

Научная библиотека

БНТУ



Изучаем программирование на JavaScript

Хорошо бы найти книгу по JavaScript, которая была бы приятнее визита к зубному врачу и понятнее налоговой декларации? Как жаль, что это только мечты...

Эрик Фримен
Элизабет Робсон



НАВУКОВАЯ БІБЛІЯТЭКА

Беларускага нацыянальнага
тэхнічнага ўніверсітэта

Інв. № 1887309

 ПИТЕР®

Москва · Санкт-Петербург · Минск

2022

Содержание (сводка)

| | |
|--|-----|
| Введение | 25 |
| 1 Первое знакомство с JavaScript. <i>В незнакомых водах</i> | 37 |
| 2 Настоящий код. <i>Следующий шаг</i> | 79 |
| 3 Знакомство с функциями. <i>Функции для всех</i> | 113 |
| 4 Наведение порядка в данных. <i>Массивы</i> | 157 |
| 5 Знакомьтесь: объекты. <i>Поездка в Объектвилль</i> | 203 |
| 6 Взаимодействие с веб-страницей. <i>Модель DOM</i> | 257 |
| 7 Типы, равенство, преобразования и все такое. <i>Серьезные типы</i> | 291 |
| 8 Все вместе. <i>Построение приложения</i> | 341 |
| 9 Асинхронное программирование. <i>Обработка событий</i> | 403 |
| 10 Первоклассные функции. <i>Функции без ограничений</i> | 449 |
| 11 Анонимные функции, область действия и замыкания. <i>Серьезные функции</i> | 495 |
| 12 Нетривиальное создание объектов. <i>Создание объектов</i> | 539 |
| 13 Использование прототипов. <i>Сильные объекты</i> | 579 |

Содержание (настоящее)

Введение

Ваш мозг и JavaScript. Вы учитесь — готовитесь к экзамену. Или пытаетесь освоить сложную техническую тему. Ваш мозг пытается оказать вам услугу. Он старается сделать так, чтобы на эту очевидно несущественную информацию не тратились драгоценные ресурсы. Их лучше потратить на что-нибудь важное. Так как же заставить его изучить JavaScript?



| | |
|---|----|
| Для кого написана эта книга | 24 |
| Мы знаем, о чем вы думаете | 25 |
| Эта книга для тех, кто хочет учиться | 26 |
| Метапознание: наука о мышлении | 27 |
| Вот что сделали МЫ: | 28 |
| Что можете сделать ВЫ, чтобы заставить свой мозг повиноваться | 29 |
| Примите к сведению | 30 |
| Научные редакторы | 33 |
| Благодарности | 34 |

первое знакомство с javascript

1

В незнакомых водах

JavaScript открывает фантастические возможности. JavaScript, основной язык программирования Всемирной паутины, позволяет определять расширенное поведение в веб-страницах. Забудьте о сухих, скучных, статичных страницах, которые просто занимают место на экране, — с JavaScript вы будете взаимодействовать с пользователями, реагировать на события, получать и использовать данные из Интернета, выводить графику... и многое, многое другое. При хорошем знании JavaScript вы сможете даже программировать совершенно новое поведение на своих страницах.

И не сомневайтесь — ваши знания будут востребованы. Сейчас JavaScript не только является одним из самых популярных языков программирования, но и поддерживается всеми современными (и многими несовременными) браузерами; более того, появились встроенные реализации JavaScript, существующие отдельно от браузеров. А впрочем, хватит разговоров. Пора браться за дело!



| | |
|--|----|
| Как работает JavaScript | 38 |
| Как пишется код JavaScript | 39 |
| Как включить код JavaScript в страницу | 40 |
| JavaScript, ты проделал длинный путь, детка... | 42 |
| Как создаются команды | 46 |
| Переменные и значения | 47 |
| Осторожно, ключевые слова! | 48 |
| Поаккуратнее с выражениями! | 51 |
| Многokратное выполнение операций | 53 |
| Как работает цикл while | 54 |
| Принятие решений в JavaScript | 58 |
| А если нужно принять МНОГО решений... | 59 |
| Привлекайте пользователя к взаимодействию со страницей | 61 |
| Близкое знакомство с console.log | 63 |
| Как открыть консоль | 64 |
| Пишем серьезное приложение на JavaScript | 65 |
| Как добавить код в страницу? (считаем способы) | 68 |
| Разметка и код: пути расходятся | 69 |

