ЛЕГКОЕ ПРОГРАММИРОВАНИЕ

JAVASCRIPT ДЛЯ ДЕТЕЙ

САМОУЧИТЕЛЬ ПО ПРОГРАММИРОВАНИЮ

НИК МОРГАН

МИФ ДЕТСТВО

no starch press

By Nick Morgan

JAVASCRIPT FOR KIDS

A PLAYFUL INTRODUCTION TO PROGRAMMING

no starch press

San Francisco

Ник Морган

JAVASCRIPT ДЛЯ ДЕТЕЙ

САМОУЧИТЕЛЬ
ПО ПРОГРАММИРОВАНИЮ

Москва
«Манн, Иванов и Фербер»
2016

УДК 087.5:004.43
ББК 76.1,62:32.973.412
М79

Перевод с английского Станислава Ломакина

Издано с разрешения *No Starch Press, Inc., a California Corporation*

На русском языке публикуется впервые

Возрастная маркировка в соответствии
с Федеральным законом № 436-ФЗ: 6+

Морган, Ник

М79 JavaScript для детей. Самоучитель по программированию / Ник Морган ; пер. с англ. Станислава Ломакина ; [науч. ред. Д. Абрамова]. — М. : Манн, Иванов и Фербер, 2016. — 288 с.

ISBN 978-5-00100-295-6

Эта книга позволит вам погрузиться в программирование и с легкостью освоить JavaScipt. Вы напишете несколько настоящих игр — поиск сокровищ на карте, «Виселицу» и «Змейку». На каждом шаге вы сможете оценить результаты своих трудов — в виде работающей программы, а с понятными инструкциями, примерами и забавными иллюстрациями обучение будет только приятным. Книга для детей от 10 лет.

УДК 087.5:004.43
ББК 76.1,62:32.973.412

ISBN 978-5-00100-295-6

ОГЛАВЛЕНИЕ

ЧАСТЬ II. ПРОДВИНУТЫЙ JAVASCRIPT

ЧАСТЬ III. ГРАФИКА

Посвящается Филли
(и Оладушку)

ВВЕДЕНИЕ

Эта книга научит вас писать программы на JavaScript — одном из популярных языков программирования. А освоив язык программирования, вы станете программистом — человеком, который не просто *пользуется* компьютерами, а *управляет* ими. Научившись программированию, вы сможете вертеть компьютерами как хотите, и они всегда будут послушно следовать вашим указаниям.

Изучить именно JavaScript — отличная идея, потому что этот язык используется повсюду. Его поддерживают браузеры Chrome, Firefox и Internet Explorer. Возможности JavaScript позволяют программистам делать из обычных веб-страниц полноценные интерактивные приложения и видеоигры. Но это еще не все: JavaScript также работает на интернет-серверах и даже может использоваться для управления роботами и другими устройствами.

Для кого эта книга?

Эта книга предназначена для всех, кто хочет изучить именно JavaScript или же просто начать программировать с нуля. Она написана для детей, но может стать первым самоучителем по программированию для человека любого возраста.

Работая с книгой, вы будете постепенно узнавать новое, закреплять прочитанное и двигаться дальше и дальше. Начав с простых типов данных, вы перейдете к более сложным, по пути освоив управляющие конструкции и функции. После этого вы научитесь писать код, реагирующий на перемещения мышки или нажатия клавиш, и наконец познакомитесь с элементом `canvas`, который позволяет создавать рисунки и анимации — любые, какие только пожелаете!

Как читать эту книгу

Самое главное, читайте по порядку! Может быть, этот совет звучит странно, однако нередко людям не терпится сразу перейти к чему-нибудь занимательному, например к созданию игр. Но поверьте — вам будет гораздо проще создать игру, если вы все-таки будете читать с начала, глава за главой, так как каждый новый раздел основывается на материале предыдущих.

