

УДК 004.438ECMAScript 6
ББК 32.973.2
П70

П70 Нараян Прасти

Введение в ECMAScript 6. / пер. с англ. Рагимов Р. Н.– М.: ДМК Пресс, 2016. – 176 с.: ил.

ISBN 978-5-97060-392-5

Данная книга содержит пошаговые инструкции по использованию новых возможностей ECMAScript 6 вместо устаревших трюков и приемов программирования на JavaScript.

Книга начинается с знакомства со всеми встроенными объектами ES6 и описания создания итераторов ES6. Затем она расскажет, как писать асинхронный код с помощью ES6 в обычном стиле синхронного кода. Далее описывается использование программного интерфейса рефлексии Reflect API для исследования и изменения свойств объектов. Затем рассматривается создание прокси-объектов и их применение для перехвата и изменения поведения операций с объектами. Наконец, демонстрируются устаревшие методы модульного программирования, такие как IIFE, CommonJS, AMD и UMD, и сравниваются с модулями ES6, способными значительно увеличить производительность веб-сайтов.

Издание предназначено для программистов на JavaScript, обладающих базовыми навыками разработки, и желающих освоить новейшие возможности ECMAScript 6 для совершенствования своих программ, выполняемых на стороне клиента.

Original English language edition published by Published by Packt Publishing Ltd., Livery Place, 35 Livery Street, Birmingham B3 2PB, UK. Copyright © 2015 Packt Publishing. Russian-language edition copyright © 2016 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-78588-444-3 (англ.)
 ISBN 978-5-97060-392-5 (рус.)

Copyright © 2015 Packt Publishing
 © Оформление, перевод на русский язык,
 ДМК Пресс, 2016



ОГЛАВЛЕНИЕ

| | |
|---|-----------|
| Предисловие | 10 |
| Об авторе | 12 |
| О технических рецензентах | 13 |
| Введение | 16 |
| О чем рассказывается в этой книге | 16 |
| Что понадобится при чтении этой книги | 17 |
| Совместимость с ECMAScript 6 | 18 |
| Запуск ECMAScript 6 в несовместимых реализациях | 18 |
| Кому адресована эта книга | 19 |
| Соглашения | 19 |
| Отзывы и пожелания | 20 |
| Скачивание исходного кода примеров | 21 |
| Нарушение авторских прав | 21 |
| Глава 1. Игры с синтаксисом..... | 22 |
| Ключевое слово let | 22 |
| Объявление переменных с областью видимости в пределах функции... 23 | |
| Объявление переменных с областью видимости в пределах блока | 24 |
| Повторное объявление переменных | 25 |
| Ключевое слово const | 27 |
| Область видимости констант | 27 |
| Ссылки на объекты при помощи констант | 28 |
| Значения параметров по умолчанию | 29 |
| Оператор расширения | 30 |
| Другие применения оператора расширения | 31 |
| Расширение нескольких массивов | 32 |
| Дополнительные параметры | 32 |
| Деструктивное присваивание | 33 |
| Деструктивное присваивание массивов..... | 34 |
| Деструктивное присваивание объектов | 37 |
| Стрелочные функции..... | 39 |

| | |
|---|-----------|
| Расширенные литералы объектов | 41 |
| Определение свойств | 41 |
| Определение методов | 41 |
| Вычисляемые имена свойств | 42 |
| Итоги | 42 |
| Глава 2. Знакомство с библиотекой | 43 |
| Работа с числами | 43 |
| Двоичное представление | 44 |
| Восьмеричное представление | 44 |
| Метод <code>Number.isInteger(number)</code> | 45 |
| Метод <code>Number.isNaN(value)</code> | 45 |
| Метод <code>Number.isFinite(number)</code> | 46 |
| Метод <code>Number.isSafeInteger(number)</code> | 47 |
| Свойство <code>Number.EPSILON</code> | 48 |
| Объект <code>Math</code> | 49 |
| Тригонометрические операции | 49 |
| Алгебраические операции | 49 |
| Прочие методы | 50 |
| Работа со строками | 52 |
| Управляющая последовательность для больших кодовых пунктов | 53 |
| Метод <code>codePointAt(index)</code> | 53 |
| Метод <code>String.fromCodePoint(number1, ..., number 2)</code> | 53 |
| Метод <code>repeat(count)</code> | 54 |
| Метод <code>includes(string, index)</code> | 54 |
| Метод <code>startsWith(string, index)</code> | 54 |
| Функция <code>endsWith(string, index)</code> | 55 |
| Нормализация | 55 |
| Шаблонные строки | 57 |
| Выражения | 57 |
| Массивы | 60 |
| Метод <code>Array.from(iterable, mapFunc, this)</code> | 60 |
| Метод <code>Array.of(values...)</code> | 61 |
| Метод <code>fill(value, startIndex, endIndex)</code> | 61 |
| Метод <code>find(testingFunc, this)</code> | 62 |
| Метод <code>findIndex(testingFunc, this)</code> | 63 |
| Метод <code>copyWithin(targetIndex, startIndex, endIndex)</code> | 63 |
| Методы <code>entries()</code> , <code>keys()</code> и <code>values()</code> | 64 |
| Коллекции | 64 |
| Буферные массивы | 65 |
| Типизированные массивы | 67 |
| Объект <code>Set</code> | 68 |
| Объект <code>WeakSet</code> | 69 |
| Объект <code>Map</code> | 69 |
| Объект <code>WeakMap</code> | 70 |
| Объект <code>Object</code> | 71 |

