

НИКОЛАС ЗАКАС

ESMAScript 6

для разработчиков



Санкт-Петербург · Москва · Екатеринбург · Воронеж
Нижний Новгород · Ростов-на-Дону
Самара · Минск

2017

Закас, Н. ECMAScript 6 для разработчиков : [перевод с английского] / Николас Закас. — Санкт-Петербург [и др.] : Питер, 2017. — 349 с. : ил. — (Библиотека программиста).

УДК 004.438ECMAScript

ББК 32

Аб. №1 — 3 экз.

Ч/З №1 — 2 экз.

Познакомьтесь с радикальными изменениями в языке JavaScript, которые произошли благодаря новому стандарту ECMAScript 6. Николас Закас — автор бестселлеров и эксперт-разработчик — создал самое полное руководство по новым типам объектов, синтаксису и интересным функциям. Каждая глава содержит примеры программ, которые будут работать в любой среде JavaScript и познакомят вас с новыми возможностями языка. Прочитав эту книгу, вы узнаете о том, чем полезны итераторы и генераторы, чем ссылочные функции отличаются от обычных, какие дополнительные опции позволяют работать с данными, о наследовании типов, об асинхронном программировании, о том, как модули меняют способ организации кода, и многом другом.

Более того, Николас Закас заглядывает в будущее, рассказывая про изменения, которые появятся в ECMAScript 7. Неважно, являетесь вы веб-разработчиком или работаете с node.js, в этой книге вы найдете самую необходимую информацию, позволяющую эффективно использовать все возможности ECMAScript 6.

12+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

Оглавление

Предисловие	16
Благодарности	18
Введение	19
История ECMAScript 6.....	19
О книге.....	20
Совместимость с браузерами и Node.js	20
Кому адресована книга	20
Обзор содержания	21
Используемые соглашения	22
От издательства	23
Глава 1. Блочные привязки	24
Объявление и местоположение переменных	24
Объявления на уровне блока.....	25
Объявления let.....	26
Повторное объявление недопустимо.....	26
Объявления const	27
Временная мертвая зона.....	29
Блочные привязки в циклах.....	30
Функции в циклах.....	31
Объявления let в циклах	32
Объявления const в циклах.....	33
Блочные привязки на глобальном уровне.....	34
Новые приемы, появившиеся с введением блочных привязок	35
В заключение	35

Глава 2. Строки и регулярные выражения.....	37
Улучшенная поддержка Юникода	37
Кодовые пункты UTF-16	37
Метод codePointAt()	39
Метод String.fromCodePoint()	39
Метод normalize()	40
Флаг u в регулярных выражениях.....	42
Другие изменения в поддержке строк	43
Методы идентификации подстроки	44
Метод repeat().....	45
Другие изменения в регулярных выражениях	45
Флаг y в регулярных выражениях.....	46
Создание копий регулярных выражений.....	48
Свойство flags.....	49
Литералы шаблонов.....	50
Основной синтаксис	51
Многострочный текст	51
Подстановка значений	53
Теги шаблонов.....	54
В заключение.....	58
Глава 3. Функции	59
Функции со значениями параметров по умолчанию	59
Имитация значений параметров по умолчанию в ECMAScript 5.....	59
Значения параметров по умолчанию в ECMAScript 6.....	60
Как значения параметров по умолчанию влияют на объект arguments	62
Выражения в параметрах по умолчанию.....	63
Временная мертвая зона для параметров по умолчанию	65
Неименованные параметры.....	67
Неименованные параметры в ECMAScript 5.....	67
Остаточные параметры.....	68
Дополнительные возможности конструктора Function.....	70
Оператор расширения.....	71
Свойство name.....	72
Выбор соответствующих имен.....	73
Специальные случаи свойства name.....	73

