



# Содержание

<b>Предисловие к русскому изданию</b> .....	15
<b>Благодарности</b> .....	16
<b>Предисловие</b> .....	17
<b>Глава 1. Вниз по кроличьей норе</b> .....	26
Почему Clojure? .....	26
Как получить Clojure .....	29
Интерактивная оболочка REPL для Clojure .....	30
Вам не придется путаться в частоколе скобок .....	34
Выражения, операторы, синтаксис и очередность .....	35
Гомоиконность .....	38
Механизм чтения.....	41
Скалярные литералы .....	43
Строки .....	43
Логические значения .....	43
nil .....	43
Знаки (characters) .....	44
Ключевые слова (keywords) .....	44
Символы (symbols) .....	46
Числа .....	46
Регулярные выражения .....	48
Комментарии .....	49
Пробелы и запятые .....	51
Литералы коллекций .....	51
Прочий синтаксический сахар механизма чтения .....	52
Пространства имен .....	53
Интерпретация символов .....	56
Специальные формы .....	57
Подавление вычислений: quote .....	59
Блоки кода: do .....	60
Определение переменных: def .....	61

Связывание локальных значений: let .....	62
Деструктуризация (let, часть 2).....	64
Деструктуризация упорядоченных коллекций .....	65
Деструктуризация ассоциативных массивов .....	69
Создание функций: fn.....	74
Деструктуризация аргументов функций .....	77
Литералы функций .....	80
Выполнение по условию: if .....	82
Организация циклов: loop и recur .....	83
Ссылки на переменные: var .....	85
Взаимодействие с Java: . и new .....	86
Обработка исключений: try и throw .....	86
Специализированная операция set! .....	87
Примитивы блокировок: monitor-enter и monitor-exit .....	87
Все вместе .....	87
eval.....	88
Это лишь начало.....	90

## **Часть I. ФУНКЦИОНАЛЬНОЕ И КОНКУРЕНТНОЕ ПРОГРАММИРОВАНИЕ .....**

91

### **Глава 2. Функциональное программирование .....**

92

Что подразумевается под термином «Функциональное программирование»? .....	93
О важности значений.....	94
О значениях .....	95
Сравнение значений изменяемых объектов .....	96
Важность выбора .....	101
Функции, как сущности первого порядка, и функции высшего порядка .....	103
Частичное применение .....	111
Композиция функций .....	116
Создание функций высшего порядка.....	120
Создание простейшей системы журналирования с применением композиции функций высшего порядка .....	121
Чистые функции .....	126
В чем преимущество чистых функций? .....	129
Функциональное программирование в реальном мире .....	132

### **Глава 3. Коллекции и структуры данных.....**

134

Главенство абстракций над реализациями .....	135
Коллекции .....	139
Последовательности .....	142

Последовательности не являются итераторами .....	145
Последовательности не являются списками .....	146
Создание последовательностей .....	147
Ленивые последовательности .....	148
Удержание мусора .....	156
Ассоциативные коллекции .....	157
Берегитесь значения nil .....	161
Индексирование .....	162
Стек .....	164
Множество .....	165
Сортированные коллекции .....	166
Определение порядка с помощью компараторов и предикатов .....	168
Упрощенный доступ к коллекциям .....	173
Идиоматические приемы использования .....	175
Коллекции, ключи и функции высшего порядка .....	176
Типы структур данных .....	177
Списки .....	178
Векторы .....	179
Векторы как кортежи .....	180
Множества .....	181
Ассоциативные массивы .....	182
Ассоциативные массивы как специализированные структуры .....	183
Другие применения ассоциативных массивов .....	185
Неизменяемость и сохранность .....	189
Сохранность и совместное использование .....	190
Визуализация сохранности: списки .....	191
Визуализация сохранности: ассоциативные массивы (векторы и множества) .....	193
Очевидные преимущества .....	196
Переходные структуры данных .....	198
Метаданные .....	205
Включаем коллекции Clojure в работу .....	207
Идентификаторы и циклы .....	208
Думайте иначе: от императивного к функциональному .....	210
Вспоминаем классику: игра «Жизнь» .....	211
Генерация лабиринтов .....	220
Навигация, изменение и зипперы (zippers) .....	228
Манипулирование зипперами .....	229
Собственные зипперы .....	231
Зиппер Ариадны .....	232
В заключение .....	236