2



Следующий шаг

Вы уже знаете, что такое переменные, типы, выражения... и так далее. Вы уже кое-что знаете о JavaScript. Более того, знаний достаточно для того, чтобы начать писать настоящие программы, которые делают что-то интересное, которыми кто-то будет пользоваться. Правда, вам не хватает практического опыта написания кода, и мы прямо сейчас начнем решать эту проблему. Как? А просто возьмемся за написание несложной игры, полностью реализованной на JavaScript. Задача масштабная, но мы будем двигаться к цели постепенно, шаг за шагом. Итак, беремся за дело, а если вам вдруг захочется использовать нашу разработку в своих проектах — мы не против, распоряжайтесь кодом, как считаете нужным.



| | |
|--|-----|
| Давайте реализуем игру «Морской бой» | 80 |
| Первый заход... | 80 |
| Начнем с проектирования | 81 |
| Разбираем псевдокод | 83 |
| Стоп! Прежде чем идти дальше, вспомните про HTML! | 85 |
| Пишем код упрощенной версии «Морского боя» | 86 |
| Переходим к реализации логики | 87 |
| Как работает функция prompt | 89 |
| Проверка на попадание | 90 |
| Добавление кода проверки попадания | 93 |
| Вывод данных после игры | 94 |
| Реализация логики готова! | 96 |
| Немного о контроле качества | 97 |
| А нельзя ли покороче... | 101 |
| Упрощенный «Морской бой» почти готов | 102 |
| Как получить случайную позицию | 103 |
| Всемирно известный рецепт генерирования случайных чисел | 103 |
| Возвращаемся к контролю качества | 105 |
| Поздравляем, вы создали свою первую программу на JavaScript! Теперь пара слов о повторном использовании кода | 107 |

3

Функции для всех

В этой главе вы овладеете своей первой суперспособностью. Вы уже кое-что знаете о программировании; пришло время сделать следующий шаг и освоить работу с функциями. Функции позволяют писать код, который может повторно использоваться в разных ситуациях; код, существенно более простой в сопровождении; код, который можно абстрагировать и присвоить ему простое имя, чтобы вы могли забыть о рутинных подробностях и заняться действительно важными делами. Вы увидите, что функции не только открывают путь к мастерству программиста, но и играют ключевую роль в стиле программирования JavaScript. В этой главе мы начнем с основ: механики и всех тонкостей работы функций, а потом в оставшейся части книги будем совершенствовать ваши навыки работы с функциями. Итак, начнем с азов... прямо сейчас.



| | |
|--|-----|
| Так чем плох этот код? | 115 |
| Кстати, а вы когда-нибудь слышали о ФУНКЦИЯХ? | 117 |
| Хорошо, но как все это работает? | 118 |
| Что можно передать функции? | 123 |
| В JavaScript используется передача по значению | 126 |
| Эксперименты с функциями | 128 |
| А еще функции могут возвращать значения | 129 |
| Пошаговое выполнение функции с командой return | 130 |
| Глобальные и локальные переменные | 133 |
| Область действия локальных и глобальных переменных | 135 |
| Короткая жизнь переменных | 136 |
| Не забывайте объявлять локальные переменные! | 137 |

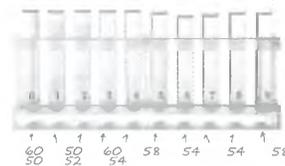


наведение порядка в данных

4

Массивы

JavaScript может работать не только с числами, строками и логическими значениями. До настоящего момента мы работали исключительно с **примитивами** — простыми строками, числами и логическими значениями (например, «Fido», 23 и true). С примитивными типами можно сделать многое, но в какой-то момент возникнет необходимость в **расширенных данных** для представления всех позиций в корзине покупок, всех песен в плейлисте, группы звезд и их звездных величин или целого каталога продуктов. Подобные задачи требуют более серьезных средств. Типичным инструментом для представления таких однородных данных является **массив** JavaScript. В этой главе вы узнаете, как помещать данные в массив, передавать их и работать с ними. В последующих главах будут рассмотрены другие способы **структурирования данных**, но начнем мы с массивов.



