ПРОГРАММИРОВАНИЕ*, JAVASCRIPT*

Я веб-разработчик и уже 10 дней не могу написать простейшее приложение

ИЗ ПЕСОЧНИЦЫ

halfcupgreentea 16 февраля 2016 в 11:38 130k

Предлагаю вашему вниманию перевод статьи "I'm a web developer and I've been stuck with the simplest app for the last 10 days".

От переводчика: мнение автора местами частично, или полностью, не совпадает с моим, но вопрос поднимается, как мне кажется, правильный. Рекомендую почитать комментарии в блоге автора.

В основном я занимаюсь full-stack web-разработкой. Периодически пишу бэкенд на Python или Ruby, иногда работаю с C#. Еще я пишу консольные утилиты на C++ и Node.js. Мне нравится Closure, я познакомился с web много лет назад, когда писал на Perl и PHP, а первые годы профессиональной разработки посвятил программированию на Java.

Когда я впервые встретился с Javascript, он в основном использовался для того, чтобы добавить на страничку "Текущее время". Это были девяностые, когда все хотели приправить свои странички так, чтобы пользователи могли оценить, как это клево: текущий день недели выводится *динамически*. А спустя какое-то

время оказывалось, что Javascript может гораздо больше, и мы получаем полностью динамический HTML — DHTML!

Последнее время я разрабатывал довольно большие SPA с использованием разных фреймворков, а, когда торопился, то и с кучей грязного кода из которого то тут, то там торчала jQuery-лапша.

Десять дней назад я решил сделать небольшое SPA для себя — маленькая утилита, потенциальный pet-project. Работы дня на два-три. А последние полгода я работал над десктопным проектом на C#. Это была довольно скучная программулина для управления рабочими процессами: webservice-бэкенд и winforms на фронте.

Как только мне пришла в голову идея написать маленькое web-приложение, я сразу решил опробовать на нем несколько модных фишек, о которых слышал, обновить свой инструментарий, а еще просто немного развлечься. Вроде ничего сложного.

Как оказалось, я не смог написать код для этого простого приложения, потому что впал в "исследовательский ступор"

Я начинал и бросал уже раз пять. Проблема была в *выборе* и удручающем обилии инструментов из которых надо было выбирать.

Кто захочет писать

```
MyNotReallyClass.prototype.getCarrots = function () {}
```

когда ES6 уже *почти* здесь, со своими *почти* классами, которые уже *вот-вот* будут поддерживаться?
Кто захочет вставлять десять

```
<script src=”library-12.js”></script>
```

сверху страницы, когда есть множество утилит для упаковки кода в бандлы?
А кто захочет писать

```
$('.carrots').innerHTML(myJson.some.property[3])
```

когда вокруг так много фреймворков, которые помогают структурировать код? Кто хочет игнорировать тот факт, что теперь клиентский код компилируется с помощью утилит, написанных на Node.js?

Так что я решил полностью погрузиться во все эти новые штуки, вспомнить те, которые забыл и так далее. И знаете что — я не продвинулся дальше пары HTML форм.

Напомню, это был простой проект "для себя", я вообще-то в основном собирался получить удовольствие, так что настроился на "zero-tolerance mode" (прим. переводчика: рука не поднялась

переводить такую чудесную фразу). Как только что-то начинало раздражать, я бросал это и искал другое, лишь бы ничего не мешало.

Вот пара примеров того, с чем мне пришлось столкнуться, пока я пробовал все эти новые штуки из современного Javascript.

Для начала, я захотел попробовать Typescript. Поскольку последнее время я работал в основном с C#, я помнил, как это клево, когда язык статически типизирован: это дает больше уверенности в своем коде, рефакторинг становится проще, а IDE с хорошим автокомплитом пишут за тебя половину кода, и не важно, как ты намудрил с классами.

Мне были нужны две библиотеки, для основного функционала. Они не были написаны на Typescript, так что я потратил пол-дня на изучение `.d.ts` файлов и написание оберток к тем библиотекам. Не самое продуктивное занятие, но ладно.