<b>Глава 4. Конкуренция и параллелизм</b> .....	237
Сдвиг вычислений в пространстве и времени.....	238
delay.....	238
Механизм future .....	241
Механизм promise .....	243
Параллельная обработка по невысокой цене.....	246
Состояние и идентичность .....	250
Ссылочные типы.....	253
Классификация параллельных операций.....	255
Атомы.....	258
Уведомление и ограничение.....	261
Функции-наблюдатели .....	261
Функции-валидаторы .....	264
Ссылки.....	266
Программная транзакционная память .....	266
Механика изменения ссылок .....	268
Функция alter.....	271
Уменьшение конфликтов в транзакциях с помощью commute ...	273
Затирание состояния ссылки с помощью ref-set.....	279
Проверка локальной согласованности с помощью валидаторов.....	279
Острые углы программной транзакционной памяти .....	283
Функции с побочными эффектами строго запрещены .....	283
Минимизируйте продолжительность выполнения транзакций.....	284
Читающие транзакции могут повторяться .....	287
Искажение при записи .....	289
Переменные.....	291
Определение переменных.....	292
Приватные переменные .....	293
Строки документации.....	294
Константы .....	295
Динамическая область видимости .....	296
Переменные в языке Clojure не являются переменными в классическом понимании .....	303
Опережающие объявления.....	305
Агенты.....	307
Обработка ошибок в заданиях агентов .....	310
Режимы и обработчики ошибок в агентах .....	312
Ввод/вывод, транзакции и вложенная передача заданий.....	313
Сохранение состояний ссылок в журнале на основе агента....	315
Использование агентов для распределения нагрузки.....	318
Механизмы параллельного выполнения в Java .....	328
Блокировки .....	329
В заключение .....	330

<b>Часть II. СОЗДАНИЕ АБСТРАКЦИЙ</b> .....	331
<b>Глава 5. Макросы</b> .....	332
Что такое макрос? .....	333
Чем не являются макросы .....	335
Что могут макросы, чего не могут функции? .....	336
Сравнение макросов и механизма eval в Ruby .....	339
Пишем свой первый макрос .....	341
Отладка макросов .....	343
Функции развертывания макросов .....	344
Синтаксис .....	346
Сравнение quote и syntax-quote .....	348
unquote и unquote-splicing .....	349
Когда следует использовать макросы .....	351
Гигиена .....	353
Генераторы символов во спасение .....	355
Предоставление пользователю права выбора имен .....	359
Двукратное вычисление .....	360
Распространенные идиомы и шаблоны макросов .....	362
Неявные аргументы: &env и &form .....	364
&env .....	364
&form .....	367
Вывод сообщений об ошибках в макросах .....	367
Сохранение определений типов, сделанных пользователем .....	370
Тестирование контекстных макросов .....	373
Подробности: -> и ->> .....	375
В заключение .....	379
<b>Глава 6. Типы данных и протоколы</b> .....	380
Протоколы .....	381
Расширение существующих типов .....	383
Определение собственных типов .....	389
Записи .....	392
Конструкторы и фабричные функции .....	396
Когда использовать ассоциативные массивы, а когда записи ...	398
Типы .....	399
Реализация протоколов .....	402
Встроенная реализация .....	403
Встроенные реализации интерфейсов Java .....	405
Определение анонимных типов с помощью reify .....	407
Повторное использование реализаций .....	408
Интроспекция протоколов .....	413
Пограничные случаи использования протоколов .....	415