| | |
|--|-----|
| Вы нам поможете? | 158 |
| Как представить набор значений в JavaScript | 159 |
| Как работают массивы | 160 |
| Сколько же элементов в массиве? | 162 |
| Генератор Красивых Фраз | 164 |
| Тем временем в фирме Bubbles-R-Us... | 167 |
| Как перебрать элементы массива | 170 |
| Но постойте, существует и более удобный способ перебора! | 172 |
| Что, опять?.. Нельзя ли покороче? | 178 |
| Доработка цикла for с оператором постфиксного увеличения | 179 |
| Создание пустого массива (и добавление элементов) | 183 |
| А вот и наши победители... | 187 |
| Краткий обзор кода... | 189 |
| Работа над функцией printAndGetHighScore | 190 |
| Рефакторинг кода с определением функции printAndGetHighScore | 191 |
| А теперь все вместе... | 193 |



знакомьтесь: объекты

5

Поездка в Объектвилль

До настоящего момента мы использовали примитивы и массивы. И при этом применялась методология процедурного программирования с простыми командами, условиями, циклами `for/while` и функциями. Такой подход был далек от принципов объектно-ориентированного программирования. Собственно, он вообще не имел *ничего общего* с объектно-ориентированным программированием. Мы использовали объекты время от времени (причем вы об этом даже не знали), но еще не написали ни одного собственного объекта. Пришло время покинуть скучный процедурный город и заняться созданием собственных **объектов**. В этой главе вы узнаете, почему объекты сильно улучшают нашу жизнь — во всяком случае в области **программирования**. Так и знайте: привыкнув к объектам, вы уже не захотите возвращаться обратно. Да, и не забудьте прислать открытку, когда обживетесь.



| | |
|---|-----|
| Кто-то сказал «объекты»?! | 204 |
| Подробнее о свойствах... | 205 |
| Как создать объект | 207 |
| Что такое «объектно-ориентированный подход»? | 210 |
| Как работают свойства | 211 |
| Как объект хранится в переменной? | |
| Любознательные умы интересуются... | 216 |
| Сравнение примитивов с объектами | 217 |
| Объекты способны на большее... | 218 |
| Предварительная проверка | 219 |
| Проверка шаг за шагом | 220 |
| Еще немного поговорим о передаче объектов функциям | 222 |
| Ведите себя прилично! И объекты свои научите... | 228 |
| Усовершенствование метода <code>drive</code> | 229 |
| Почему метод <code>drive</code> не знает о свойстве <code>started</code> ? | 232 |
| Как работает <code>this</code> | 234 |
| Как поведение влияет на состояние | 240 |
| Состояние влияет на поведение | 241 |
| Поздравляем с первыми объектами! | 243 |
| Представьте, вас окружают сплошные объекты! (и они упрощают вашу работу) | 244 |

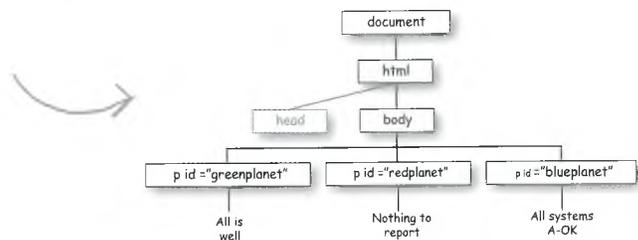
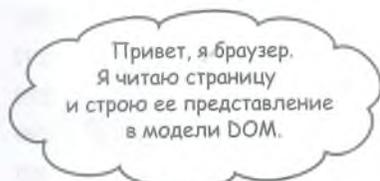
Взаимодействие с Веб-страницей

6

Модель DOM

Вы значительно продвинулись в изучении JavaScript. Фактически вы из новичка в области сценарного программирования превратились в... **программиста**. Впрочем, кое-чего не хватает: для полноценного использования ваших навыков JavaScript необходимо уметь взаимодействовать с веб-страницей, в которой располагается ваш код. Только в этом случае вы сможете писать **динамические** страницы, которые реагируют на действия пользователя и обновляются после загрузки. Как взаимодействовать со страницей? Через объектную модель документа **DOM (Document Object Model)**. В этой главе мы рассмотрим DOM и общие принципы работы с этой моделью из JavaScript для расширения возможностей страницы.