Языки программирования похожи на обычные языки. Вы, наверное, знаете — чтобы овладеть языком, нужно выучить грамматику и запомнить достаточно много слов. Это требует времени. Это же правило работает и с JavaScript — чтобы научиться пользоваться этим языком, нужно постоянно исследовать код и писать на нем программы. По мере того как вы будете писать больше и больше, вы обнаружите, что пользуетесь командами все более естественно, и в конце концов сможете свободно выражать свои мысли в коде.

Я настоятельно рекомендую тестировать примеры кода по мере чтения книги. Если вам не до конца понятно, как код работает, попробуйте вносить в него небольшие изменения и смотреть, как изменится результат. Если же ваши правки не приводят к ожидаемому эффекту, постарайтесь выяснить, почему это происходит.

Обязательно выполняйте задания из разделов «Попробуйте сами» и «Упражнения». Вводить в компьютер код из книги — отличное начало, но по-настоящему вы станете понимать программирование только тогда, когда начнете писать собственный код. Если задания покажутся вам интересными, не останавливайтесь! Придумывайте свои задачи по усовершенствованию написанных вами программ.

Вы можете найти примеры выполнения заданий и исходный код игр по адресу *www.nostarch.com/javascriptforkids* или на странице книги на сайте *www.mann-ivanov-ferber.ru*. Постарайтесь заглядывать в решения лишь после того, как выполните задания, чтобы сравнить свой подход с моим. И только если вы зашли в тупик, обратитесь за подсказкой. Однако помните, что это лишь *варианты* решения — в JavaScript существует множество способов выполнить одну и ту же задачу, так что не беспокойтесь, если ваше решение получится совсем не похожим на мое.

Если вы встретите слово, значение которого не понимаете, загляните в глоссарий в конце книги.

Что вас ждет?

Глава 1 содержит краткое введение в JavaScript. Кроме того, вы узнаете, как писать код в консоли Google Chrome.

Глава 2 расскажет про переменные и основные типы данных в JavaScript: числа, строки и булевы значения.

Глава 3 посвящена массивам, предназначенным для хранения наборов других элементов данных.

Глава 4 расскажет об объектах, содержащих пары «ключ-значение».

Глава 5 — это введение в HTML, язык для создания веб-страниц.

Глава 6 научит, как управлять выполнением кода с помощью конструкций `if`, циклов `for` и других структур.

Глава 7 покажет, как на основе изученного материала создать простую игру на отгадывание слов — «Виселицу».

Глава 8 научит писать собственные функции, что позволит группировать фрагменты кода и использовать их повторно.

Глава 9 — это введение в jQuery, инструмент, облегчающий управление веб-страницами из JavaScript-кода.

Глава 10 научит, как использовать таймеры, интервалы и обработчики событий, делая код более интерактивным.

Глава 11 использует функции, jQuery и обработчики событий для создания игры «Найди клад!».

Глава 12 научит элементам объектно-ориентированного программирования.

Глава 13 расскажет об элементе `canvas`, позволяющем создавать графические изображения на веб-страницах.

Глава 14 на основе способов анимации из главы 10 покажет, как создавать анимации на «холсте» `canvas`,

тогда как

Глава 15 научит, как управлять этими анимациями с клавиатуры.

В **главах 16 и 17** вы создадите полноценную игру «Змейка», используя все знания, полученные в предыдущих пятнадцати главах!

Послесловие подскажет, куда двигаться дальше при изучении программирования.

Глоссарий даст определения множества новых слов, которые вам встретятся.

Повеселитесь!

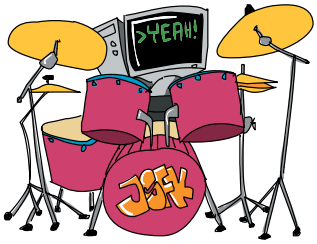
И еще один момент, о котором не стоит забывать: веселитесь! Программирование может быть увлекательным, творческим занятием, как рисование или игры (а работая с книгой, вы изрядно порисуете и поиграете). Как только вы научитесь программировать, для вас не будет иных преград, кроме собственного воображения. Добро пожаловать в потрясающий мир компьютерного программирования — и желаю вам отлично провести время!

