Содержание

Благодарности[1]

Я говорил это в своей первой книге, и повторю здесь: труд писателя невероятно тяжел! Поверьте, никто не скажет, что это не так. Если же кто-то попробует опровергнуть мои слова, он либо лгун, либо Стивен Кинг (Stephen King). К счастью для меня, я нахожусь где-то посередине.

Работа над книгой, с одной стороны, это труд одного, независимого человека, а с другой – труд целой команды. Уложив детей спать, я устремляюсь в свой офис, вскрываю несколько банок Гиннесса (Guiness), завожу музыку и за работу, в уединении, поздно ночью. Закончив главу, я отсылаю ее своему редактору, а он рассылает ее множеству людей, которые берут мою писанину и улучшают ее. Улучшениями могут быть простые исправления грамматических и орфографических ошибок или что-то более сложное, например, помощь в улучшении организации книги или рекомендации по улучшению примеров программного кода, с целью сделать их более понятными. То есть, с одной стороны книга может писаться в одиночку, в темной комнате, но конечный продукт является результатом упорного труда множества людей.

В этом разделе книги я имею возможность сказать спасибо всем, кто помогал создавать, оформлять или как-то иначе обеспечивать высочайшее качество книги, которую вы сейчас держите в руках (или загрузили из Интернета). Без дальнейших обиняков я хочу выразить свои благодарности в стиле лауреата кинопремии Оскара: знайте, я уверен, что кого-то наверняка пропущу, о чем невероятно сожалею.

---

[1] Многие в моем издательстве считали, что мой раздел с выражением благодарности, а также другие части этой книги содержали слишком рискованные шутки, поэтому текст оригинала был исправлен до того, что вы видите перед собой. Я приношу свои извинения, если что-то из написанного мной оскорбило вас, это не было моей целью. Мне уже говорили, что мое чувство юмора понятно далеко не всем. Если вы ничего не имеете против неприличных шуток, следуйте за мной в Твиттере (учетная запись: @markbates).

Прежде всего я хочу поблагодарить мою красавицу жену Рейчел (Rachel). Рейчел – одна из самых участливых и сильных людей, когда-либо встречавшихся мне. Каждую ночь я ложусь с ней в постель, и каждое утро я радуюсь, просыпаясь рядом с ней. Мне доставляет удовольствие заглядывать в ее глаза и видеть там безграничную любовь. Она поддерживает меня, когда я пишу книги, поддерживала, когда я открывал свое дело, и делает все, чтобы моя жизнь была счастливее. Она дала мне двух красавцев сыновей, а я взамен даю ей скабрезные шутки и мои использованные сотовые телефоны. Я определенно отхватил самый лучший кусок в этой сделке, называемой браком, за что бесконечно благодарен ей.

Далее, я хочу поблагодарить моих сыновей, Дилана (Dylan) и Лео (Leo). Даже при том, что ни один из них не принимал прямого участия в создании книги, они придают ценность моей жизни, наполняют ее энергией и эмоциями, которые могут дать только дети. Я люблю вас, мальчики, очень-очень.

Прежде чем перейти от моей семьи к другим людям, я хотел бы сказать спасибо моим родителям (особенно тебе, Мама!) и всем остальным членам моей семьи за то, что всегда были рядом и одновременно поддерживали меня и возвращали к действительности. Я люблю вас всех.

Далее я хочу поблагодарить Дебру Вильямс Коли (Debra Williams Cauley). Дебра была моим редактором, наставником и психиатром, когда я работал над первой книгой «Distributed Programming with Ruby». Я могу только пожелать удачи другим авторам работать с таким же хорошим редактором, как Дебра. Она обладает по-настоящему ангельским терпением.

Надеюсь, если когда-нибудь мне доведется писать еще одну книгу, Дебра окажется рядом. Я не могу представить себе процесс создания книги без ее участия. Спасибо тебе, Дебра.