Поддержка абстракций коллекций.....	417
В заключение .....	427
<b>Глава 7. Мультиметоды.....</b>	<b>428</b>
Основы мультиметодов .....	428
Навстречу иерархиям .....	431
Иерархии .....	434
Независимые иерархии.....	437
Сделаем выбор по-настоящему множественным! .....	441
Кое что еще .....	443
Множественное наследование .....	443
Интроспекция мультиметодов .....	445
type и class; или месь ассоциативного массива .....	446
Функции выбора не имеют ограничений.....	447
В заключение .....	449
<b>Часть III. ИНСТРУМЕНТЫ, ПЛАТФОРМЫ И ПРОЕКТЫ .....</b>	<b>450</b>
<b>Глава 8. Создание и организация проектов на Clojure .....</b>	<b>451</b>
География проекта .....	451
Определение и использование пространств имен.....	452
Пространства имен и файлы .....	461
Знакомство с classpath .....	465
Местоположение, местоположение, местоположение .....	467
Организация программного кода по функциональным признакам ...	469
Основные принципы организации проектов .....	471
Сборка .....	472
Предварительная компиляция.....	473
Управление зависимостями .....	476
Модель Maven управления зависимостями .....	477
Артефакты и координаты.....	477
Репозитории .....	479
Зависимости.....	480
Инструменты сборки и шаблоны настройки.....	483
Maven.....	484
Leiningen .....	488
Настройка предварительной компиляции .....	491
Сборка гибридных проектов.....	493
В заключение .....	496
<b>Глава 9. Java и взаимодействие с JVM .....</b>	<b>497</b>
JVM – основа Clojure.....	498
Использование классов, методов и полей Java .....	499

Удобные утилиты взаимодействий .....	503
Исключения и обработка ошибок .....	506
Отказ от контролируемых исключений .....	509
with-open, прощай finally .....	510
Указание типов для производительности .....	512
Массивы .....	518
Определение классов и реализация интерфейсов .....	519
Экземпляры анонимных классов: proху .....	520
Определение именованных классов .....	523
gen-class .....	524
Аннотации .....	532
Создание аннотированных тестов для JUnit .....	533
Реализация конечных точек веб-службы JAX-RS .....	534
Использование Clojure из Java .....	537
Использование классов, созданных с помощью deftype и defrecord .....	541
Реализация интерфейсов протоколов .....	544
Сотрудничество .....	546

## **Глава 10. REPL-ориентированное программирование .....**

Интерактивная разработка .....	547
Постоянное изменяющееся окружение .....	552
Инструменты .....	554
Оболочка REPL .....	555
Интроспекция пространств имен .....	557
Eclipse .....	560
Emacs .....	563
clojure-mode и paredit .....	564
inferior-lisp .....	565
SLIME .....	567
Отладка, мониторинг и исправление программ в REPL во время эксплуатации .....	570
Особые замечания по поводу «развертываемых» оболочек REPL .....	574
Ограничения при переопределении конструкций .....	576
В заключение .....	579

## **Часть IV. ПРАКТИКУМ .....**

### **Глава 11. Числовые типы и арифметика .....**

Числовые типы в Clojure .....	581
В Clojure предпочтение отдается 64-битным (или больше) представлениям .....	583
Clojure имеет смешанную модель числовых типов .....	583

Рациональные числа .....	586
Правила определения типа результата .....	587
Арифметика в Clojure .....	589
Ограниченная и произвольная точность .....	589
Неконтролируемые операции .....	593
Режимы масштабирования и округления в операциях с вещественными числами произвольной точности.....	595
Равенство и эквивалентность .....	597
Идентичность объектов (identical?).....	597
Равенство ссылок (=).....	598
Числовая эквивалентность (==) .....	600
Эквивалентность может защитить ваш рассудок.....	601
Оптимизация производительности операций с числами .....	603
Объявление функций, принимающих и возвращающих значения простых типов .....	604
Ошибки и предупреждения, вызванные несоответствием типов .....	608
Используйте простые массивы осмысленно.....	610
Механика массивов значений простых типов .....	613
Автоматизация указания типов в операциях с многомерными массивами .....	618
Визуализация множества Мандельброта в Clojure .....	620
<b>Глава 12. Шаблоны проектирования .....</b>	<b>629</b>
Внедрение зависимостей.....	631
Шаблон Стратегия (Strategy) .....	636
Цепочка обязанностей (Chain of Responsibility) .....	638
Аспектно-ориентированное программирование .....	642
В заключение .....	647
<b>Глава 13. Тестирование.....</b>	<b>648</b>
Неизменяемые значения и чистые функции .....	648
Создание фиктивных значений.....	649
clojure.test .....	651
Определение тестов .....	653
«Комплекты» тестов .....	656
Крепления (fixtures) .....	658
Расширение HTML DSL.....	662
Использование контрольных проверок.....	668
Предусловия и постусловия .....	670
<b>Глава 14. Реляционные базы данных .....</b>	<b>673</b>
clojure.java.jdbc .....	673
Подробнее о with-query-results .....	678



