

Оглавление

Предисловие	15
Благодарности	16
О книге	17
Структура	17
Правила оформления и загрузка примеров кода	18
Об авторах	19
Алекс Янг	19
Брэдли Мек	19
Иллюстрация на обложке	20
От издательства	20
Часть I. Знакомство с Node	21
Глава 1. Знакомство с Node.js	22
1.1. Типичное веб-приложение Node	22
1.1.1. Неблокирующий ввод/вывод	23
1.1.2. Цикл событий	25
1.2. ES2015, Node и V8	26
1.2.1. Node и V8	29
1.2.2. Работа с функциональными группами	30
1.2.3. График выпуска версий Node	31
1.3. Установка Node	31
1.4. Встроенные средства Node	32
1.4.1. прм	33
1.4.2. Базовые модули	34
1.4.3. Отладчик	35
1.5. Три основных типа программ Node	36
1.5.1. Веб-приложения	37
1.5.2. Средства командной строки и демоны	38
1.5.3. Настольные приложения	39
1.5.4. Приложения, хорошо подходящие для Node	39
1.6. Заключение	40

Глава 2. Основы программирования Node	41
2.1. Структурирование и повторное использование функциональности Node	41
2.2. Создание нового проекта Node	44
2.2.1. Создание модулей	44
2.3. Настройка создания модуля с использованием module.exports	47
2.4. Повторное использование модулей с папкой node_modules	48
2.5. Потенциальные проблемы	48
2.6. Средства асинхронного программирования	51
2.7. Обработка одноразовых событий в обратных вызовах	52
2.8. Обработка повторяющихся событий с генераторами событий	56
2.8.1. Пример генератора событий	56
2.8.2. Реакция на событие, которое должно происходить только один раз	58
2.8.3. Создание генераторов событий: публикация/подписка	58
2.8.4. Доработка генератора событий: отслеживание содержимого файлов	62
2.9. Проблемы с асинхронной разработкой	64
2.10. Упорядочение асинхронной логики	65
2.11. Когда применяется последовательный поток выполнения	67
2.12. Реализация последовательного потока выполнения	68
2.13. Реализация параллельного потока выполнения	71
2.14. Средства, разработанные в сообществе	73
2.15. Заключение	75
Глава 3. Что представляет собой веб-приложение Node?	76
3.1. Структура веб-приложения Node	77
3.1.1. Создание нового веб-приложения	77
3.1.2. Сравнение с другими платформами	79
3.1.3. Что дальше?	79
3.2. Построение REST-совместимой веб-службы	80
3.3. Добавление базы данных	83
3.3.1. Проектирование собственного API модели	84
3.3.2. Преобразование статей в удобочитаемую форму и их сохранение для чтения в будущем	87
3.4. Добавление пользовательского интерфейса	88
3.4.1. Поддержка разных форматов	89
3.4.2. Визуализация шаблонов	89
3.4.3. Использование прм для зависимостей на стороне клиента	90
3.5. Заключение	92

Часть II. Веб-разработка с использованием Node	93
Глава 4. Системы построения фронтэнда	94
4.1. Фронтэнд-разработка с использованием Node	94
4.2. Использование прт для запуска сценариев	95
4.2.1. Создание специализированных сценариев прт..	97
4.2.2. Настройка средств построения фронтэнда	98
4.3. Автоматизация с использованием Gulp	98
4.3.1. Добавление Gulp в проект	99
4.3.2. Создание и выполнение задач Gulp	100
4.3.3. Отслеживание изменений	102
4.3.4. Использование отдельных файлов в больших проектах..	102
4.4. Построение веб-приложений с использованием webpack..	104
4.4.1. Пакеты и плагины..	104
4.4.2. Настройка и запуск webpack	105
4.4.3. Использование сервера для разработки webpack	106
4.4.4. Загрузка модулей и активов CommonJS	107
4.5. Заключение..	109
Глава 5. Фреймворки на стороне сервера	110
5.1. Персонажи	110
5.1.1. Фил: штатный разработчик	111
5.1.2. Надин: разработчик открытого кода	111
5.1.3. Элис: разработчик продукта	112
5.2. Что такое фреймворк?	112
5.3. Кoa	113
5.3.1. Настройка	115
5.3.2. Определение маршрутов	116
5.3.3. REST API..	116
5.3.4. Сильные стороны	117
5.3.5. Слабые стороны..	117
5.4. Kraken	117
5.4.1. Настройка	118
5.4.2. Определение маршрутов	118
5.4.3. REST API..	119
5.4.4. Сильные стороны	119
5.4.5. Слабые стороны..	120
5.5. hapi	120
5.5.1. Настройка	121
5.5.2. Определение маршрутов	121
5.5.3. Плагины	122