В создании технической литературы очень важную роль играют технические рецензенты. Работа технического рецензента заключается в том, чтобы читать главы и критиковать их с технической точки зрения, а также постоянно отвечать на вопрос: «Имеет ли смысл рассказывать об этом здесь?». Эти рецензенты выступают в роли ваших читателей. Они технически подкованы и знают свой предмет. Поэтому их отзывы имеют огромное значение. С этой книгой работало несколько технических рецензентов. Но особенно я хотел бы отметить двоих из них – Стюарта Гарнера (Stuart Garner) и Дена Пикетта (Dan Pickett). При работе над этой книгой, Стюарт и

Ден не ограничились лишь служебным долгом и не боялись прямо высказывать свое мнение, когда я делал или говорил какие-нибудь глупости. Они откликались на мои истеричные телефонные звонки и электронные письма в любое время дня и ночи и давали бесценные советы. Если бы я не хотел сам получить все эти суммы гонораров, проставленные в чеках, я не смог бы устоять перед соблазном включить в чеки и их фамилии. (Не волнуйтесь, их труд не остался неоплаченным. Каждый из них получил купон на один бесплатный час времени «Марка».) Спасибо вам, Ден и Стюарт, и всем остальным техническим рецензентам, за ваш кропотливый труд.

В эту книгу вложен труд многих людей. Кто-то занимался оформлением обложки, кто-то алфавитным указателем, кто-то создавал язык (CoffeeScript), и еще множество людей, так или иначе вовлеченных в процесс. Ниже приводится список некоторых из них (о ком я знаю) без какого-то определенного порядка: Джереми Ашкенас (Jeremey Ashkenas), Тревор Барнхам (Trevor Burnham), Ден Фишман (Dan Fishman), Крис Зан (Chris Zahn), Грегг Поллак (Gregg Pollack), Гари Адайр (Gary Adair), Сандра Шредер (Sandra Schroeder), Оби Фернандес (Obie Fernandez), Кристи Харт (Kristy Hart), Энди Бестер (Andy Beaster), Барбара Хача (Barbara Hacha), Тим Райт (Tim Wright), Дебби Вильямс (Debbie Williams), Брайен Франс (Brian France), Ванесса Эванс (Vanessa Evans), Ден Шерф (Dan Scherf), Гари Адайр (Gary Adair), Нони Ратклифф (Nonie Ratcliff) и Ким Бодигреймер (Kim Boedigheimer).

Я также хотел бы сказать спасибо всем, с кем встречался, пока работал над книгой, и кто выслушивал мою бесконечную болтовню об этом. Я знаю, многим это совсем не интересно, но, черт возьми, я обожаю звук моего голоса. Спасибо всем вам, что не врезали мне за мою болтливость, хотя я это, возможно, и заслуживаю.

Наконец, я хотел бы поблагодарить вас, мой читатель. Спасибо, что приобрели эту книгу и помогли поддержать всех, кто, как и я сам, по-настоящему хотят помочь своим собратьям разработчикам, поделившись своими знаниями с остальным миром. Именно для вас я вложил массу времени и сил в эту книгу. Я надеюсь, что когда вы перевернете последнюю страницу, вы будете лучше понимать язык CoffeeScript, и как он может повлиять на вашу работу. Удачи!

Об авторе

Марк Бейтс (Mark Bates) является основателем и главным архитектором консалтинговой компании Meta42 Labs со штаб-квартирой в Бостоне. Марк проводит дни, занимаясь разработкой новых приложений для своих клиентов и консультируя их. После работы он пишет книги, воспитывает детей, а иногда созывает свой ансамбль и «пытается сделать это».

Марк занимается разработкой веб-приложений, в том или ином виде, начиная с 1996 года. Свою карьеру он начинал, как разработчик пользовательского интерфейса, используя HTML и JavaScript, а затем переключился на разработку более сложного программного обеспечения на языках Java и Ruby. В настоящее время Марк тратит свое время, изменяя Ruby со своей новой возлюбленной – CoffeeScript.