Для начала, я захотел написать немного тестов с **Mocha**. Добро пожаловать в ад. Я поискал способ, как завести несколько `.tsconfig.json` файлов в проекте, но WebStorm не поддерживал такое, так что компилятор продолжал смешивать тесты с кодом. Я начал читать маны, примеры кода, **StackOverflow**. *Используйте вот этот **Gulp** конфиг. Вам нужно скомпилировать код перед тем как тестировать, но подождите, вы же пишете тесты на Typescript? Тогда используйте вот этот Gulp плагин, правда он*

*не очень хорошо работает с **watchify***. На следующий день у меня была каша из скомпилированных и склеенных файлов, папки `src`, `dest` и `test`, которые триггерили непонятные таски. Я перестал понимать, что происходит в фоне. Что где скомпилировалось, где лежат зависимости, должен ли я написать `require` или `import` или `reference` чтобы использовать вон тот файл? *Да пошло оно все*.

У меня было короткое, но приятное знакомство с **React** на одном очень маленьком проекте, и сейчас я решил попробовать его. Даже наскреб каких-то Gulp конфигов для быстрого старта. Но тут проблема оказалась с самим React. Я всегда стараюсь делать чистые модели, а React любит смешивать модели с `state` и `properties`, так что мне пришлось переосмысливать свои подходы. Мое приложение довольно простое, но использует много форм, а теперь угадайте, что я прочитал в официальной документации:

> Если вы впервые изучаете фреймворк, обратите внимание, что `ReactLink` не нужен в большинстве приложений, и должен использоваться осторожно.
> В React реализован однонаправленный поток данных: от родительского к дочернему элементу. Это следует из модели фон Неймана. Вы можете себе это представить как "однонаправленное связывание данных"

Отлично, но формам, особенно сложным, вообще-то очень нужен

`two-way binding`. A React без плагинов и миксинов не очень хорошо работает, когда ты пытаешься использовать его для ввода большого количества данных. Тебе нужно написать декораторы для всех своих инпутов, чтобы все работало нормально. Очень скоро это начинает раздражать. Кстати, о миксинах, я использую классы из ES6, но React не поддерживает их. *Да пошло оно все*.

Так что же, мне нужен `two-way binding`, правильно? **Knockout** для этого подходит как нельзя лучше, и я имел с ним раньше дело. И снова, я пытаюсь использовать классы ES6, но связывание контекстов запутывает код. А без классов код запутывается еще быстрее. Javascript мешается с HTML и все это выглядит ужасно после React. Вот у вас верстка в Javascript, но это хотя бы имеет смысл, пока помогает структурировать код и держать логические элементы изолированно друг от друга.

(Немного о боли. Похоже, вне зависимости от того, что вы выберете, вам придется держать в фоне какие-то `watch` таски для компиляции, линтовки, тестирования. Я что-то написал в редакторе, `Cmd+S` сохранил, `Cmd+Tab` перешел в браузер, `Cmd+R` обновил страничку, и не вижу изменений. Угадайте что произошло? Я сделал все эти манипуляции быстрее, чем компилятор отработал, или `watch` таск не заметил моих изменений, или выкинул какую-нибудь ошибку.)

Итак, еще перед тем, как написать первую строчку кода, вам приходится выбрать, чем вы все это сдобрите (пакетный

менеджер, сборка, тестирование и т.д.), и каждая опция открывает врата в царство альтернативных решений, более или менее стандартизированных, более или менее дополняющих или заменяющих друг друга.

Javascript с каким вкусом желаете? Хотите транспайлер? А из какого языка? Grunt? Gulp? Bower? Yeoman? Browserify? Webpack? Babel? Common.js? Amd? Angular? Ember? Linting? О чем я сейчас вообще? Я что-то напутал? Да пошло оно все? Да, пошло оно все.

Если вы еще тут, похоже, что сделать web-приложение сегодня, это как игра в очень сложный интерактивный квест (ориг. Interactive Fiction) (а-ля Zork). Давайте вернемся в недалекое прошлое и напишем программу:

```
Вы в комнате с программами. Вы можете сделать только консольную
программу. Вы видите язык C и язык Ассемблер
> get C language
Молодец, можешь писать свою программу
```