ЧАСТЬ I

Основы

1

ЧТО ТАКОЕ JAVASCRIPT?



Компьютеры — необычайно мощные машины, способные делать потрясающие вещи. Например, они могут играть в шахматы, обслуживать тысячи интернет-страничек и менее чем за несколько секунд выполнять миллионы сложных вычислений. Однако сами по себе компьютеры неразумны, и делают они лишь то, что прикажут люди. Мы сообщаем компьютерам, что нам от них нужно, с помощью наборов инструкций, которые называются программами. Без программ компьютеры вообще ничего не умеют!

Знакомьтесь: JavaScript

Более того, компьютеры не знают ни английского, ни русского, ни других естественных языков; и компьютерные программы создают на специальных *языках программирования*. Одним из таких языков является JavaScript. Даже если вы слышите про JavaScript впервые, вы определенно заходили на сайты, которые его используют. Например, JavaScript может управлять внешним видом странички или делать так, чтобы страница реагировала на нажатие клавиши или перемещение мышки.

Такие сайты, как Gmail, Facebook и Twitter, используют JavaScript для облегчения работы с почтой, отправки комментариев или улучшения навигации. К примеру, когда вы, читая в Twitter сообщения от @nostarch, проматываете страничку вниз и видите все больше и больше сообщений, это происходит благодаря JavaScript.

Чтобы понять, чем же так хорош JavaScript, достаточно посетить несколько сайтов:

- JavaScript позволяет проигрывать музыку и создавать яркие визуальные эффекты. Например, вы можете полетать в интерактивном видеоклипе от студии HelloEnjoy на песню Элли Голдинг Lights (http://lights.helloenjoy.com/), рис. 1.1.

Рис. 1.1. В клипе Lights нужно управлять искрящимся курсором

- С помощью JavaScript можно создавать инструменты для творчества. Patatap (http://www.patatap.com/) — это нечто вроде виртуальной драм-машины, которая издает всевозможные шумы и звуки, а также проигрывает забавные анимации, рис. 1.2.

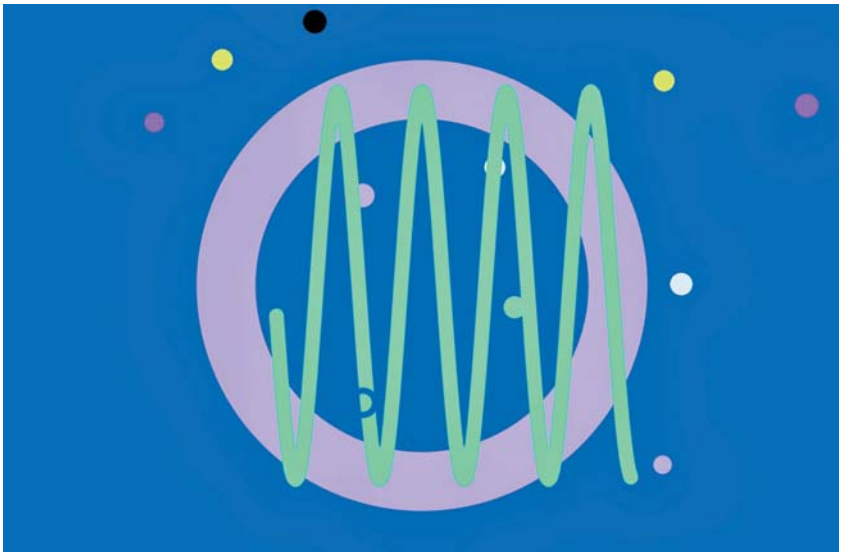


Рис. 1.2. Зайдя на страничку Patatap, нажимайте на разные клавиши, чтобы услышать разные звуки!

- JavaScript дает нам возможность играть в увлекательные игры. CubeSlam (https://www.cubeslam.com/) — это трехмерное подобие классической игры «Понг», похожее на аэрохоккей. Посоревнуйтесь с кем-нибудь из друзей или с медведем, за которого играет компьютер. См. рис. 1.3.