Всегда желая поделиться своими знаниями, а точнее, просто услышать звук своего голоса, Марк неоднократно выступал на широко известных конференциях, таких как RubyConf, RailsConf, и jQueryConf. Кроме того, он преподавал на курсах изучения Ruby и Ruby on Rails. В 2009 году издательством Addison-Wesley была опубликована первая книга Марка (и самое замечательное, что не последняя!) «Distributed Programming with Ruby».

Марк живет в пригороде Бостона со своей супругой Рейчел и двумя сыновьями, Диланом и Лео. Марка можно найти в Веб, по адресам: http://www.markbates.com, http://twitter.com/markbates и http://github.com/markbates.

Предисловие

Моя профессиональная карьера разработчика началась в 1999 году, когда я получил первую зарплату как разработчик. (Я не считаю несколько лет, когда я просто получал удовольствие, играя с Веб.) В 1999 году Всемирная паутина была жутким местом. HTML-файлы были перегружены тегами `font` и `table`. Каскадные таблицы стилей CSS только-только начали выходить на сцену. Язык JavaScript [1] существовал всего несколько лет, а война браузеров была в самом разгаре. Безусловно, тогда можно было написать JavaScript-сценарий, выполняющий некоторые операции в одном браузере, но смог бы он работать в другом? Скорее всего, нет. Из-за этого в 2000 годах язык JavaScript получил дурную славу.

В середине 2000-х произошло два важных события, которые помогли JavaScript подняться в глазах разработчиков. Первым из них было появление технологии AJAX. [2] Технология AJAX позволяет разработчикам создавать более быстрые и более интерактивные вебстраницы, благодаря возможности отправлять запросы на сервер в фоновом режиме и устранению необходимости для конечного пользователя обновлять содержимое окна браузера.

Вторым событием стало появление популярных библиотек на JavaScript, таких как Prototype, [3] которые существенно упростили создание JavaScript-сценариев, совместимых со всеми типами браузеров. Появилась возможность использовать технологию AJAX, чтобы сделать приложения более интерактивными и удобными в использовании, и задействовать библиотеку, такую как Prototype, чтобы обеспечить совместимость с основными типами браузеров.

В 2010 году, а точнее в 2011, развитие Всемирной паутины пошло по пути создания «одностраничных» приложений. Такие приложения выполняются под управлением JavaScript-фреймворков, таких как Backbone.js. [4] Эти фреймворки позволяют применять шаблон проектирования MVC [5] с использованием JavaScript. Стало возможным писать на JavaScript целые приложения, а затем загружать их и выполнять в браузере конечного пользователя. Все вместе это

позволяет писать поразительно интерактивные и полнофункциональные клиентские приложения.

Однако, с точки зрения разработчика, ситуация выглядела не так радужно. Несмотря на то, что фреймворки и инструменты значительно упростили разработку подобных приложений, сам язык JavaScript оставался болезненным местом. Язык JavaScript является одновременно невероятно мощным, и в то же время чрезвычайно запутанным. Он полон парадоксов и ловушек, которые быстро могут сделать ваш программный код неконтролируемым и наполненным ошибками.

Так чего же хотят разработчики? Они хотят создавать эти замечательные новые приложения, но единственным языком, который понимают все браузеры, является JavaScript. Конечно, они могут писать эти приложения на Flash, [6] но для этого в браузеры необходимо устанавливать расширения, к тому же эти расширения отсутствуют для некоторых платформ, таких как устройства на iOS [7].

Впервые с языком CoffeeScript [8] я столкнулся в октябре 2010 года. CoffeeScript давал мне надежду приручить JavaScript и подчеркивал наиболее выгодные стороны замысловатого языка, каковым является JavaScript. Он имеет ясный синтаксис, отдавая предпочтение пробелам вместо знаков пунктуации, и защищает от ловушек, поджидающих JavaScript-разработчиков на каждом шагу, таких как неочевидные правила видимости и неправильное употребление операторов сравнения. Но самое замечательное, что в конечном итоге программный код на CoffeeScript компилируется в стандартный программный код на JavaScript, который может выполняться в любом браузере или в другой среде выполнения JavaScript.

