

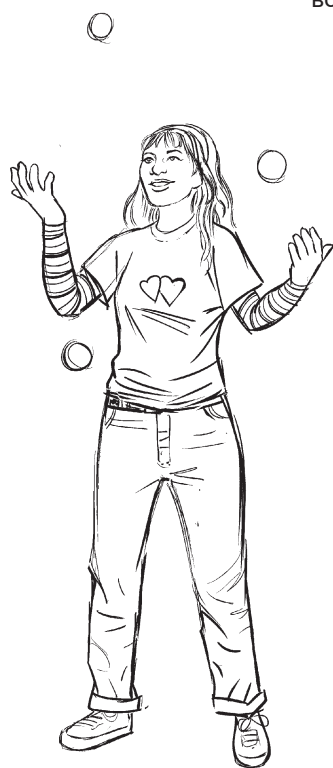
# 1 Первое знакомство с JavaScript

## В незнакомых водах

### JavaScript открывает фантастические возможности.

JavaScript, **основной язык программирования** Всемирной паутины, позволяет **добавлять поведение** к веб-страницам. 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 на страницах... и многое, многое другое. При хорошем знании JavaScript вы сможете дальше программировать **совершенно новое** поведение своих пользователей.

И вы будете в хорошей компании. JavaScript — не только один из **самых популярных** языков программирования, но он еще и **поддерживается** всеми современными браузерами; более того, появились встроенные реализации JavaScript, существующие отдельно от браузеров. А впрочем, хватит разговоров. Пора браться за дело!



Как работает JavaScript	36
Как пишется код JavaScript	37
Как включить код JavaScript в страницу	38
JavaScript, ты проделал длинный путь...	40
Как создаются команды	44
Переменные и значения	45
Константы	46
Осторожно, ключевые слова!	48
Поаккуратнее с выражениями	51
Многократное 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

## Пастрящий код

### Следующий шаг

**Вы уже знаете, что такое переменные, типы, выражения... и так далее.** Вы уже кое-что знаете о JavaScript. Более того, этих знаний достаточно, чтобы начать писать настоящие программы, которые делают что-то интересное и 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
Чуть подробнее...	82
Разбираем псевдокод	83
Стоп! Прежде чем идти дальше, вспомните об HTML!	85
Пишем код упрощенной версии «Морского боя»	86
Переходим к реализации логики	87
Шаг 1: создание цикла, получение данных	88
Как работает функция prompt	89
Шаг 2: проверка на попадание	90
Ну что, попал?	92
Добавление кода проверки попадания	93
Шаг 3: да тебе просто повезло!	93
Шаг 4: 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	102
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



## Знакомство с функциями

## 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 рутине 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 видимости локальных и глобальных переменных	135
Но это еще не все	136
Не забывайте объявлять локальные переменные!	138
Короткая жизнь переменных	139

# 4

## Приведение порядка в данных

### 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?	160
Как представить набор значений в JavaScript	160
Как работают массивы	162
Как обратиться к элементу массива	163
Обновление значения в массиве	163
Сколько же элементов в массиве?	164
Генератор Красивых Фраз	166
Тем временем в фирме Bubbles-R-Us...	169
Как перебрать элементы массива	172
Но постойте, существует и более удобный способ перебора!	174
Тест-драйв	178
Что, опять?.. Нельзя ли покороче?	181
Доработка цикла for с оператором постфиксного увеличения	182
Короткий тест-драйв	182
Создание пустого массива (и добавление элементов)	186
Тест-драйв итоговой версии отчета	190
А вот и наши победители...	190
Краткий обзор кода	192
Работа над функцией printAndGetHighScore	193
Рефакторинг кода с определением функции printAndGetHighScore	194
А теперь все вместе...	196

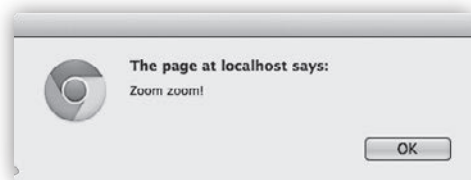
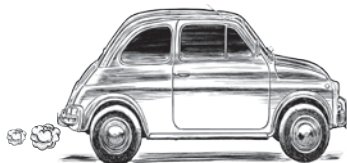
# 5

## Знакомьтесь: объекты

### 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»?!	208
Подробнее о свойствах...	209
Как создать объект	211
Что такое «объектно-ориентированный подход»?	214
Как работают свойства	215
Как объект хранится в переменной? Любознательные умы интересуются...	220
Сравнение примитивов с объектами	221
Объекты способны на большее...	222
Предварительная проверка	223
И что получилось?	223
Проверка шаг за шагом	224
Еще несколько слов о передаче объектов функциям	226
Генератор машин	229
Ведите себя прилично! И объекты свои научите...	232
Усовершенствование метода <code>drive</code>	233
Машина отправляется на тест-драйв	235
Стоп, не так быстро...	235
Почему метод <code>drive</code> не знает о свойстве <code>started</code> ?	236
Тест-драйв для <code>this</code>	237
Как работает <code>this</code>	238
Сокращенная запись методов	244
Поведение влияет на состояние	245
Состояние влияет на поведение	246
Приготовиться к тест-драйву	246
Поздравляем с первыми объектами!	248
Представьте, вас окружают сплошные объекты!	249