Транзакции.....	680
Пулы соединений .....	681
Korma.....	682
Вступление .....	683
Запросы .....	685
Зачем использовать предметно-ориентированный язык? .....	686
Hibernate .....	689
Настройка .....	690
Сохранение данных.....	694
Выполнение запросов.....	695
Избавление от шаблонного кода .....	695
В заключение .....	698
<b>Глава 15. Нереляционные базы данных.....</b>	<b>699</b>
Настройка CouchDB и Clutch .....	700
Простейшие CRUD-операции .....	701
Представления.....	703
Простое представление (на JavaScript) .....	704
Представления на языке Clojure .....	706
_changes: использование CouchDB в роли очереди сообщений .....	711
Очереди сообщений на заказ .....	713
В заключение .....	717
<b>Глава 16. Clojure и Веб.....</b>	<b>718</b>
«Стек Clojure» .....	718
Основа: Ring.....	720
Запросы и ответы.....	721
Адаптеры .....	724
Обработчики .....	725
Промежуточные функции .....	727
Маршрутизация запросов с помощью Compojure .....	729
Обработка шаблонов.....	743
Enlive: преобразование HTML с применением селекторов.....	745
Попробуем воду .....	746
Селекторы .....	748
Итерации и ветвление .....	750
Объединяем все вместе.....	752
В заключение .....	756
<b>Глава 17. Развертывание веб-приложений на Clojure.....</b>	<b>758</b>
Веб-архитектура Java и Clojure .....	758
Упаковка веб-приложения .....	762
Сборка .war-файлов с помощью Maven .....	764

Сборка .war-файлов с помощью Leiningen .....	767
Запуск веб-приложений на локальном компьютере .....	769
Развертывание веб-приложения .....	770
Развертывание приложений на Clojure с помощью	
Amazon Elastic Beanstalk .....	771
За пределами развертывания простых веб-приложений .....	775
<b>Часть V. РАЗНОЕ .....</b>	<b>776</b>
<b>Глава 18. Выбор форм определения типов .....</b>	<b>777</b>
<b>Глава 19. Внедрение Clojure .....</b>	<b>780</b>
Только факты.....	780
Подчеркните особую продуктивность .....	782
Подчеркните широту сообщества.....	784
Будьте благоразумны .....	786
<b>Глава 20. Что дальше? .....</b>	<b>788</b>
(dissoc Clojure 'JVM).....	788
ClojureCLR.....	788
ClojureScript .....	789
4Clojure .....	790
Overtone .....	790
core.logic .....	791
Pallet .....	792
Avout .....	793
Clojure на платформе Heroku .....	793
<b>Об авторах.....</b>	<b>795</b>
Иллюстрация на обложке .....	796
<b>Предметный указатель .....</b>	<b>797</b>



## Предисловие к русскому изданию

Вы держите в руках новую книгу, выпущенную издательством «ДМК Пресс», которая посвящена относительно молодому языку программирования Clojure, объединяющим мощь Lisp с популярностью платформы JVM, позволяя очень быстро создавать сложные приложения.

Долгое время на русском языке было доступно небольшое количество материалов, например, статья «Clojure, или “Вы все еще используете Java? Тогда мы идем к вам!”», опубликованная в четвертом номере журнала «Практика функционального программирования» (<http://fprog.ru/>). Но постепенно новый язык начал набирать популярность и среди русско-язычных разработчиков. Одно из свидетельств этого – на вводный курс о Clojure, организованный Дмитрием Бушенко (<https://www.facebook.com/groups/clojure.course/>), записалось почти 90 человек.

Данная книга написана разработчиками, давно работающими с Clojure и создавшими большое количество популярных библиотек. Книга описывает как сам язык, так и основные приемы программирования на нем, очень часто непривычные для людей, использовавших только императивные или объектно-ориентированные языки. Но непривычность языка не должна вас отпугивать, вы сможете использовать новые приемы не только при программировании на Clojure, но и в «стандартных» языках.

Помимо описания самого языка достаточно большая часть книги посвящена практическим аспектам его применения – веб-программированию, работе с базами данных, взаимодействию с кодом написанным на Java и т.д.