Когда я впервые попробовал использовать CoffeeScript, язык был еще далек от совершенства, даже в версии 0.9.4. Я использовал его в проекте моего клиента, чтобы попробовать и увидеть, является ли правдой все, что я слышал о нем. К сожалению, две причины заставили меня отложить его в сторону. Во-первых, язык еще не был готов к широкому использованию. В нем было слишком много ошибок и в нем отсутствовали многие возможности.

Вторая причина, заставившая меня отказаться от CoffeeScript, заключалась в том, что приложение, на котором я проводил эксперименты, не было настоящим JavaScript-приложением. Мне требовалось реализовать лишь кое-какие проверки и организовать отправку запросов с использованием технологии AJAX, что, благодаря

помощи Ruby on Rails [9] достигалось совсем небольшим объемом программного кода на JavaScript.

Так что же заставило меня вернуться к CoffeeScript? Спустя примерно шесть месяцев после первого знакомства с CoffeeScript, было объявлено, [10] что Rails 3.1 будет распространяться вместе с CoffeeScript, в качестве механизма JavaScript по умолчанию. Как и большинство разработчиков, я был ошеломлен этой новостью. Я пытался использовать язык CoffeeScript и не считал его чем-то выдающимся. О чем они думали?

В отличие от большинства моих собратьев разработчиков я решил уделить время, чтобы по-новому взглянуть на CoffeeScript. Шесть месяцев – достаточно долгий срок в разработке любого проекта. Язык CoffeeScript проделал длинный, очень длинный путь. И я решил еще раз попробовать использовать его, на этот раз в приложении, содержащем достаточно большой объем программного кода на JavaScript. Спустя несколько дней повторного использования CoffeeScript, я не только изменил свои взгляды, но и полюбил этот язык.

Не могу сказать точно, что повлияло на мои убеждения, и не буду пытаться объяснить, почему я полюбил этот язык. Я хочу, чтобы вы сами сформировали свое мнение о нем. Надеюсь, что в ходе чтения этой книги вы не только станете приверженцами, но и активными сторонниками этого замечательного маленького языка по вашим собственным причинам. А я коротко расскажу вам о том, что ждет вас впереди. Ниже приводится небольшой фрагмент программного кода на CoffeeScript из действующего приложения, а вслед за ним – эквивалентный фрагмент на JavaScript. Наслаждайтесь!

Пример: (исходный файл: `sneak_peak.coffee`)

```
@updateAvatars = ->
    names = $('.avatar[data-name]').map -> $(this).data('name')
      Utils.findAvatar(name) for name in $.unique(names)
```

Пример: (исходный файл: `sneak_peak.js`)

```
(function() {
    this.updateAvatars = function() {
        var name, names, _i, _len, _ref, _results;
        names = $('.avatar[data-name]').map(function() {
            return $(this).data('name');
        });
        _ref = $.unique(names);
```

```
        _results = [];
        for (_i = 0, _len = _ref.length; _i < _len; _i++) {
            name = _ref[_i];
            _results.push(Utils.findAvatar(name));
        }
        return _results;
    };
}).call(this);
```

Что такое CoffeeScript?

CoffeeScript – это язык программирования, программный код на котором компилируется в программный код на языке JavaScript. Не слишком понятно, знаю, но это так и есть. Язык CoffeeScript близко напоминает такие языки программирования, как Ruby [11] и Python. [12] Он создавался с целью помочь разработчикам повысить производительность труда при разработке программного кода на JavaScript. За счет избавления от необходимости использовать знаки пунктуации, такие как скобки, точки с запятой, и прочие, и использования значимых пробелов взамен этих символов, вы сможете быстро сосредотачиваться на программном коде, а не на бесконечных проверках – расставлены ли все закрывающие фигурные скобки.

