

# Оглавление

<b>Предисловие</b> .....	<b>15</b>
<b>Благодарности</b> .....	<b>16</b>
<b>О книге</b> .....	<b>17</b>
Структура .....	17
Правила оформления и загрузка примеров кода .....	18
<b>Об авторах</b> .....	<b>19</b>
Алекс Янг .....	19
Брэдли Мек .....	19
<b>Иллюстрация на обложке</b> .....	<b>20</b>
От издательства .....	20
<b>Часть I. Знакомство с Node</b> .....	<b>21</b>
<b>Глава 1. Знакомство с Node.js</b> .....	<b>22</b>
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. npm .....	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

<b>Глава 2. Основы программирования Node</b> .....	<b>41</b>
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 однократных 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
<b>Глава 3. Что представляет собой веб-приложение Node?</b> .....	<b>76</b>
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 собственной 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 прм для запуска сценариев . . . . . 95
  - 4.2.1. Создание специализированных сценариев прм . . . . . 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. Коа . . . . . 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







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 prn.. . . . . .	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

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