Если у вас возникнут вопросы по разработке на Clojure, вы можете задать их в группе `ru_clojure` в LiveJournal (<http://ru-clojure.livejournal.com/>) или в списке рассылки `clojure-russian` (<https://groups.google.com/forum/?fromgroups#!forum/clojure-russian>). Существует также агрегатор русско-язычных блогов, посвященных Clojure (<http://feeds.feedburner.com/RussianClojurePlanet>), а если вы заинтересованы вопросами функционального программирования, то найдете много полезной информации в `Russian Lambda Planet` (<http://fprog.ru/planet/>).

Если вы захотите продолжить знакомство с языком, то можете получить больше информации в основном списке рассылки Clojure (<https://groups.google.com/forum/?fromgroups#!forum/clojure>), в постингах на `Planet Clojure` (<http://planet.clojure.in>) и других книгах, таких как «The Joy of Clojure».

Интересного чтения & happy hacking!

Алекс Отт



## Благодарности

При подготовке перевода данной книги к печати ее рукописи были переданы для обсуждения членам русскоязычного сообщества пользователей Clojure, принявшим самое деятельное участие в ее улучшении.

Особую благодарность издательство выражает Дмитрию Бушенко, Сергею Париеву и Федору Русаку. Спасибо вам, друзья! Вашу помощь переоценить невозможно!



## Предисловие

Clojure – динамический, строго типизированный язык программирования, основанный на виртуальной машине Java (Java Virtual Machine, JVM) и созданный уже пять лет тому назад. Он с восторгом был воспринят программистами, использующими самые разные языки и работающими в самых разных областях. Язык Clojure обладает весьма привлекательным набором возможностей и характеристик, приспособленных для решения современных задач программирования:

- ❑ поддержка функционального программирования, включая комплекс неизменяемых структур данных, по своей производительности приближающихся к обычным, изменяемым структурам;
- ❑ зрелая и эффективная среда выполнения, предоставляемая JVM;
- ❑ механизмы взаимодействий с JVM/Java, отвечающие самым широким архитектурным и эксплуатационным требованиям;
- ❑ комплекс надежных средств поддержки параллельного выполнения и семантики параллельного выполнения;
- ❑ будучи диалектом языка Lisp, предоставляет особенно гибкие и мощные средства метапрограммирования.

Язык Clojure представляет собой серьезную альтернативу для тех, кому постоянно приходится бороться с ограничениями обычных языков программирования и их окружений. Мы постараемся продемонстрировать это, показывая, насколько прозрачным выглядит взаимодействие Clojure с существующими технологиями, библиотеками и службами, используемыми программистами в повседневной работе. На протяжении всей книги мы будем знакомить вас с основами Clojure, опираясь в первую очередь на общественный опыт и знания, а не на (зачастую чужеродные) основные принципы информатики.

## Кому адресована эта книга?

Мы писали эту книгу с расчетом на две категории специалистов. Надеемся, что вы принадлежите к одной из них.

Язык Clojure не только не отстает, но и часто превосходит многие основные языки программирования по выразительности, лаконичности и

гибкости, позволяя при этом пользоваться преимуществами высокой производительности, богатства библиотек, наличия обширного сообщества и устойчивости JVM. Все это делает его естественным шагом вперед для разработчиков программ на языке Java (и даже разработчиков для JVM, использующих интерпретируемые и не особенно быстрые языки программирования, отличные от Java), кого не устраивает низкая производительность или кто не может отказаться от своих инвестиций в JVM. Не менее естественным шагом вперед Clojure является также для разработчиков на Ruby и Python, которые не готовы пожертвовать выразительностью языка, но желающие получить более надежную и эффективную платформу выполнения и огромный выбор высококачественных библиотек.

## **Для Java-разработчиков**

В мире живут и работают миллионы разработчиков на Java, но лишь немногие из них пишут программы для окружения, предъявляющего строгие требования, решая нетривиальные, часто узкоспециализированные задачи. Если вы относитесь к их числу, вероятно, вы постоянно находитесь в поисках более удобных инструментов и приемов, которые позволили бы повысить производительность труда, а также ценность вашего коллектива, организации или сообщества. Кроме того, вам, возможно, уже приходилось огорчаться, сталкиваясь с ограничениями в Java, отсутствующими в других языках, но при этом экосистема JVM не потеряла для вас своей привлекательности: трудно отказаться от зрелой среды выполнения, богатого выбора сторонних библиотек, поддержки поставщиками и огромного накопленного опыта, независимо от того, какими яркими не выглядели бы перспективы использования альтернативных языков.