Двойственная природа функций	74
Как в ECMAScript 5 определить, каким способом вызвана функция	75
Метасвойство <code>new.target</code>	76
Функции уровня блоков.....	77
Когда использовать функции уровня блока.....	78
Функции уровня блока в нестрогом режиме	79
Стрелочные функции	79
Синтаксис стрелочных функций.....	81
Создание выражений немедленно вызываемых функций	82
Отсутствие привязки <code>this</code>	83
Стрелочные функции и массивы	85
Отсутствие привязки <code>arguments</code>	86
Идентификация стрелочных функций	86
Оптимизация хвостовых вызовов.....	87
Отличия хвостовых вызовов в ECMAScript 6.....	87
Как использовать оптимизацию хвостовых вызовов.....	89
В заключение	90
Глава 4. Расширенные возможности объектов.....	92
Категории объектов.....	92
Расширение синтаксиса литералов объектов	93
Сокращенный синтаксис инициализации свойств.....	93
Сокращенный синтаксис определения методов	94
Вычисляемые имена свойств.....	95
Новые методы	96
Метод <code>Object.is()</code>	96
Метод <code>Object.assign()</code>	97
Дубликаты свойств в литералах объектов	100
Порядок перечисления собственных свойств.....	101
Расширения в прототипах	102
Смена прототипа объекта	102
Простой доступ к прототипу с помощью ссылки <code>super</code>	103
Формальное определение метода.....	106
В заключение	107
Глава 5. Деструктуризация для упрощения доступа к данным	108
Какие выгоды дает деструктуризация?.....	108
Деструктуризация объектов.....	109

Присваивание с деструктуризацией	110
Значения по умолчанию.....	111
Присваивание локальным переменным с другими именами	112
Деструктуризация вложенных объектов.....	113
Деструктуризация массивов	115
Присваивание с деструктуризацией	116
Значения по умолчанию.....	117
Деструктуризация вложенных массивов	117
Остаточные элементы	118
Смешанная деструктуризация	119
Деструктурированные параметры	120
Деструктурированные параметры являются обязательными	121
Значения по умолчанию для деструктурированных параметров.....	122
В заключение	122
Глава 6. Символы и символьные свойства	124
Создание символов.....	124
Использование символов.....	126
Совместное использование символов.....	126
Приведение типов для символов.....	128
Извлечение символьных свойств	129
Экспортирование внутренних операций в виде стандартных символов	130
Метод Symbol.hasInstance.....	131
Свойство Symbol.isConcatSpreadable	132
Свойства Symbol.match, Symbol.replace, Symbol.search и Symbol.split	134
Метод Symbol.toPrimitive	136
Свойство Symbol.toStringTag	138
Свойство Symbol.unscopables	141
В заключение	143
Глава 7. Множества и ассоциативные массивы	144
Множества и ассоциативные массивы в ECMAScript 5	145
Недостатки обходных решений	145
Множества в ECMAScript 6.....	147
Создание множеств и добавление элементов	147
Удаление элементов	149
Метод forEach() для множеств	149
Преобразование множества в массив	151

Множества со слабыми ссылками	152
Ассоциативные массивы в ECMAScript 6.....	155
Методы ассоциативных массивов.....	156
Инициализация ассоциативных массивов	157
Метод forEach() ассоциативных массивов	157
Ассоциативные массивы со слабыми ссылками	158
В заключение	163
Глава 8. Итераторы и генераторы	165
Проблемы использования циклов	165
Что такое итераторы?	166
Что такое генераторы?	167
Выражения функций-генераторов	169
Методы-генераторы объектов.....	170
Итерируемые объекты и циклы for-of	170
Доступ к итератору по умолчанию	172
Создание итерируемых объектов.....	172
Встроенные итераторы.....	173
Итераторы коллекций.....	174
Итераторы строк	178
Итераторы NodeList.....	179
Оператор расширения и итерируемые объекты, не являющиеся массивами.....	180
Дополнительные возможности итераторов.....	181
Передача аргументов в итераторы	181
Возбуждение ошибок внутри итераторов	183
Инструкции return в генераторах	184
Делегирование генераторов	186
Асинхронное выполнение заданий.....	188
Простой инструмент выполнения заданий	189
Выполнение заданий с данными	190
Инструмент асинхронного выполнения заданий.....	191
В заключение	193
Глава 9. Введение в классы JavaScript	195
Структуры в ECMAScript 5, подобные классам	195
Объявление класса	196
Объявление простого класса	196
В чем преимущества синтаксиса определения классов?	197

Классы-выражения.....	199
Простой класс-выражение.....	200
Именованные классы-выражения	200
Классы как сущности первого класса	202
Свойства с методами доступа.....	203
Вычисляемые имена членов.....	205
Методы-генераторы	206
Статические члены	207
Наследование в производных классах.....	208
Затенение методов класса.....	211
Унаследованные статические члены	211
Производные классы из выражений	212
Наследование встроенных объектов.....	214
Свойство Symbol.species	216
Использование new.target в конструкторах классов	219
В заключение.....	221
Глава 10. Расширенные возможности массивов.....	222
Создание массивов	222
Метод Array.of().....	222
Метод Array.from()	224
Новые методы всех массивов.....	227
Методы find() и findIndex()	227
Метод fill()	228
Метод copyWithin()	229
Типизированные массивы.....	230
Числовые типы данных	231
Буферы массивов.....	231
Управление буферами массивов с помощью представлений	232
Сходства типизированных и обычных массивов.....	239
Общие методы.....	240
Те же самые итераторы	241
Методы of() и from().....	241
Различия типизированных и обычных массивов	242
Различия в поведении.....	242
Отсутствующие методы.....	243
Дополнительные методы.....	244
В заключение.....	245