А теперь, вернемся обратно в 2016:

```
Вы в комнате с web-приложениями
> make web app
Вы делаете игру? Попробуйте "Unity" или "GameMaker", или еще похожий
софт для web
> make web app
А вы не думали сделать десктопное приложение на web-технологиях?
Попробуйте "NW.js" или, вот, "Electron"
> make web app
Появляется несколько языков. Вы видете "Javascript", "Coffescript",
"Typescript", "Clojurescript", "Dart", "asm.js".
Показать остальные 127 вариантов?
> get javascript
Появляется несколько языков: "ES5" и "ES6"
```

```
> get ES6
Вы в комнате с транспайлерами. Вы видите "Babel", "Traceur" и можете
только надеяться, что браузеры уже поддерживают фичи, которые вы
собираетесь использовать.
Показать весь список транспайлеров? Не забудьте также прочитать
книгу "Транспайлеры aka Некрономикон"
> get Babel
Вы попадаете в коридор Таск раннеров. Вы видете "Grunt" в углу,
"Gulp" в другом. "Babelify" атакует вас, "Webpack" повсюду. В
соседней комнате вы слышите как "Browserify" вопит и дерется с
"Require.js". В вашем рюкзаке есть "транспайлинг по сохранению"
> бежим отсюда
"Yeoman" отсвечивает в соседней нише, у вас в руке есть "npm", но
"project.json" сломан, на полу вы видите "Gruntfile", ".jshintrc",
".babelrc" и "tsconfig.json". Вы слышите завывания "Broccoli" и
"Jasmine" вдалеке.
> Да пошло оно
Вы не можете просто "послать все", потому что семь комнат назад вы
выбрали "npm install node-jsx" и это несовместимо с вашей
конфигурацией "да пошло оно"
> выход.
```

update

Этот пост вызвал интерес на Hacker News. Некоторые комментарии мне кажутся немного ироничными:

> Я всегда стараюсь делать чистые модели, а React любит смешивает модели с `state` и `properties`, так что мне пришлось переосмысливать свои подходы...
>
> > Redux
>
> Gulp конфиг...
>
> > Webpack

В общем, разговор с комьюнити о моем "исследовательском ступоре", вызванном количеством доступных инструментов, вылился в советы попробовать и потратить время на еще парочку технологий, которые я упустил в начале своих поисков. Хорошая работа, Javascript!

Похожие публикации

Удаленная отладка JavaScript с VS2015. Часть 3 (F12 Chooser)

0

TrickyCat • 13 февраля 2016 в 15:01

Удаленная отладка JavaScript с VS2015. Часть 2

0

TrickyCat • 9 февраля 2016 в 19:00

Удаленная отладка JavaScript с VS2015. Часть 1

0

ИЗ ПЕСОЧНИЦЫ

TrickyCat • 19 января 2016 в 11:39

Популярное за сутки

Наташа — библиотека для извлечения структурированной информации из текстов на русском языке

14

alexkuku • вчера в 16:12

Unit-тестирование скриншотами: преодолеваем звуковой барьер. Расшифровка доклада

4

lahmatiy • вчера в 13:05

Люди не хотят чего-то действительно нового — они хотят привычное, но сделанное иначе

25

ПЕРЕВОД

Smileek • вчера в 10:32

Руководство по SEO JavaScript-сайтов. Часть 2. Проблемы, эксперименты и рекомендации

2

ПЕРЕВОД

ru_vds • вчера в 12:04

Как адаптировать игру на Unity под iPhone X к апрелю

0

P1CACHU • вчера в 16:13

Лучшее на Geektimes

Стивен Хокинг, автор «Краткой истории времени», умер на 77 году жизни

33

HostingManager • вчера в 13:49

Обзор рынка моноколес 2018

70

lozga • вчера в 06:58

«Битва за Telegram»: 35 пользователей подали в суд на ФСБ

40

alizar • вчера в 15:14

Стивен Хокинг и его работа — что дал ученый человечеству?

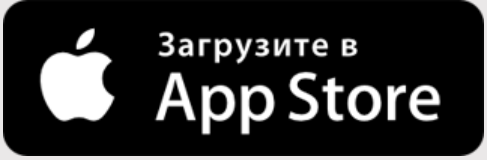
8

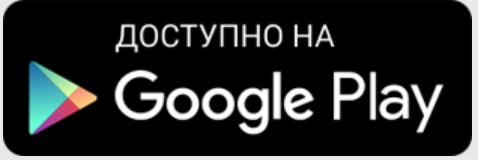
marks • вчера в 14:46

Sunlike — светодиодный свет нового поколения

17

AlexeyNadezhin • вчера в 20:32





Полная версия