Представьте, что вы пишите такой код на JavaScript:

Пример: (исходный файл: `punctuation.js`)

```
(function() {
    if (something === something_else) {
        console.log('do something');
    } else {
        console.log('do something else');
    }
}).call(this);
```

Почему бы не записать его так:

Пример: (исходный файл: `punctuation.coffee`)

```
if something is something_else
    console.log 'do something'
else
    console.log 'do something else'
```

Кроме того, язык CoffeeScript предоставляет ряд сокращенных форм записи, позволяющих записывать сложные фрагменты программного кода на JavaScript гораздо короче. Например, следующий фрагмент позволяет обойти в цикле значения, находящиеся в массиве, не заботясь об их индексах:

Пример: (исходный файл: `array.coffee`)

```
for name in array
      console.log  name
```

Пример: (исходный файл: `array.js`)

```
(function() {
    var name, _i, _len;
    for (_i = 0, _len = array.length; _i < _len; _i++) {
        name = array[_i];
        console.log(name);
    }
}).call(this);
```

В довесок к синтаксическому сахару, язык CoffeeScript помогает также писать более надежный программный код на JavaScript, следя за видимостью переменных и классов, гарантируя использование соответствующих операторов сравнения, и беря на себя решение многих других рутинных задач, в чем вы убедитесь, читая эту книгу.

Языки CoffeeScript, Ruby и Python часто упоминаются вместе и на то есть свои причины. За основу модели CoffeeScript были взяты краткость и простота синтаксиса этих языков. Благодаря этому CoffeeScript создает «ощущение» более современного языка, чем JavaScript, за основу которого были взяты такие языки, как Java [13] и C++. [14] Как и JavaScript, язык CoffeeScript можно использовать в любом программном окружении. Не имеет никакого значения, на каком языке вы пишете свое приложение, Ruby, Python, PHP, [15] Java или .Net [16]. Программный код, скомпилированный в JavaScript, будет работать со всеми ими.

Поскольку программный код на CoffeeScript компилируется в программный код на JavaScript, сохраняется возможность использовать любые библиотеки на JavaScript. Вы можете использовать jQuery, [17] Zepto, [18] Backbone, [19] Jasmine [20] или любую другую библиотеку, и все они будут работать. Такое не слишком часто говорят о новых языках.

Звучит неплохо, но я слышу, как вы спрашиваете: а есть ли недостатки у CoffeeScript, в сравнении со старым добрым JavaScript? Отличный вопрос. Ответ на него: не так много. Во-первых, несмотря на то, что CoffeeScript предоставляет по-настоящему замечательный способ разработки программного кода на JavaScript, он не дает ничего сверх того, что возможно в JavaScript. Например, я по-прежнему не могу создать JavaScript-версию знаменитого метода *method_missing* в языке Ruby. [21] Самый большой недостаток заключается в том, что вам и членам вашей команды придется учить новый язык. К счастью это легко поправимо. Как вы сами увидите, CoffeeScript чрезвычайно прост в изучении.

Наконец, если по каким-либо причинам выбор CoffeeScript окажется ошибочным для вас или вашего проекта, вы сможете взять сгенерированный программный код на JavaScript и работать с ним. Поэтому в действительности вы ничего не потеряете, если попробуете использовать CoffeeScript в следующем или даже в текущем своем проекте (CoffeeScript и JavaScript очень хорошо уживаются друг с другом).

Кому адресована эта книга?

Эта книга адресована разработчикам на JavaScript со средним и высоким уровнем подготовки. Есть несколько причин, по которым я не могу рекомендовать эту книгу тем, кто не знаком с JavaScript, или тем, кто имеет лишь общие представления о нем.

Во-первых, эта книга не учит программированию на языке JavaScript. Эта книга рассказывает о CoffeeScript. Попутно вы, конечно, узнаете кое-что о JavaScript (и CoffeeScript обладает особым талантом заставлять узнавать больше о JavaScript), но мы не будем погружаться в основы JavaScript и постепенно изучать его.