В Clojure вы найдете долгожданное облегчение. Программы на этом языке выполняются под управлением JVM и обладают превосходной производительностью. Этот язык позволяет использовать любые имеющиеся библиотеки, инструменты и приложения. Он *проще* чем Java, лаконичнее и при этом намного выразительнее.

## **Для разработчиков на Ruby, Python и других языках**

Языки Ruby и Python далеко не новые, но в последние годы они приобрели особую популярность (можно даже сказать: «заняли господствующее положение»). Не трудно понять почему: оба являются выразительными, динамическими языками, поддерживаются бурно разрастающимися сообществами, обеспечивают высокую производительность труда во многих сферах.

Для вас язык Clojure также является естественным выбором. Как программист на Ruby или Python вы, вероятно, не желаете терять мощь этих языков, но вы можете испытывать потребность в более надежной платформе, обла-

дающей более высокой производительностью и богатым выбором библиотек. Язык Clojure, основанный на JVM, целиком и полностью отвечает этим требованиям – он полностью соответствует, а иногда и превосходит другие языки по своей выразительности и способности повышать производительность труда разработчика.

---

**Примечание.** Мы часто будем сравнивать Clojure с Java, Ruby и Python, чтобы помочь вам перенести свой опыт на Clojure. В таких сравнениях мы всегда будем подразумевать канонические реализации этих других языков:

- Ruby MRI (также называется, как CRuby);
  - CPython;
  - Java 6/7.
- 

## Как читать эту книгу

Работая над этой книгой, мы хотели как можно больше наполнить ее конкретными, подробными и практическими примерами, достаточно понятными, чтобы вы могли избежать ошибок. Мы не раз сталкивались с книгами, где от начала до конца рассматривался пример разработки единственной программы или приложения. Такой подход, на наш взгляд, разрушает целостность повествования и вынуждает авторов от главы к главе исследовать вымученный «практический» пример, который может не соответствовать кругу задач, решаемых читателем.

Учитывая это, мы разделили книгу на две основные части, и одну вступительную, где излагаются основы языка, интересными примерами две трети книги. За ними следует четвертая часть с большим количеством практических примеров из разных прикладных областей. Такое деление на части с совершенно разным содержанием позволяет квалифицировать эту книгу, как «книгу двойного назначения». (Автором этого термина, возможно, является Мартин Фаулер (Martin Fowler), употребивший его в статье <http://martinfowler.com/bliki/DuplexBook.html>.) В любом случае, мы предполагаем два подхода к чтению этой книги.

## *Начните с практического применения Clojure*

Часто лучший способ изучить язык заключается в том, чтобы применить его на практике. Если такой подход кажется вам более привлекательным, есть шанс, что вы найдете в книге пару практических примеров, демонстрирующих решение задач, с которыми вам приходится сталкиваться ежедневно, благодаря которым вы сможете провести параллели в решении некоторых категорий задач на используемом вами языке (или языках) и на языке Clojure. В этих примерах вы можете столкнуться с большим количеством не известных вам понятий и языковых конструкций. В подобных ситуациях, для по-

нимания новых концепций, используйте в качестве отправной точки контекст предметной области и учебный материал в первой части книги.

## ***Начните с последовательного изучения основ языка Clojure***

Иногда, чтобы по-настоящему разобраться в чем-то, необходимо досконально изучить внутреннее устройство, начиная с самых основ. Если вы предпочитаете такой подход, тогда лучше будет начать «переваривать» эту книгу с самого начала, с первой страницы в главе 1. Мы постарались последовательно и подробно объяснить все основополагающие принципы и конструкции языка Clojure, поэтому вам редко придется заглядывать вперед, чтобы понять концепции, встречающиеся раньше их описания. Выбирая подход последовательного освоения основ языка Clojure, не стесняйтесь забегать вперед и заглядывать в практическую часть книги, где вы найдете интересные примеры, схожие с задачами, решаемыми вами.