| | |
|---|-----|
| В предыдущей главе мы предложили вам одну головоломку на «вскрытие кода» | 258 |
| Что же делает этот код? | 259 |
| Как JavaScript на самом деле взаимодействует со страницей | 261 |
| Как приготовить модель DOM | 262 |
| DOM: первые впечатления | 263 |
| Получение элемента методом getElementById | 268 |
| Что именно мы получаем от DOM? | 269 |
| В поисках внутреннего HTML | 270 |
| Что происходит при внесении изменений в DOM | 272 |
| И не вздумайте выполнять мой код до того, как страница будет загружена! | 277 |
| «Обработчик события» или «функция обратного вызова» | 278 |
| Как задать атрибут методом setAttribute | 283 |
| Веселье с атрибутами продолжается! (значения атрибутов можно ЧИТАТЬ) | 284 |
| И не забывайте, что getElementById тоже может вернуть null! | 284 |
| Каждый раз, когда вы запрашиваете некоторое значение, стоит убедиться в том, что вы получили то, что просили... | 284 |
| Что еще можно сделать с моделью DOM? | 286 |

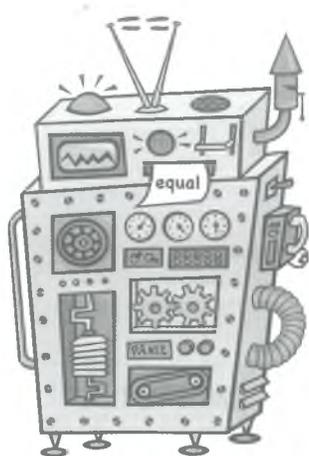


типы, равенство, преобразования и Все такое

7

Серьезные типы

Настало время серьезно поговорить о типах. Одна из замечательных особенностей JavaScript заключается в том, что начинающий может достаточно далеко продвинуться, не углубляясь в подробности языка. Но чтобы действительно **овладеть языком**, получить повышение по работе и заняться тем, чем действительно стоит заниматься, нужно хорошо разбираться в **типах**. Помните, что мы говорили о JavaScript, — что у этого языка не было такой роскоши, как академическое определение, прошедшее экспертную оценку? Да, это правда, но отсутствие академической основы не помешало Стиву Джобсу и Биллу Гейтсу; не помешало оно и JavaScript. Это означает, что система типов JavaScript... ну, скажем так — не самая продуманная, и в ней найдется немало **странностей**. Но не беспокойтесь, в этой главе мы все разберем, и вскоре вы научитесь благополучно обходить все эти неприятные моменты с типами.



| | |
|--|-----|
| Истина где-то рядом... | 292 |
| Будьте осторожны: undefined иногда появляется совершенно неожиданно... | 294 |
| Как использовать null | 297 |
| Работа с NaN | 299 |
| А дальше еще удивительнее | 299 |
| Мы должны вам признаться... | 301 |
| Оператор проверки равенства (также известный как ==) | 302 |
| Как происходит преобразование операндов (все не так страшно, как может показаться) | 303 |
| Как проверить строгое равенство | 306 |
| Еще больше преобразований типов... | 312 |
| Как проверить два объекта на равенство | 315 |
| Псевдоистина где-то рядом... | 317 |
| Что JavaScript считает «псевдоложью» | 318 |
| Тайная жизнь строк | 320 |
| Строка может выглядеть и как примитив, и как объект | 321 |
| Краткий обзор методов (и свойств) строк | 323 |
| Битва за кресло | 327 |