Рис. 1.3. Игра CubeSlam написана целиком на JavaScript!

Зачем изучать JavaScript?

JavaScript — далеко не единственный язык программирования. В сущности, языков очень много, счет идет на сотни, однако есть немало причин выбрать именно JavaScript. Например, изучать его гораздо проще (и интереснее), чем многие другие языки. Но, пожалуй, самая веская причина такова: чтобы писать и выполнять JavaScript-программы, достаточно интернет-браузера — такого, как Internet Explorer, Mozilla Firefox или Google Chrome. В каждый из этих браузеров встроен интерпретатор JavaScript, который сможет выполнять JavaScript-программы. И никакого специального программного обеспечения вам не понадобится.

Написав программу на JavaScript, отправьте ссылку на нее другим людям, и они тоже смогут ее запустить — у себя на компьютере, в браузере (см. «Обмен кодом с помощью JSFiddle» на с. 280).

Пробуем JavaScript

Давайте напишем простую JavaScript-программку с помощью браузера Google Chrome *(www.google.com/chrome)*. Установите Chrome на свой компьютер (если он еще не установлен), запустите его и введите слова about:blank в адресной строке. Теперь нажмите ENTER — откроется пустая страничка, как на рис. 1.4.

Начнем с программирования в JavaScript-консоли Chrome (это секретный инструмент для тестирования коротких программ на JavaScript). Если ваш компьютер работает под управлением Microsoft Windows или Linux, нажмите и не отпускайте клавиши CTRL и SHIFT, а затем нажмите J. Если же вы пользуетесь системой MacOS, нажмите и удерживайте COMMAND и OPTION, а затем нажмите J.

Если вы все сделали правильно, то увидите пустую веб-страницу, под которой стоит значок угловой скобки (>), а после него мигает курсор (|). Здесь нам и предстоит писать код на языке JavaScript!

Текст в консоли Chrome подсвечивается разными цветами в зависимости от типа данных. В этой книге код для ввода в консоль напечатан такими же цветами там, где это имеет значение. Но там, где разноцветный код будет вас только отвлекать, синим мы будем выделять то, что сами вводим в консоль, а данные, которые автоматически выдаст программа, будут цветными.

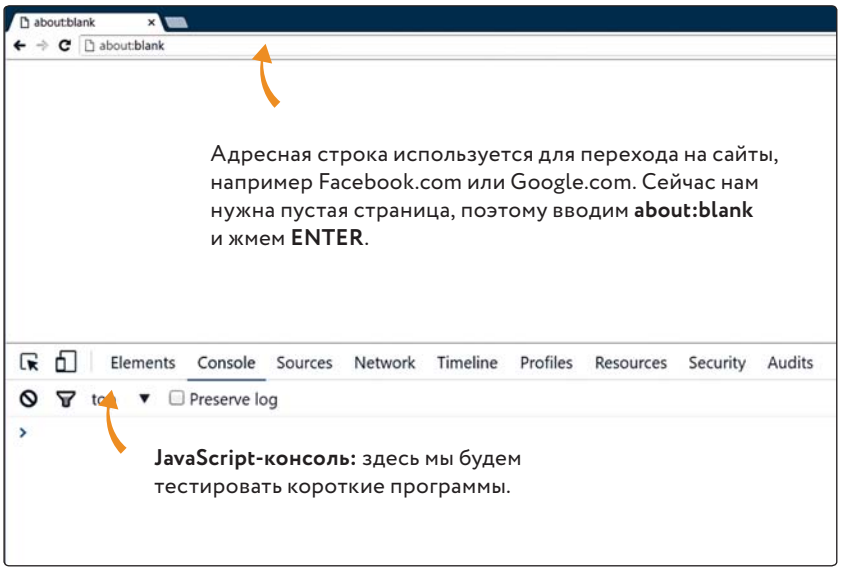


Рис. 1.4. JavaScript-консоль Google Chrome

Когда вы введете код и нажмете ENTER, JavaScript должен запустить (иначе говоря, *выполнить*) ваш код, показав на следующей строке результат (когда он есть). Например, введите в консоли:

```
3 + 4;
```

Теперь нажмите ENTER. JavaScript должен напечатать результат сложения (7) на следующей строке:

```
3 + 4;
7
```

Как видите, ничего сложного. Но JavaScript — это нечто определенно большее, чем просто затейливый калькулятор. Давайте попробуем кое-что еще.