## **Кто мы?**

Мы – три разработчика программного обеспечения (ПО), разными путями пришедшие в Clojure и осознавшие его ценность. В процессе работы над книгой мы старались вложить в нее все наши представления о том, как и почему следует использовать Clojure, чтобы вы с успехом смогли применять его на практике.

## ***Чаз Эмерик***

Чаз Эмерик (Chas Emerick) стал постоянным членом сообщества Clojure в начале 2008. Занимался разработкой ядра языка, принимал участие в десятках проектов с открытым исходным кодом на языке Clojure, а также часто выступал и писал статьи о языке Clojure и разработке ПО в целом.

Чаз занимается сопровождением проекта Clojure Atlas (<http://clojureatlas.com>) визуализации языка Clojure и его стандартной библиотеки с целью обучения.

Является основателем Snowtide (<http://snowtide.com>), небольшой компании в Западном Массачусетсе, занимающейся производством ПО. Основной круг интересов Чаза лежит в области извлечения неструктурированных данных с уклоном в обработку документов PDF. Он пишет статьи и книги о Clojure занимается разработкой ПО и предпринимательством, а также имеет другие пристрастия (<http://cemerick.com>).

## ***Брайен Карпер***

Брайен Карпер (Brian Carper) – программист на Ruby, ставший приверженцем Clojure. Занимается программированием на Clojure, начиная с 2008,



и использует этот язык и дома, и на работе для всего подряд, от разработки веб-приложений до анализа данных в приложениях с графическим интерфейсом.

Брайен является автором приложения Gaka (<https://github.com/briancarper/gaka>), компилятора Clojure-to-CSS, и библиотеки объектно-реляционного отображения Oyako (<https://github.com/briancarper/oyako>). Пишет статьи о Clojure и на другие темы по адресу: <http://briancarper.net>.

## **Кристоф Гранд**

Кристоф Гранд (Christophe Grand) – давний поклонник функционального программирования, заплутавший на просторах Java, пока в начале 2008 не встретил Clojure и не влюбился в него с первого взгляда! Является автором Enlive (<http://github.com/cgrand/enlive>), библиотеки управления шаблонами, обеспечивающей возможность преобразования и извлечения HTML/XML; Parsley (<http://github.com/cgrand/parsley>), инкрементального парсер-генератора; и Moustache (<http://github.com/cgrand/moustache>), веб-фреймворка, включающего поддержку маршрутизации и промежуточных функций для Ring.

Как независимый консультант, занимается разработкой ПО и предлагает обучение языку Clojure. Он также пишет статьи о Clojure на сайте <http://clj-me.cgrand.net>.

## **Благодарности**

Как любой значительный труд, эта книга была бы невозможна без усилий десятков, если не сотен людей.

В числе первых хотелось бы назвать Ричарда Хикки (Rich Hickey), создателя языка Clojure. Всего за несколько коротких лет он спроектировал, реализовал и вывел в свет новый язык программирования, который для многих стал не просто еще одним инструментом, но и вдохнул новую жизнь в программирование. Кроме того, он лично многому научил нас, не только программированию, но также терпению, скромности и видению перспектив. Спасибо, Рич!

Дейв Файрам (Dave Faugam) и Майк Лукидис (Mike Loukides) оказали существенную помощь в выработке первоначальной концепции и организации книги. Конечно, вы сейчас не держали бы эту книгу в руках, если бы не наш редактор Джулия Стил (Julie Steele) и остальные сотрудники O'Reilly, занимавшиеся подготовкой книги к публикации.

Качество книги было бы намного ниже, если бы не наши технические редакторы, в число которых вошли Сэм Аарон (Sam Aaron), Энтони Батчелли (Antoni Batchelli), Том Фоулхабер (Tom Faulhaber), Крис Грангер (Chris Granger), Энтони Гримс (Anthony Grimes), Фил Хагельберг (Phil Hagelberg), Том Хикс (Tom Hicks), Алекс Миллер (Alex Miller), Вильям Морган (William Morgan), Лоурент Петит (Laurent Petit) и Дин Вамплер

(Dean Wampler). Мы также выражаем признательность всем, кто присылал свои отзывы и комментарии к черновикам книги на форумах издательства O'Reilly, по электронной почте, в твиттере и так далее.