Все Вместе

8

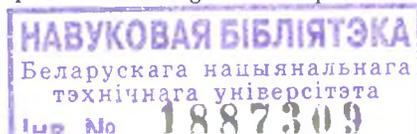
Построение приложения

Подготовьте свой инструментарий к работе. Да, ваш инструментарий — ваши новые навыки программирования, ваше знание DOM и даже некоторое знание HTML и CSS. В этой главе мы объединим все это для создания своего первого полноценного **веб-приложения**. Довольно **примитивных игр** с одним кораблем, который размещается в одной строке. В этой главе мы построим **полную версию**: большое игровое поле, несколько кораблей, ввод данных пользователем прямо на веб-странице. Мы создадим структуру страницы игры в разметке HTML, применим визуальное оформление средствами CSS и напомним код JavaScript, определяющий поведение игры. Приготовьтесь: в этой главе мы займемся полноценным, серьезным программированием и напомним вполне серьезный код.



| | |
|--|-----|
| На этот раз мы построим НАСТОЯЩУЮ игру «Морской бой» | 342 |
| Возвращаемся к HTML и CSS | 343 |
| Создание страницы HTML: общая картина | 344 |
| Добавление стиливого оформления | 348 |
| Использование классов hit и miss | 351 |
| Как спроектировать игру | 353 |
| Реализация представления | 355 |
| Как работает метод <code>displayMessage</code> | 355 |
| Как работают методы <code>displayHit</code> и <code>displayMiss</code> | 357 |
| Модель | 360 |
| Как мы будем представлять данные кораблей | 362 |
| Реализация объекта модели | 365 |
| Подготовка метода <code>fire</code> | 366 |
| Реализация контроллера | 373 |
| Обработка выстрела | 374 |
| Планирование кода... | 375 |
| Реализация метода <code>parseGuess</code> | 376 |
| Подсчет и обработка выстрелов | 379 |
| Как связать обработчик событий с кнопкой <code>Fire</code> | 383 |
| Передача данных контроллеру | 384 |
| Как размещать корабли | 388 |
| Метод <code>generateShip</code> | 389 |
| Генерирование начальной позиции нового корабля | 390 |
| Завершение метода <code>generateShip</code> | 391 |

| | | | | | | | | | | | |
|---|-----------|---|---|-----------|---|---|---|--|--|--|--|
| A | | | | | | | | | | | |
| B | Корабль 1 | | | | | | | | | | |
| C | | | | | | | | | | | |
| D | Корабль 1 | | | Корабль 2 | | | | | | | |
| E | | | | | | | | | | | |
| F | | | | | | | | | | | |
| G | | | | Корабль 3 | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | | | | |



9

Обработка событий

В этой главе вам предстоит подняться на принципиально новый уровень. До настоящего момента мы писали код, который обычно выполняется сверху вниз. Конечно, в нем использовались функции, объекты и методы, но выполнение шло по заранее намеченной колее. Жаль, что нам приходится сообщать такие новости во второй половине книги, но **такая структура кода не характерна для JavaScript.** Большая часть кода JavaScript пишется **для обработки событий.** Каких событий? Да любых. Пользователь щелкает на странице, данные поступают из сети, в браузере срабатывает таймер, в DOM происходят изменения... Это далеко не полный список. Более того, в браузере **постоянно** происходят события, которые в основном остаются незамеченными. В этой главе мы пересмотрим свой подход к программированию и узнаем, для чего же нужно писать код, реагирующий на события.



| | |
|--|-----|
| Что такое «событие»? | 405 |
| Что такое «обработчик события»? | 406 |
| Как создать первый обработчик событий | 407 |
| Тест-драйв | 408 |
| Как разобраться в событиях? Написать игру, конечно! | 410 |
| Реализация игры | 411 |
| Тест-драйв | 412 |
| Добавим несколько изображений | 416 |
| Теперь нужно назначить один обработчик всем свойствам onclick всех изображений | 417 |
| Как использовать один обработчик для всех изображений | 418 |
| Как работает объект события | 421 |
| Работаем с объектом события | 423 |
| Тест-драйв: объект события и источник | 424 |
| Очереди и события | 426 |
| Еще больше событий | 429 |
| Как работает setTimeout | 430 |
| Завершение кода игры | 434 |
| Тест-драйв таймеров | 435 |