Глава 11. Объект Promise и асинхронное программирование.....	246
Основы асинхронного программирования.....	246
Модель событий.....	247
Обратные вызовы.....	247
Основы объектов Promise.....	250
Жизненный цикл объекта Promise.....	250
Создание неустановившихся объектов Promise.....	253
Создание установившихся объектов Promise.....	255
Ошибки исполнителя.....	257
Глобальная обработка отклоненных объектов Promise.....	258
Обработка отказов в Node.js.....	259
Обработка отказов в браузерах.....	261
Составление цепочек из объектов Promise.....	263
Перехват ошибок.....	264
Возврат значений в цепочке объектов Promise.....	265
Возврат объектов Promise в цепочке.....	266
Обработка сразу нескольких объектов Promise.....	268
Метод Promise.all().....	268
Метод Promise.race().....	270
Наследование Promise.....	271
Выполнение асинхронных заданий с помощью Promise.....	272
В заключение.....	276
Глава 12. Прокси-объекты и Reflection API.....	278
Проблема с массивами.....	278
Введение в прокси-объекты и Reflection API.....	279
Создание простого прокси-объекта.....	280
Проверка свойств с помощью ловушки set.....	281
Проверка формы объектов с помощью ловушки get.....	283
Соккрытие свойств с помощью ловушки has.....	285
Предотвращение удаления свойств с помощью ловушки deleteProperty.....	286
Ловушки операций с прототипом.....	288
Как действуют ловушки операций с прототипом.....	288
Почему поддерживается два набора методов?.....	290
Ловушки, связанные с расширяемостью объектов.....	291
Два простых примера.....	291
Дубликаты методов управления расширяемостью.....	292

Ловушки операций с дескрипторами свойств	293
Блокирование вызова Object.defineProperty().....	294
Ограничения объекта дескриптора	295
Дубликаты методов для операций с дескрипторами.....	297
Ловушка ownKeys.....	298
Обработка вызовов функций с помощью ловушек apply и construct.....	299
Проверка параметров функции	300
Вызов конструкторов без ключевого слова new	302
Переопределение конструкторов абстрактных базовых классов	303
Вызываемые конструкторы классов	304
Отключение прокси-объектов.....	305
Решение проблемы с массивами.....	306
Определение индексов массива.....	307
Увеличение значения length при добавлении новых элементов.....	308
Удаление элементов при уменьшении значения length	309
Реализация класса MyArray	311
Использование прокси-объекта в качестве прототипа.....	313
Использование ловушки get в прототипе	314
Использование ловушки set в прототипе.....	315
Использование ловушки has в прототипе	316
Прокси-объекты как прототипы классов	317
В заключение	320
Глава 13. Инкапсуляция кода в модули.....	321
Что такое модули?	321
Основы экспортирования	322
Основы импортирования	323
Импортирование единственной привязки.....	324
Импортирование нескольких привязок	324
Импортирование всего модуля	324
Тонкая особенность импортированных привязок.....	326
Переименование экспортируемых и импортируемых привязок	326
Значения по умолчанию в модулях.....	327
Экспортирование значений по умолчанию	327
Импортирование значений по умолчанию.....	328
Реэкспорт привязки	329
Импортирование без привязок	330

Загрузка модулей	331
Использование модулей в веб-браузерах	331
Разрешение спецификаторов модулей в браузерах	336
В заключение	337
Приложение А. Мелкие изменения в ECMAScript 6	338
Новые приемы работы с целыми числами	338
Идентификация целых чисел	338
Безопасные целые числа	339
Новые математические методы	340
Идентификаторы с символами Юникода	341
Формализованное свойство <code>__proto__</code>	342
Приложение Б. Введение в ECMAScript 7 (2016)	344
Оператор возведения в степень	344
Порядок операций	345
Ограничения операндов	345
Метод <code>Array.prototype.includes()</code>	346
Как используется метод <code>Array.prototype.includes()</code>	346
Сравнение значений	347
Изменение области видимости функций в строгом режиме	348