

# Оглавление

<b>Предисловие</b> .....	<b>10</b>
<b>Введение</b> .....	<b>12</b>
Задача .....	13
О книге .....	15
Типографские соглашения .....	15
Использование программного кода примеров .....	16
От издательства .....	17
<b>Глава 1. Асинхронность: сейчас и потом</b> .....	<b>18</b>
Блочное строение программы .....	19
Асинхронный вывод в консоль .....	22
Цикл событий .....	23
Параллельные потоки .....	26
Выполнение до завершения .....	30
Параллельное выполнение .....	33
Отсутствие взаимодействий .....	36
Взаимодействия .....	36
Кооперация .....	42
Задания .....	45
Упорядочение команд .....	46
Итоги .....	50
<b>Глава 2. Обратные вызовы</b> .....	<b>52</b>
Продолжения .....	53
Последовательное мышление .....	55

Работа и планирование . . . . .	56
Вложенные/сцепленные обратные вызовы . . . . .	59
Проблемы доверия . . . . .	65
История о пяти обратных вызовах. . . . .	66
Не только в чужом коде. . . . .	69
Попытки спасти обратные вызовы. . . . .	71
Итоги . . . . .	76
<b>Глава 3. Обещания . . . . .</b>	<b>78</b>
Что такое обещание? . . . . .	79
Будущее значение . . . . .	80
Событие завершения . . . . .	86
События обещаний . . . . .	90
Утиная типизация с методом <code>then()(thenable)</code> . . . . .	93
Доверие <code>Promise</code> . . . . .	96
Слишком ранний обратный вызов . . . . .	97
Слишком поздний обратный вызов. . . . .	97
Обратный вызов вообще не вызывается. . . . .	100
Слишком малое или слишком большое количество вызовов . . . . .	101
Отсутствие параметров/переменных среды . . . . .	102
Поглощение ошибок/исключений. . . . .	102
Обещания, заслуживающие доверия? . . . . .	104
Формирование доверия . . . . .	108
Сцепление . . . . .	109
Терминология: разрешение, выполнение и отказ . . . . .	118
Обработка ошибок . . . . .	121
Бездна отчаяния . . . . .	125
Обработка неперехваченных ошибок . . . . .	126
Бездна успеха . . . . .	128
Паттерны обещаний . . . . .	131
<code>Promise.all([ .. ])</code> . . . . .	131
<code>Promise.race([ .. ])</code> . . . . .	133
Вариации на тему <code>all([ .. ])</code> и <code>race([ .. ])</code> . . . . .	137

Параллельно выполняемые итерации . . . . .	139
Снова о Promise API . . . . .	140
Конструктор new Promise(..) . . . . .	141
Promise.resolve(..) и Promise.reject(..) . . . . .	141
then(..) и catch(..) . . . . .	142
Promise.all([ .. ]) и Promise.race([ .. ]) . . . . .	143
Ограничения обещаний. . . . .	145
Последовательность обработки ошибок . . . . .	145
Единственное значение . . . . .	147
Инерция . . . . .	152
Неотменяемость обещаний . . . . .	157
Эффективность обещаний . . . . .	159
Итоги . . . . .	161
<b>Глава 4. Генераторы. . . . .</b>	<b>162</b>
Нарушение принципа выполнения до завершения. . . . .	162
Ввод и вывод. . . . .	166
Передача сообщений при итерациях . . . . .	167
Множественные итераторы . . . . .	171
Генерирование значений . . . . .	176
Производители и итераторы . . . . .	176
Итерируемые объекты . . . . .	180
Итераторы генераторов . . . . .	182
Асинхронный перебор итераторов. . . . .	186
Синхронная обработка ошибок. . . . .	190
Генераторы + обещания . . . . .	192
Выполнение генератора с поддержкой обещаний. . . . .	195
Параллелизм обещаний в генераторах. . . . .	199
Делегирование . . . . .	204
Почему делегирование? . . . . .	207
Делегирование сообщений. . . . .	208
Делегирование асинхронности. . . . .	213
Делегирование рекурсии . . . . .	214
Параллельное выполнение генераторов . . . . .	216

Преобразователи . . . . .	222
s/promise/thunk/. . . . .	227
Генераторы до ES6 . . . . .	231
Ручное преобразование . . . . .	231
Автоматическая транспиляция . . . . .	237
Итоги . . . . .	239
<b>Глава 5. Быстродействие программ . . . . .</b>	<b>241</b>
Веб-работники . . . . .	242
Рабочая среда . . . . .	246
Передача данных. . . . .	247
Общие работники . . . . .	248
Полифилы для веб-работников . . . . .	250
SIMD. . . . .	251
asm.js . . . . .	253
Как оптимизировать с asm.js . . . . .	254
Модули asm.js . . . . .	255
Итоги . . . . .	258
<b>Глава 6. Хронометраж и настройка . . . . .</b>	<b>260</b>
Хронометраж . . . . .	261
Повторение . . . . .	262
Benchmark.js . . . . .	264
Все зависит от контекста. . . . .	267
Оптимизации движка . . . . .	268
jsPerf.com . . . . .	271
Проверка на здравый смысл . . . . .	272
Написание хороших тестов . . . . .	276
Микробыстродействие. . . . .	277
Различия между движками . . . . .	282
Общая картина . . . . .	285
Оптимизация хвостовых вызовов (TCO). . . . .	288
Итоги . . . . .	291

**Приложение А. Библиотека `asynquence` . . . . . 292**

Последовательности и архитектура, основанная на абстракциях . . . . .	293
<code>asynquence</code> API . . . . .	297
Шаги . . . . .	297
Ошибки . . . . .	300
Параллельные шаги . . . . .	303
Ветвление последовательностей . . . . .	311
Объединение последовательностей . . . . .	311
Значение и последовательности ошибки . . . . .	313
Обещания и обратные вызовы . . . . .	314
Итерируемые последовательности . . . . .	316
Выполнение генераторов . . . . .	318
Обертки для генераторов . . . . .	319
Итоги . . . . .	319

**Приложение Б. Расширенные асинхронные паттерны. . . . . 321**

Итерируемые последовательности . . . . .	321
Расширение итерируемых последовательностей . . . . .	325
Реакция на события . . . . .	330
Наблюдаемые объекты в ES7 . . . . .	332
Реактивные последовательности . . . . .	334
Генераторные сопрограммы (Generator Coroutine) . . . . .	338
Конечные автоматы . . . . .	340
Взаимодействующие последовательные процессы . . . . .	343
Передача сообщений . . . . .	343
Эмуляция CSP в <code>asynquence</code> . . . . .	346
Итоги . . . . .	349

**Об авторе . . . . . 350**