первоклассные функции

10

Функции без ограничений

Изучайте функции и блистайте. В каждом ремесле, искусстве и дисциплине есть ключевой принцип, который отличает игроков «среднего звена» от настоящего профессионала, — и когда речь заходит о JavaScript, признаком профессионализма является хорошее понимание **функций**. Функции играют фундаментальную роль в JavaScript, и многие приемы, применяемые при **проектировании и организации** кода, основаны на хорошем знании функций и умении использовать их. Путь изучения функций на этом уровне интересен и непросто, так что приготовьтесь... Эта глава немного напоминает экскурсию по шоколадной фабрике Вилли Вонка — во время изучения функций JavaScript вы увидите немало странного, безумного и замечательного.

| | |
|---|-----|
| Двойная жизнь ключевого слова function | 450 |
| Объявления функций и функциональные выражения | 451 |
| Разбор объявления функции | 452 |
| Что дальше? Браузер выполняет код | 453 |
| Двигаемся вперед... Проверка условия | 454 |
| И напоследок... | 455 |
| Функции как значения | 459 |
| Функции как полноправные граждане JavaScript | 462 |
| Полеты первым классом | 463 |
| Написание кода для обработки и проверки пассажиров | 464 |
| Перебор пассажиров | 466 |
| Передача функции другой функции | 467 |
| Тест-драйв... вернее, полет | 467 |
| Возвращение функций из функций | 470 |
| Код заказа напитков | 471 |
| Код заказа напитков: другой подход | 472 |
| Постойте, одного напитка недостаточно! | 473 |
| Заказ напитков с использованием первоклассной функции | 474 |
| Тест-драйв-полет | 475 |
| «Веб-кола» | 477 |
| Как работает метод массивов sort | 479 |
| Все вместе | 480 |
| Тем временем в «Веб-коле» | 481 |
| Тест-драйв сортировки | 482 |



Серьезные функции

Мы узнали много нового о функциях, но это далеко не всё. В этой главе мы пойдем дальше и разберемся в темах, которыми обычно занимаются профессионалы. Вы научитесь **действительно эффективно** работать с функциями. Глава будет не слишком длинной, но с довольно интенсивным изложением материала, так что к концу главы выразительность вашего кода JavaScript превзойдет все ожидания. Вы также будете готовы к тому, чтобы взяться за анализ кода коллеги или заняться изучением библиотеки JavaScript с открытым кодом, потому что мы заодно изучим некоторые распространенные идиомы и соглашения, связанные с использованием функций. А если вы никогда не слышали об **анонимных функциях и замыканиях**, то это самое подходящее место для знакомства!



| | |
|--|-----|
| Посмотрим на функции с другой стороны... | 496 |
| Как использовать анонимную функцию? | 497 |
| Когда определяется функция? Здесь возможны варианты... | 503 |
| Что произошло? Почему функция <code>flu</code> не определена? | 504 |
| Как создаются вложенные функции | 505 |
| Как вложение влияет на область действия | 506 |
| В двух словах о лексической области действия | 508 |
| Чем интересна лексическая область действия | 509 |
| Снова о функциях | 511 |
| Вызовы функций (снова) | 512 |
| Что такое «замыкание»? | 515 |
| Как замкнуть функцию | 516 |
| Использование замыканий для реализации счетчика | 518 |
| Тест-драйв волшебного счетчика | 519 |
| Взгляд за кулисы | 519 |
| Создание замыкания с передачей функционального выражения в аргументе | 521 |
| Замыкание содержит непосредственное окружение, а не его копию | 522 |
| Создание замыкания в обработчике события | 523 |
| Программа без замыкания | 524 |
| Программа с замыканием | 524 |
| Тест-драйв счетчика нажатий | 525 |
| Как работает замыкание | 526 |

нетривиальное создание объектов

12

Создание объектов

До настоящего момента мы создавали объекты вручную. Для каждого объекта использовался **объектный литерал**, который задавал все без исключения свойства. Для небольших программ это допустимо, но для серьезного кода потребуется что-то получше, а именно **конструкторы объектов**. Конструкторы упрощают создание объектов, причем вы можете создавать объекты по единому **шаблону** — иначе говоря, конструкторы позволяют создавать серии объектов, обладающих одинаковыми свойствами и содержащих одинаковые методы. Код, написанный с использованием конструкторов, получается гораздо более **компактным** и снижает риск ошибок при создании большого количества объектов. Уверяем, что после изучения этой главы вы будете пользоваться конструкторами так, словно занимались этим всю сознательную жизнь.