Строение JavaScript-программы

Давайте позабавимся — напишем JavaScript-программу, которая печатает японские смайлики каомодзи в виде кошачьей мордочки:

```
=^.^=
```

В отличие от простого сложения, с которого мы начали, эта программа занимает несколько строк. Чтобы ввести ее в консоли, нужно будет в конце каждой строки переходить на новую строку **нажатием SHIFT-ENTER**. (Если нажать просто ENTER, Chrome попытается выполнить те команды, которые вы уже ввели, и программа не будет работать правильно. Сами по себе компьютеры ничего не соображают — я предупреждал!)

Введите в консоли браузера:

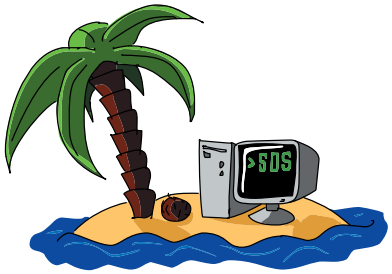
```
// Рисуем столько котиков, сколько захотим!
var drawCats = function (howManyTimes) {
  for (var i = 0; i < howManyTimes; i++) {
    console.log(i + " =^.^=");
  }
};
drawCats(10); // Вместо 10 тут может быть другое число
```

Draw cats — рисовать котиков

Function — функция

How many times — сколько раз

В конце последней строки нажмите ENTER, а не SHIFT-ENTER. Программа должна напечатать следующее:




```
0 =^.^=
1 =^.^=
2 =^.^=
3 =^.^=
4 =^.^=
5 =^.^=
6 =^.^=
7 =^.^=
8 =^.^=
9 =^.^=
```

Если при вводе программы вы где-то ошиблись, результат может оказаться другим — возможно, вы даже получите сообщение об ошибке. Это я и имел в виду, говоря, что компьютеры неразумны, — даже простейшая программа должна быть написана идеально, чтобы компьютер понял, что от него требуется!

Я не буду сейчас вдаваться в подробности, объясняя, как работает этот код (мы еще вернемся к нему в восьмой главе), однако давайте рассмотрим некоторые особенности этой программы, да и JavaScript-программ в целом.

Синтаксис

В нашей программе встречается много символов, таких как скобки (), точки с запятой ;, фигурные скобки {}, знаки плюс +, а также некоторые таинственные на первый взгляд слова (например, `var` и `console.log`). Все это является частью *синтаксиса* JavaScript — то есть правил, указывающих, как объединять символы и слова, чтобы составить работающую программу.

Одна из главных сложностей при освоении нового языка программирования — запомнить правила написания команд. Поначалу легко пропустить какие-нибудь скобки или запутаться в очередности записи значений. Не волнуйтесь, с опытом вы привыкнете писать код правильно.

В этой книге мы будем изучать материал медленно, постепенно знакомясь с новыми командами языка, чтобы вы могли писать все более и более мощные программы.

Комментарии

В первой строке нашей программы написано:

```
// Рисуем столько котиков, сколько захотим!
```

Это называется *комментарий*. Программисты пишут комментарии, чтобы другим программистам было легче читать и понимать их код. Компьютер же комментарии игнорирует. В JavaScript комментарии начинаются с двух символов наклонной черты (//). Все, что идет следом за ними (в той же строке), интерпретатор JavaScript пропускает, поэтому комментарии не оказывают влияния на выполнение программы — это всего лишь пояснение.