| | |
|---|-----------|
| Свойство <code>__proto__</code> | 71 |
| Метод <code>Object.is(value1, value2)</code> | 72 |
| Метод <code>Object.setPrototypeOf(object, prototype)</code> | 72 |
| Метод <code>Object.assign(targetObj, sourceObjs...)</code> | 72 |
| Итоги | 73 |
| Глава 3. Использование итераторов..... | 75 |
| Символы в спецификации ES6 | 75 |
| Оператор <code>typeof</code> | 76 |
| Оператор <code>new</code> | 76 |
| Использование символов как ключей свойств..... | 77 |
| Метод <code>Object.getOwnPropertySymbols()</code> | 77 |
| Метод <code>Symbol.for(string)</code> | 78 |
| Встроенные символы | 79 |
| Протоколы итераций | 79 |
| Протокол итератора | 79 |
| Итерационный протокол | 80 |
| Генераторы | 81 |
| Метод <code>return(value)</code> | 83 |
| Метод <code>throw(exception)</code> | 84 |
| Ключевое слово <code>yield*</code> | 84 |
| Цикл <code>for...of</code> | 85 |
| Оптимизация хвостового вызова | 86 |
| Преобразование неконцевых вызовов в концевые вызовы..... | 87 |
| Итоги | 88 |
| Глава 4. Асинхронное программирование | 89 |
| Модель выполнения JavaScript | 89 |
| Разработка асинхронного кода | 90 |
| Асинхронный код, основанный на событиях | 91 |
| Асинхронный код, основанный на обратных вызовах | 94 |
| Объекты Promise в помощь..... | 95 |
| Конструктор Promise | 96 |
| Результат асинхронной операции | 97 |
| Метод <code>then(onFulfilled, onRejected)</code> | 98 |
| Метод <code>catch(onRejected)</code> | 104 |
| Метод <code>Promise.resolve(value)</code> | 106 |
| Метод <code>Promise.reject(value)</code> | 107 |
| Метод <code>Promise.all(iterable)</code> | 107 |
| Метод <code>Promise.race(iterable)</code> | 108 |
| Программные интерфейсы JavaScript, основанные на объектах Promise | 109 |
| Программный интерфейс состояния батареи | 109 |
| Программный интерфейс веб-криптографии | 110 |

| | |
|--|------------|
| Итоги | 111 |
| Глава 5. Реализация Reflect API..... | 112 |
| Объект Reflect | 112 |
| Метод Reflect.apply(function, this, args) | 113 |
| Метод Reflect.construct(constructor, args, prototype) | 113 |
| Метод Reflect.defineProperty(object, property, descriptor) | 114 |
| Метод Reflect.deleteProperty(object, property) | 117 |
| Метод Reflect.enumerate(object) | 118 |
| Метод Reflect.get(object, property, this) | 118 |
| Метод Reflect.set(object, property, value, this)..... | 119 |
| Метод Reflect.getOwnPropertyDescriptor(object, property) | 119 |
| Метод Reflect.getPrototypeOf(object) | 120 |
| Метод Reflect.setPrototypeOf(object, prototype) | 120 |
| Метод Reflect.has(object, property) | 121 |
| Метод Reflect.isExtensible(object) | 121 |
| Метод Reflect.preventExtensions(object) | 121 |
| Метод Reflect.ownKeys(object) | 122 |
| Итоги | 122 |
| Глава 6. Использование прокси-объектов | 123 |
| Основы прокси-объектов | 123 |
| Терминология | 124 |
| Программный интерфейс Proxy API | 124 |
| Ловушки | 125 |
| Метод Proxy.revocable(target, handler) | 137 |
| Возможный сценарий использования | 138 |
| Использование прокси | 138 |
| Итоги | 138 |
| Глава 7. Прогулка по классам | 139 |
| Понимание объектно-ориентированной модели JavaScript | 139 |
| Типы данных JavaScript | 140 |
| Создание объектов | 140 |
| Понятие наследования..... | 141 |
| Конструкторы элементарных типов данных..... | 145 |
| Использование классов | 146 |
| Определение классов | 147 |
| Методы прототипа | 149 |
| Статические методы | 152 |
| Реализация наследования классов | 152 |
| Вычисляемые имена методов | 154 |
| Атрибуты свойств..... | 155 |
| Классы не всплывают! | 155 |
| Переопределение результата метода constructor | 156 |

| | |
|--|------------|
| Статическое свойство со средствами доступа <code>Symbol.species</code> | 156 |
| Неявный параметр <code>new.target</code> | 158 |
| Использование <code>super</code> в литералах объектов | 159 |
| Итоги | 159 |
| Глава 8. Модульное программирование | 160 |
| Введение в модули JavaScript | 160 |
| Реализация модулей по-старому | 161 |
| Немедленно вызываемые функции-выражения | 161 |
| Асинхронное определение модулей..... | 162 |
| CommonJS | 164 |
| Универсальное определение модуля | 164 |
| Реализация модулей – новый подход | 165 |
| Создание модулей ES6 | 166 |
| Импорт модулей в ES6 | 167 |
| Загрузчик модулей | 169 |
| Использование модулей в браузерах..... | 169 |
| Использование модулей в функции <code>eval()</code> | 170 |
| Экспорт по умолчанию или экспорт по именам | 170 |
| Пример | 170 |
| Итоги | 172 |
| Предметный указатель | 173 |