Майкл Фогус (Michael Fogus) и Крис Хаузер (Chris Houser) многократно становились для нас источником вдохновения. Например, представив способы взаимодействия с REPL в своей книге о Clojure «The Joy of Clojure», которые мы бессовестно скопировали и повторили.

Если мы кого-то забыли упомянуть, примите нашу благодарность и извинения; закончив этот тяжелый труд, мы хотели бы оставаться честными и последовательными!

## ***И последнее, но не в последнюю очередь***

Сообщество Clojure оставалось моим вторым домом на протяжении ряда лет. Программисты на Clojure, отличающиеся приветливостью и положительной энергетикой, продолжают вдохновлять меня и служить примером для подражания. Например, многие завсегдааи канала #clojure на сайте Freenode IRC, ставшие моими друзьями, помогли мне освоить многое из того, в чем без их помощи разобраться было бы очень трудно.

Моим соавторам, Кристофу и Брайену: работать с вами было честью для меня. Я не представляю, как можно было бы закончить этот труд без вашего участия.

Моим родителям, Чарли (Charley) и Дарлин (Darleen): мое неутолимое любопытство к внутреннему устройству вещей, моя любовь к языку и риторике и мои интересы в бизнесе – все это было заложено вами много лет тому назад. Без вашего влияния я определенно не смог бы найти свой путь в жизни, основать свою компанию или написать эту книгу – всего этого я достиг, благодаря настойчивости, воспитанной вами.

Наконец, моей супруге Крисси (Krissy): жертвы, которые ты принесла, чтобы позволить мне следовать своим амбициям, безмерны. Боюсь, что никогда не смогу в полной мере отблагодарить тебя за это. Поэтому я скажу просто: я люблю тебя.

– Чаз Эмерик (Chas Emerick),  
февраль 2012

Всем членам сообщества, помогавшим в создании Clojure: спасибо за ваш беспримерный труд, за то что сделали мою профессиональную и личную жизнь намного более приятной и за то, что открыли мне глаза на новые перспективы.

Моим соавторам, Кристофу и Чазу: никогда прежде мне не доводилось работать со столь умными людьми. Для меня это было честью и привилегией.

Моей жене Николь (Nicole): прости, что не давал тебе заснуть по ночам треском клавиатуры.

– Брайен Карпер (Brian Carper),  
февраль 2012

Ричарду Хикки (Rich Hickey), создавшему Clojure и сплотившему такое дружное сообщество.

Сообществу, приведшему меня к более высоким стандартам.

Моим соавторам, Брайену и Чазу: для было большой честью работать с вами.

Моему профессору Даниелю Гоффинье (Daniel Goffinet), чей острый ум радикально изменил мои взгляды на программирование и информатику в целом — этим я в значительной мере обязан ему, как никому другому.

Моим родителям: за вашу любовь и первый 8-разрядный компьютер, за которым я проводил слишком много времени, вызывая у вас беспокойство.

Моей супруге Эмили (Emilie) и моему сыну Гэлу (Gaël): спасибо вам за поддержку, которую вы оказывали, пока я работал над этой книгой.

— Кристоф Гранд (Christophe Grand),  
февраль 2012

## Типографские соглашения

В этой книге приняты следующие типографские соглашения:

### *Курсив*

Используется для обозначения новых терминов, имен файлов и расширений имен файлов.

### Моноширинный шрифт

Применяется для оформления листингов программ и программных элементов внутри обычного текста, таких как имена переменных и функций, типов данных, переменных окружения, инструкций и ключевых слов.

; строки в листингах, начинающиеся с точки с запятой

Обозначают вывод (в стандартные потоки вывода и ошибок), полученный в результате выполнения программного кода в интерактивной оболочке REPL.

;= строки в листингах, начинающиеся с точки с запятой и знака равенства

Обозначают результат/возвращаемое значение, полученный в ходе вычислений в интерактивной оболочке REPL.

### Моноширинный жирный

Обозначает команды или другой текст, который должен вводиться пользователем.

### Моноширинный курсив

Обозначает текст, который должен замещаться фактическими значениями, вводимыми пользователем или определяемыми из контекста.

---

**Примечание.** Так обозначаются советы, предложения и примечания общего характера.

---

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

[e-Univers.ru](http://e-Univers.ru)