В примерах кода, которые встретятся вам в этой книге, комментарии описывают, что и как там происходит. При написании своего кода тоже добавляйте комментарии — когда вы заглянете в программу некоторое время спустя, они напомнят вам, как работает код и что происходит на том или ином этапе.

В конце нашей программы-примера есть еще один комментарий. Напоминаю: все, что записано после символов //, компьютер игнорирует!

```
drawCats(10); // Вместо 10 тут может быть другое число
```

Комментарии могут занимать отдельную строку или следовать сразу после кода. Но если вы поставите // перед кодом, вот так:

```
// drawCats(10);
```

...то не произойдет вообще ничего! Chrome решит, что вся эта строка — комментарий, хоть там и записаны инструкции на языке JavaScript.

Когда вы, помимо примеров в этой книге, начнете изучать чужой JavaScript-код, вам будут попадаться комментарии, которые выглядят иначе:

```
/*
Рисуем столько котиков,
сколько захотим!
*/
```

Это другая разновидность комментариев; их обычно используют, когда текст примечания не помещается на одной строке. Однако принцип здесь тот же: текст, записанный между /* и */, — это комментарий, и выполнять его компьютер не будет.

Что мы узнали

В этой главе мы познакомились с языком JavaScript и узнали, что можно делать с его помощью. Кроме того, мы научились запускать JavaScript-код в браузере Google Chrome и ввели несложную программу-пример. Все примеры из этой книги можно (и нужно!) запускать в JavaScript-консоли Google Chrome (если только я не скажу, что этого делать не надо). Просто читать код недостаточно — проверяйте, как он работает! Это единственный способ научиться программировать.

В следующей главе мы приступим к изучению основ языка JavaScript, начиная с трех основных типов данных, с которыми вам предстоит работать: чисел, строк и булевых значений.

2

ТИПЫ ДАННЫХ И ПЕРЕМЕННЫЕ

Программирование — это работа с данными, но что такое данные? Данные — это информация, которая хранится в наших компьютерных программах. Например, ваше имя — это элемент данных, и ваш возраст тоже. Цвет волос, количество братьев и сестер, ваш адрес и пол — все это данные.

В JavaScript есть три основных типа данных: числа, строки и булевы значения. Числа — они и есть числа, тут все понятно. Например, числом можно выразить возраст или рост.

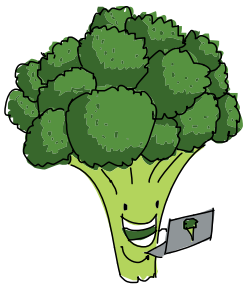
В JavaScript числа записываются так:

```
5;
```

Любые текстовые данные записываются в строки. В JavaScript ваше имя можно выразить строкой (так же как и адрес вашей электронной почты).

Строки выглядят так:

```
"Привет, я строка";
```



Булевы значения могут хранить одну из двух величин — либо это `true` («истина»), либо `false` («ложь»). Например, таким способом можно показать, носите ли вы очки или любите ли вы брокколи.

Пример булева значения:

```
true;
```

С данными разных типов и обращаться следует по-разному. Например, перемножить два числа можно, а перемножить две строки — нет. Зато, имея строку, можно выделить пять ее первых символов. Взяв два булевых значения, можно проверить, являются ли они оба «истиной» (`true`). Вот все эти действия на примере:

```
99 * 123;
12177
"Вот длинная строка".slice(0, 3);
"Вот"
true && false;
false
```

Любые данные в JavaScript — не более чем сочетание этих основных типов. Далее мы по очереди рассмотрим каждый тип данных и изучим различные способы работы с ними.

Наверное, вы заметили, что все эти команды оканчиваются на точку с запятой (;). Этим символом обозначают конец каждой отдельной команды или инструкции языка JavaScript — примерно так же, как точка отмечает конец предложения.

Числа и операторы

JavaScript позволяет выполнять основные математические операции, такие как сложение, вычитание, умножение и деление. Для их записи используются символы +, –, * и /, которые называют *операторами*.