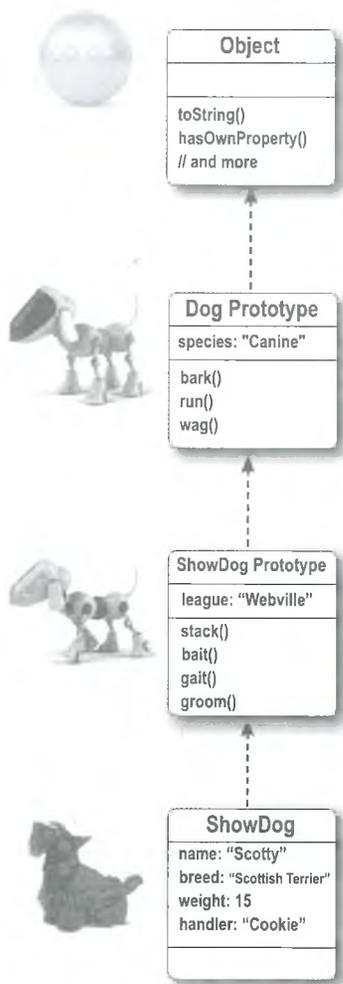
| | |
|---|-----|
| Создание объектов с использованием объектных литералов | 540 |
| О сходстве и различии между объектами | 541 |
| Конструкторы | 543 |
| Как создать конструктор | 544 |
| Как использовать конструктор | 545 |
| Как работают конструкторы | 546 |
| В конструкторы также можно добавить методы | 548 |
| Опасная зона | 551 |
| Техника безопасности | 551 |
| Даешь массовое производство! | 554 |
| Тест-драйв на новых машинах | 556 |
| Не спешите расставаться с объектными литералами | 557 |
| Преобразование аргументов в объектный литерал | 558 |
| Преобразование конструктора <code>Car</code> | 559 |
| Экземпляры | 561 |
| Даже сконструированные объекты могут содержать независимые свойства | 564 |
| Конструкторы в реальном мире | 566 |
| Объект <code>Array</code> | 567 |
| Другие встроенные объекты | 569 |

13

использование прототипов

Сильные объекты

Научиться создавать объекты — только начало. Пришло время «накачать мышцы» — изучить расширенные средства определения **отношений** между объектами и организовать **совместное использование кода**. Кроме того, нам понадобятся механизмы расширения существующих объектов. Иначе говоря, нам нужно расширить свой инструментарий работы с объектами. В этой главе вы увидите, что в JavaScript реализована достаточно мощная **объектная модель**, но она немного отличается от модели традиционных объектно-ориентированных языков. Вместо типичных объектно-ориентированных систем на базе классов JavaScript использует модель **прототипов** — объектов, способных наследовать и расширять поведение других объектов. Какая в этом польза для вас? Вскоре узнаете. Итак, за дело...



| | |
|--|-----|
| Представление объектов на диаграммах | 581 |
| Снова о конструкторах: код используется повторно, но насколько эффективно? | 582 |
| Дублирование методов действительно создает проблемы? | 584 |
| Что такое «прототип»? | 585 |
| Наследование через прототип | 586 |
| Как работает наследование | 587 |
| Переопределение прототипа | 589 |
| Как получить прототип | 591 |
| Как создать прототип | 592 |
| Переопределение унаследованного метода | 594 |
| О динамических прототипах | 598 |
| Более интересная реализация метода sit | 600 |
| Еще раз: как работает свойство sitting | 601 |
| С чего начать проектирование объектов | 605 |
| Создание цепочки прототипов | 607 |
| Как работает наследование в цепочке прототипов | 608 |
| Анализ результатов | 617 |
| Наводим чистоту | 618 |
| Еще немного усилий | 619 |
| Вызов Dog.call шаг за шагом | 620 |
| Применяем наследование с пользой... расширяя встроенный объект | 626 |
| Теория великого объединения Всего JavaScript | 628 |
| Объекты для лучшей жизни | 628 |
| Собираем все вместе | 629 |