5.5.4. REST API..	123
5.5.5. Сильные стороны	124
5.5.6. Слабые стороны..	124
5.6. Sails.js	124
5.6.1. Настройка	125
5.6.2. Определение маршрутов	126
5.6.3. REST API..	126
5.6.4. Сильные стороны	127
5.6.5. Слабые стороны..	127
5.7. DerbyJS	127
5.7.1. Настройка	128
5.7.2. Определение маршрутов	129
5.7.3. REST API..	130
5.7.4. Сильные стороны	130
5.7.5. Слабые стороны..	130
5.8. Flatiron.js	131
5.8.1. Настройка	131
5.8.2. Определение маршрутов	132
5.8.3. REST API..	133
5.8.4. Сильные стороны	133
5.8.5. Слабые стороны..	134
5.9. LoopBack	134
5.9.1. Настройка	135
5.9.2. Определение маршрутов	137
5.9.3. REST API..	137
5.9.4. Сильные стороны	138
5.9.5. Слабые стороны..	138
5.10. Сравнение	138
5.10.1. Серверы HTTP и маршруты	140
5.11. Написание модульного кода	140
5.12. Выбор персонажей	141
5.13. Заключение	141
Глава 6. Connect и Express	142
6.1. Connect	142
6.1.1. Настройка приложения Connect	143
6.1.2. Как работают промежуточные компоненты Connect	143
6.1.3. Объединение промежуточных компонентов	144
6.1.4. Упорядочение компонентов	145
6.1.5. Создание настраиваемых промежуточных компонентов	146

6.1.6. Использование промежуточных компонентов для обработки ошибок	148
6.2. Express	151
6.2.1. Генерирование заготовки приложения	152
6.2.2. Настройка конфигурации Express и приложения	157
6.2.3. Визуализация представлений	159
6.2.4. Знакомство с маршрутизацией в Express	165
6.2.5. Аутентификация пользователей	173
6.2.6. Регистрация новых пользователей	179
6.2.7. Вход для зарегистрированных пользователей	185
6.2.8. Промежуточный компонент для загрузки пользовательских данных	189
6.2.9. Создание открытого REST API	191
6.2.10. Согласование контента	197
6.3. Заключение	200
Глава 7. Шаблонизация веб-приложений	201
7.1. Поддержка чистоты кода путем шаблонизации	201
7.1.1. Шаблонизация в действии	203
7.1.2. Визуализация HTML без шаблона	205
7.2. Шаблонизация с EJS	207
7.2.1. Создание шаблона	207
7.2.2. Интеграция шаблонов EJS в приложение	209
7.2.3. Использование EJS в клиентских приложениях	210
7.3. Использование языка Mustache с шаблонизатором Hogan	211
7.3.1. Создание шаблона	212
7.3.2. Теги Mustache	212
7.3.3. Тонкая настройка Hogan	215
7.4. Шаблоны Pug	215
7.4.1. Основные сведения о Pug	217
7.4.2. Программная логика в шаблонах Pug	219
7.4.3. Организация шаблонов Pug	223
7.5. Заключение	227
Глава 8. Хранение данных в приложениях	228
8.1. Реляционные базы данных	228
8.2. PostgreSQL	228
8.2.1. Установка и настройка	229
8.2.2. Создание базы данных	229
8.2.3. Подключение к Postgres из Node	230
8.2.4. Определение таблиц	230

8.2.5. Вставка данных	231
8.2.6. Обновление данных	231
8.2.7. Запросы на выборку данных	232
8.3. Knex	232
8.3.1. jQuerу для баз данных	233
8.3.2. Подключение и выполнение запросов в Knex	234
8.3.3. Переход на другую базу данных	236
8.3.4. Остерегайтесь ненадежных абстракций	236
8.4. MySQL и PostgreSQL	237
8.5. Гарантии ACID	238
8.5.1. Атомарность	238
8.5.2. Согласованность	238
8.5.3. Изоляция	239
8.5.4. Устойчивость	239
8.6. NoSQL	239
8.7. Распределенные базы данных	240
8.8. MongoDB	241
8.8.1. Установка и настройка	242
8.8.2. Подключение к MongoDB	242
8.8.3. Вставка документов	243
8.8.4. Получение информации	243
8.8.5. Идентификаторы MongoDB	245
8.8.6. Реплицированные наборы	247
8.8.7. Уровень записи	248
8.9. Хранилища «ключ-значение»	250
8.10. Redis	251
8.10.1. Установка и настройка	252
8.10.2. Выполнение инициализации	252
8.10.3. Работа с парами «ключ-значение»	253
8.10.4. Работа с ключами	254
8.10.5. Кодирование и типы данных	254
8.10.6. Работа с хешами	256
8.10.7. Работа со списками	257
8.10.8. Работа со множествами	258
8.10.9. Реализация паттерна «публикация/подписка» на базе каналов	259
8.10.10. Повышение быстродействия Redis	260
8.11. Встроенные базы данных	260
8.12. LevelDB	261
8.12.1. LevelUP и LevelDOWN	262
8.12.2. Установка	262