Консоль JavaScript можно использовать как калькулятор. Один из примеров — сложение 3 и 4 — нам уже знаком. Давайте вычислим что-нибудь посложнее: сколько будет 12345 плюс 56789?

```
12345 + 56789;
69134
```

Посчитать это в уме не так уж просто, а JavaScript мгновенно справился с задачей.

Можно сложить несколько чисел с помощью нескольких знаков «плюс»:

```
22 + 33 + 44;
99
```

Также JavaScript умеет вычитать...

```
1000 - 17;
983
```

умножать (с помощью символа «звездочка»)...

```
123 * 456;
56088
```

и делить (с помощью косой черты — слэша)...

```
12345 / 250;
49.38
```

Кроме того, можно объединять эти простые операции, составляя более сложные выражения, вроде такого:

```
1234 + 57 * 3 - 31 / 4;
1397.25
```

Есть один нюанс — результат вычислений зависит от порядка, в котором JavaScript выполняет отдельные операции. В математике существует правило, по которому умножение и деление выполняются прежде, чем сложение и вычитание, и JavaScript ему следует.

Порядок, в котором интерпретатор JavaScript выполняет эти операции, показан на рис. 2.1. Сначала он умножает 57 * 3, получая 171 (выделено красным). Затем делит 31 / 4, получая 7.75 (выделено синим). Затем складывает 1234 + 171, получая 1405 (выделено зеленым). И наконец, вычитает 1405 – 7.75, что дает 1397.25 — окончательный результат.

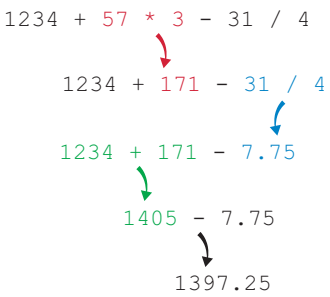


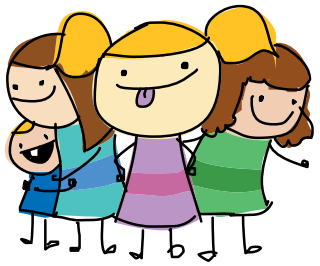
Рис. 2.1. Очередность выполнения операций: умножение, деление, сложение, вычитание

Но как быть, если вы хотите выполнить сложение и вычитание до умножения и деления? Для примера предположим, что у вас есть 1 брат, 3 сестры и 8 карамелек, которые

вы решили поровну разделить между ними (свою долю вы уже прикарманили). Нужно разделить 8 на общее количество братьев и сестер.

Попытаемся это сделать:

```
8 / 1 + 3;
11
```



Это не может быть верным ответом! Не получится дать каждому родственнику по 11 карамелек, если у вас всего-то 8 конфет! Проблема в том, что JavaScript выполняет деление прежде, чем сложение, то есть он делит 8 на 1 (что равно 8) и затем прибавляет 3, получая в результате 11. Чтобы исправить эту ошибку, заставим JavaScript сначала выполнить сложение, воспользовавшись *скобками*:

```
8 / (1 + 3);
2
```

Так гораздо лучше — вышло по две карамельки каждому из родственников. Скобки вынудили JavaScript сложить 1 и 3 *до* деления 8 на 4.

ПОПРОБУЙТЕ!

Предположим, ваша подруга пытается подсчитать с помощью JavaScript, сколько ей нужно купить воздушных шаров. Она устраивает вечеринку и хочет, чтобы каждый из гостей смог надуть по 2 шарика. Сначала было приглашено 15 человек, но потом ваша подруга позвала еще 9.

Она написала такой код:

```
15 + 9 * 2;
33
```

Однако ответ, судя по всему, неверен.

Где надо поставить скобки, чтобы JavaScript сначала складывал, а потом умножал, и сколько шариков нужно вашей подруге на самом деле?

Переменные

Значениям в JavaScript можно давать имена, используя *переменные*. Переменная похожа на ящичек, в который помещается лишь один предмет. Чтобы положить туда что-то еще, прежнее содержимое придется заменить.