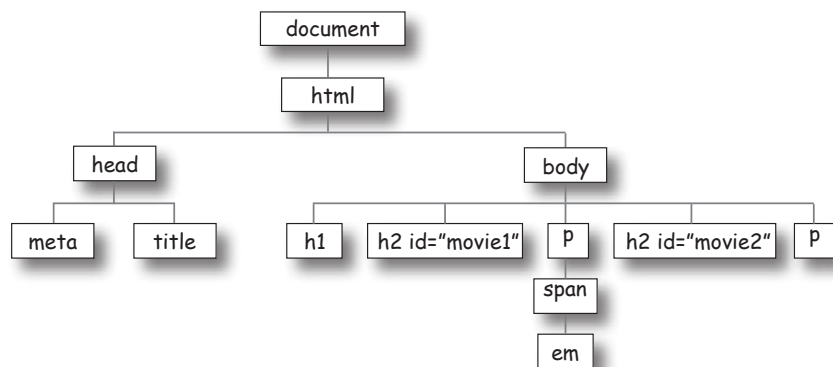
# 6 Взаимодействие с Веб-страницей

## 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»	264
Что же делает этот код?	265
В двух словах	266
Как JavaScript на самом деле взаимодействует со страницей	267
Как приготовить модель DOM	268
DOM: первые впечатления	269
Получение элемента методом getElementById	274
Что именно мы получаем от DOM?	275
В поисках внутреннего HTML	276
Что происходит при внесении изменений в DOM	278
Планетарный тест-драйв	281
И не вздумайте выполнять мой код до того, как страница будет загружена!	283
Обработчик события, или Функция обратного вызова	284
Зачем на этом останавливаться? Сделаем следующий шаг	288
Как задать атрибут методом setAttribute	289
Веселье с атрибутами продолжается!	290
А тем временем в далекой галактике...	291
Что еще можно сделать с моделью DOM?	292



## 1

Типы, равенство, преобразования и все такое

## 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 в полной мере **овладеть языком**, получить повышение по работе и заняться тем, чем действительно стоит заниматься, нужно хорошо разбираться в **типах**. Помните, что мы говорили о JavaScript в главе 1, — 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...	298
Будьте осторожны: undefined иногда появляется совершенно неожиданно...	300
Как использовать null	303
Работа с NaN	305
А дальше еще удивительнее...	305
Мы должны вам признаться...	307
Оператор проверки равенства (также известный как ==)	308
Как происходит преобразование операндов (все не так страшно, как может показаться)	309
Как проверить строгое равенство	312
Еще больше преобразований типов...	318
Как проверить два объекта на равенство	321
Псевдоистина где-то рядом...	323
Что JavaScript считает «псевдоложью»	324
Тайная жизнь строк	326
Строка может выглядеть и как примитив, и как объект	327
Как работают литералы шаблонов	330
Краткий обзор методов и свойств строк	331
Битва за кресло	336



# 8

## Всё вместе

### 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 всё 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 напишем код JavaScript, определяющий поведение игры. Приготовьтесь: в этой главе мы займемся полноценным, мощным программированием и напишем вполне серьезный код.



На этот раз мы построим НАСТОЯЩУЮ игру «Морской бой»	352
Возвращаемся к HTML и CSS	353
Создание страницы HTML: общая картина	354
Добавление стилевого оформления	358
Использование классов hit и miss	361
Как спроектировать игру	363
Реализация представления	365
Модель	370
Нам нужны большие корабли... и большое игровое поле	371
Как мы будем представлять данные кораблей	372
А теперь всё вместе...	378
Просили же — выразаться покороче!	379
Взаимодействие с представлением	381
Реализация контроллера	383
Обработка выстрела	384
Получение данных от игрока	393
Как размещать корабли	399
Как избежать столкновений	403
Поздравляем, можно запускать старт!	406

## Обработка событий

## 9

## Асинхронность

**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 нехарактерна 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»?	419
Что такое «обработчик события»?	420
Создание обработчика события	421
Как разобраться в событиях? Написать игру, конечно!	424
Реализация игры	425
Добавим несколько изображений	430
Теперь нужно назначить один обработчик всем свойствам onclick всех изображений	431
Как использовать один обработчик для всех изображений	432
Как работает объект события	435
Работаем с объектом события	437
Тест-драйв: объект события и источник	438
События и очереди	440
Еще больше событий	443
Как работает setTimeout	444
Завершение кода игры	448
Тест-драйв таймеров	449