8.12.3. Обзор API	263
8.12.4. Инициализация	263
8.12.5. Кодирование ключей и значений	264
8.12.6. Чтение и запись пар «ключ-значение»	264
8.12.7. Заменяемые подсистемы базы данных	265
8.12.8. Модульная база данных	267
8.13. Затратные операции сериализации и десериализации	268
8.14. Хранение данных в браузере	269
8.14.1. Веб-хранилище: localStorage и sessionStorage	269
8.14.2. Чтение и запись значений	270
8.14.3. localForage	273
8.14.4. Чтение и запись	273
8.15. Виртуальное хранение	274
8.15.1. S3	275
8.16. Какую базу данных выбрать?	276
8.17. Заключение	276
Глава 9. Тестирование приложений Node	278
9.1. Модульное тестирование	279
9.1.1. Модуль assert	280
9.1.2. Mocha	284
9.1.3. Vows	289
9.1.4. Chai	292
9.1.5. Библиотека should.js	293
9.1.6. Шпионы и заглушки в Sinon.JS	296
9.2. Функциональное тестирование	298
9.2.1. Selenium	299
9.3. Ошибки при прохождении тестов	302
9.3.1. Получение более подробных журналов	303
9.3.2. Получение расширенной трассировки стека	305
9.4. Заключение	307
Глава 10. Развёртывание и обеспечение доступности приложений Node ..	308
10.1. Хостинг Node-приложений	308
10.1.1. Платформа как сервис	309
10.1.2. Серверы	311
10.1.3. Контейнеры	312
10.2. Основы развертывания	314
10.2.1. Развёртывание из репозитория Git	314
10.2.2. Поддержание работы Node-приложения	315
10.3. Максимизация времени доступности и производительности приложений	317

10.3.1. Поддержание доступности приложения с Upstart	318
10.3.2. Кластерный API	320
10.3.3. Хостинг статических файлов и представительство.....	322
10.4. Заключение	324
Часть III. За пределами веб-разработки	325
Глава 11. Написание приложений командной строки.	326
11.1. Соглашения и философия.....	326
11.2. Знакомство с parse-json	328
11.3. Аргументы командной строки	328
11.3.1. Разбор аргументов командной строки.....	328
11.3.2. Проверка аргументов	329
11.3.3. Передача stdin в виде файла	330
11.4. Использование программ командной строки с npm	331
11.5. Связывание сценариев с каналами	332
11.5.1. Передача данных parse-json	332
11.5.2. Ошибки и коды завершения	333
11.5.3. Использование каналов в Node	335
11.5.4. Каналы и последовательность выполнения команд	336
11.6. Интерпретация реальных сценариев	337
11.7. Заключение	338
Глава 12. Разработка настольных приложений с использованием Electron	339
12.1. Знакомство с Electron	339
12.1.1. Технологический стек Electron	340
12.1.2. Проектирование интерфейса	341
12.2. Создание приложения Electron	342
12.3. Построение полнофункционального настольного приложения	344
12.3.1. Исходная настройка React и Babel	345
12.3.2. Установка зависимостей	345
12.3.3. Настройка webpack	346
12.4. Приложение React	348
12.4.1. Определение компонента Request	349
12.4.2. Определение компонента Response	352
12.4.3. Взаимодействие между компонентами React	354
12.5. Построение и распространение	355
12.5.1. Построение приложений с использованием Electron Packager ..	356
12.5.2. Упаковка	357
12.6. Заключение	358

Приложения	359
Приложение А. Установка Node	360
A.1. Установка Node с использованием программы установки	360
A.1.1. Программа установки для macOS	360
A.1.2. Программа установки для Windows	362
A.2. Другие способы установки Node	363
A.2.1. Установка Node из исходного кода	363
A.2.2. Установка Node из менеджера пакетов	363
Приложение Б. Автоматизированное извлечение веб-данных	365
B.1. Извлечение веб-данных	365
B.1.1. Применение извлечения веб-данных	366
B.1.2. Необходимые инструменты	367
B.2. Простейшее извлечение веб-данных с использованием cheerio	368
B.3. Обработка динамического контента с jsdom	371
B.4. Обработка «сырых» данных	374
B.5. Заключение	377
Приложение В. Официально поддерживаемые промежуточные компоненты	378
B.1. Разбор cookie, тел запросов и строк информационных запросов	378
B.1.1. cookie-parser: разбор HTTP-cookie	379
B.1.2. Разбор строк запросов	383
B.1.3. body-parser: разбор тел запросов	384
B.1.4. Сжатие ответов	391
B.2. Реализация базовых функций веб-приложения	393
B.2.1. morgan: ведение журнала запросов	393
B.2.2. serve-favicon: значки адресной строки и закладки	397
B.2.3. methodOverride — имитация методов HTTP	398
B.2.4. vhost: виртуальный хостинг	401
B.2.5. express-session: управление сеансами	402
B.3. Безопасность веб-приложений	407
B.3.1. basic-auth: базовая HTTP-аутентификация	408
B.3.2. csurf: защита от атак CSRF	410
B.3.3. errorhandler: — обработка ошибок при разработке	412
B.4. Предоставление статических файлов	414
B.4.1. serve-static — автоматическое предоставление статических файлов браузеру	414
B.4.2. serve-index: генерирование списков содержимого каталогов	416
Глоссарий	418