Пример: что делает следующий фрагмент? (исходный файл: `example.js`)

```
(function() {
    var array, index, _i, _len;
    array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
    for (_i = 0, _len = array.length; _i < _len; _i++) {
        index = array[_i];
        console.log(index);
    }
}).call(this);
```

Если вам не удалось понять, что делает этот фрагмент, я рекомендую прервать чтение книги на этом. Не переживайте, я действительно хочу, чтобы вы вернулись и продолжили чтение. Просто я полагаю, что вы получите от книги больше, если будете иметь более близкое знакомство с языком JavaScript. Я буду рассказывать о некоторых основных особенностях языка JavaScript, чтобы проиллюстрировать или помочь лучше понять, что делает тот или иной фрагмент на CoffeeScript. Несмотря на то, что для ясности изложения будут охватываться некоторые основы JavaScript, действительно важно, чтобы вы получили базовые знания о языке JavaScript прежде, чем продолжить чтение. Поэтому, пожалуйста, найдите хорошую книгу о JavaScript (их существует великое множество), прочитайте ее и снова присоединяйтесь ко мне, чтобы стать гуру CoffeeScript.

Тем, кто уже является рок-звездой в JavaScript, я предлагаю поднять уровень игры. Эта книга расскажет вам, как писать ясный, более краткий и надежный программный код на JavaScript, используя вкусности CoffeeScript.

Как читать эту книгу

Я попытался так организовать материал в этой книге, чтобы помочь вам построить фундамент в освоении CoffeeScript. Главы в первой части следует читать по порядку, потому что каждая следующая глава основана на понятиях, изучаемых в предыдущих главах, поэтому, пожалуйста, не перепрыгивайте через главы.

В процессе чтения каждой главы вы заметите несколько обстоятельств.

Во-первых, когда я буду представлять какую-нибудь внешнюю библиотеку, новую идею или понятие, я буду включать ссылку на веб-сайт, где вы сможете получить дополнительную информацию о предмете обсуждения. Я очень хотел бы рассказать вам о многом, например, о языке Ruby, однако в книге недостаточно места для этого. Поэтому, если я буду говорить о чем-то, и вы пожелаете познакомиться с этим поближе, прежде чем продолжить чтение, отправляйтесь по указанной мной ссылке, утолите свою жажду познания и возвращайтесь к книге.

Во-вторых, в каждой главе, я время от времени буду сначала показывать неправильное решение проблемы. После знакомства с неправильным путем, мы исследуем его, чтобы понять, что именно в нем неправильно, а затем найдем правильное решение проблемы.

Отличный пример такого подхода встретится вам в главе 1, «Введение», где рассказывается о различных способах компиляции программного кода на CoffeeScript в программный код на JavaScript.

Иногда в книге вам будут встречаться такие примечания:

> **Совет. Какой-нибудь полезный совет.** Так будут оформляться небольшие советы и рекомендации, которые, на мой взгляд, могут вам пригодиться.

Наконец, на протяжении всей книги я буду представлять примеры программного кода сразу по два-три блока. Сначала будет приводиться пример на языке CoffeeScript, а затем скомпилированная версия (на JavaScript) того же примера. Если в процессе выполнения пример выводит какую-нибудь информацию (на мой взгляд, что-нибудь важное), я буду включать также вывод примера. Ниже показано, как это выглядит:

Пример: (исходный файл: `example.coffee`)

```
array = [1..10]
for index in array
      console.log  index
```

Пример: (исходный файл: `example.js`)

```
(function() {
    var array, index, _i, _len;
    array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
    for (_i = 0, _len = array.length; _i < _len; _i++) {
        index = array[_i];
        console.log(index);
    }
}).call(this);
```

Вывод: (исходный файл: `example.coffee`)

```
1
2
3
4
5
6
7
```