## 10

## Анонимные функции и функции высшего порядка

## 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 <code>function</code>	464
Объявления функций и функциональные выражения	465
Раз функции — это значения, мы можем присваивать их переменным	468
Функции как полноправные граждане JavaScript	471
Еще одна сторона функций...	472
Как использовать анонимные функции	473
И снова о компактности	475
Со стрелочными функциями код становится еще компактнее	477
Создание стрелочных функций	479
«Веб-кола»	482
Как работает метод массивов <code>sort</code>	484
Всё вместе	485
Тем временем в «Веб-коле»	486
Функции высшего порядка	489
Фильтрация с использованием функций высшего порядка	490
Не забудьте об анонимных и стрелочных функциях	491
Использование <code>reduce</code> для получения общего количества упаковок	494
Сцепление <code>map</code> , <code>filter</code> и <code>reduce</code>	496
Перебор с <code>forEach</code>	499



## 11

Современный синтаксис, лексическая область видимости и замыкания

## 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 расширите свое владение синтаксисом и научитесь действительно эффективно пользоваться аргументами, параметрами и присваиванием. Мы вернемся к областям видимости и некоторым особенностям их в JavaScript. Изучение нюансов областей видимости приведет нас к замыканиям — концепции, которая нередко считается чем-то мистическим, но является чрезвычайно важной для освоения JavaScript. В результате ваш код JavaScript станет более выразительным, чем вы могли себе представить.

Серьезно о функциональном синтаксисе	510
Расширение аргументов	513
Мы кое-что не рассказали вам о функциях...	515
Объявления функций «поднимаются»	516
С объявлениями функций мы разобрались, теперь можно заняться всем остальным	517
Поговорим об области видимости	521
Функции вне глобальной области видимости	522
О лексической области видимости	525
Другой взгляд на внешние/внутренние функции	526
Использование области видимости для инкапсуляции	528
Два важных правила областей видимости JavaScript	530
Разгадка тайны	534
Как создать замыкание	534
Тест-драйв волшебного счетчика	540
Реализация счетчика на базе замыкания	541
Как работает makeTimer	545
Реализация onlyOnceMaker	550



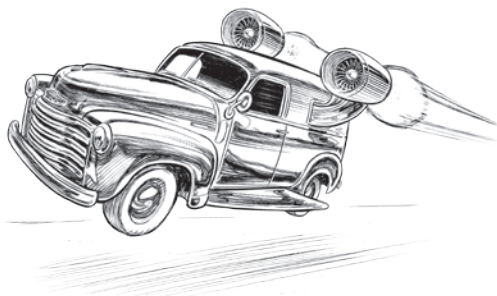
# 12

## Центривуальное создание объектов

### 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. На помощь приходят **классы**. Классы упрощают создание объектов, причем вы можете создавать объекты по единому **шаблону**, — иначе говоря, классы гарантируют, что все объекты обладают одинаковыми свойствами и содержат одинаковые методы. Код, написанный с использованием классов, получается гораздо более **компактным** и снижает риск ошибок при создании большого количества объектов. Итак, за дело...

Создание объектов с использованием объектных литералов	560
О сходстве и различии между объектами	561
Знакомство с классами	563
Как определить класс	564
Как создать объект на основе класса	565
Как работают классы	567
Добавление методов	570
Дашь массовое производство!	574
Базовый класс Car	576
Реализация класса Taxi с extends	579
Добавление новых методов в класс Taxi	580
Реализация класса RocketCar	583
Использование объектного литерала для упрощения конструктора	587
Переработка конструктора Car	590
Свойства доступа	591
Использование геттеров	592
Что есть геттер без сеттера?	593
Статические свойства и методы	598
Подсчет выпущенных машин	600



## Приложение: остратки

# А

### Главные десять тем... (не рассмотренных в книге)

**Мы прошли долгий путь, и книга почти завершена.** Мы будем скучать, но прежде чем расстаться, хотим сказать еще несколько напутственных слов. В эту относительно маленькую главу не удастся включить все, что вам нужно знать. На самом деле мы *попытались* уместить в книгу все, что необходимо знать о программировании на JavaScript (что еще не было описано в других главах), сократив размер шрифта до .00004. Все поместилось, но никто не мог прочитать текст. В общем, мы отказались от этих попыток и оставили самое интересное для этого приложения.

И это *действительно* последний раздел книги.

№ 1. Модули	608
№ 2. JSON	610
№ 3. Promise	611
№ 4. Деструктурирующее присваивание	612
№ 5. Symbol и BigInt	613
№ 6. Map и Set	614
№ 7. Расширенные возможности работы с DOM	617
№ 8. Объект window	618
№ 9. JavaScript на стороне сервера	619
№ 10. Рекурсия	620

