Долгий старт
- Нужно поддерживать
- Свое не всегда лучшее

Бройлерплейты

Преимущества и недостатки

- Экономит время (вначале)
- Фокусируемся на задаче
- Поддерживается (иногда)
- Full-featured
- Стандартный стек
- Много лишнего
- Много магии
- Трудно расширять



4GIFs.com

Теплый рельсовый CLI

Преимущества и недостатки

- Стандартный подход
- Меньше рутины
- Ускоряет вход в проект
- Ускоряет разработку
- Само-документируемость



- Много лишнего кода
- Жесткие рамки
- Постоянная доработка шаблонов

Гибридный подход



1. Изучить лучшие решения
2. Выбрать наиболее подходящее
3. Пересобрать с нуля
4. Создать шаблоны для CLI

С чего начать?

1. Осмыслить требования
2. Изучить внешние модули и Зрatty
3. Быстрый прототип на бойлерплейте и фидбек
4. Продумать тестовое окружение
5. Настроить спецификацию (Swagger, etc.)
6. Настроить код стандарты: EditorConfig и линтеры
7. Переосмыслить и переписать

Что стоит отложить?

YAGNI

You Ain't Gonna Need It!

1. Твердый выбор технологий
2. Глубокую декомпозицию на модули
3. Внедрение абстракций
4. DRY принцип
5. Принятия решений в мелких деталях

Fake it until you make it!

1. Хардкодим все что неясно
2. Mocks & Stubs
3. Оставляем место для маневра
4. Монолитный код, готовый к декомпозиции
5. Не боимся ломать и выкидывать код

Масштабирование

1. SOLID принципы
2. Компонентная архитектура
3. Контейнеры
4. Композиция вместо наследования
5. Постоянный рефакторинг

О Команде