Чтобы создать новую переменную, используйте ключевое слово `var`, после которого укажите имя переменной. Ключевое слово — это слово,

обладающее для JavaScript особым значением. В данном случае, когда JavaScript встречает слово `var`, он понимает, что следом указано имя новой переменной. Например, вот как создать переменную с именем nick:

```
var nick;
undefined
```

Undefined — значение не определено

Мы создали новую переменную под названием nick. В ответ консоль выдала `undefined` — «значение не определено». Однако это не ошибка! JavaScript всегда так делает, если команда не возвращает какого-либо значения. Вы спросите, а что такое «возвращать значение»? Вот пример: когда вы ввели `12345 + 56789;`, консоль вернула значение 69134. Однако в JavaScript команда создания переменной никакого значения не возвращает, поэтому интерпретатор печатает `undefined`.

В этом примере и дальше мы будем давать переменным англоязычные имена, потому что английский — основной язык всей IT-области и программы принято писать только латиницей (кроме комментариев и строковых значений). Использовать русскоязычные имена переменных — это как если при составлении математических уравнений вместо *x* и *y* вы использовали бы русские буквы. Можно, но не принято.

Итак, чтобы задать переменной значение, используйте знак «равно»:

```
var age = 12;
undefined
```

Age — возраст

Задание значения переменной называют *присваиванием* (здесь мы присваиваем значение 12 переменной `age`). И опять в консоли появляется `undefined`, поскольку мы только что создали новую переменную. (В дальнейших примерах я буду пропускать это `undefined`.)

Теперь в интерпретаторе есть переменная `age`, которой присвоено значение 12. И если ввести в консоли имя `age`, интерпретатор выдаст значение этой переменной:

```
age;
12
```

Здорово! При этом значение переменной не высечено в камне (*переменные* потому так и зовутся, что могут *менять* значения), и, если вам вздумается его обновить, просто используйте знак «равно» еще раз.

```
age = 13;
13
```

На этот раз я не использовал ключевое слово `var`, поскольку переменная `age` уже существует. Писать `var` нужно только при *создании* переменной, а не при ее использовании. И обратите внимание: поскольку мы не создавали новой переменной, команда присваивания вернула значение 13, которое и было напечатано в следующей строке.

Вот чуть более сложный пример — решение задачи про карамельки без помощи скобок:

Number of siblings — число братьев и сестер

Number of candies — число конфет

```
var numberOfSiblings = 1 + 3;
var numberOfCandies = 8;
numberOfCandies / numberOfSiblings;
2
```

Сначала мы создали переменную с именем `numberOfSiblings` (количество братьев и сестер) и присвоили ей значение выражения 1 + 3 (которое JavaScript вычислил, получив 4). Потом мы создали переменную `numberOfCandies` (количество карамелек) и присвоили ей значение 8. И наконец, мы ввели: `numberOfCandies / numberOfSiblings`. Поскольку переменная `numberOfCandies` содержит значение 8, а `numberOfSiblings` — 4, JavaScript вычислил, сколько будет 8 / 4, вернув в результате 2.

Имена переменных

Вводя имена переменных, будьте внимательны и не допускайте опечаток. Даже если вы перепутаете строчные и заглавные буквы, интерпретатор JavaScript не поймет, чего вы от него хотите! Например, если вы случайно введете имя `numberOfCandies` со строчной буквой *c*, возникнет ошибка:

Reference error — ошибка данных

```
numberOfcandies / numberOfSiblings;
ReferenceError: numberOfcandies is not defined
```

Увы, JavaScript следует вашим указаниям буквально. Если вы неправильно ввели имя переменной, JavaScript не поймет, что вы имели в виду, и выдаст сообщение об ошибке.

Еще один нюанс именования переменных в JavaScript — в именах не должно быть пробелов, из-за чего они могут оказаться сложными для чтения. Если бы я назвал переменную `numberofcandies`, без заглавных букв, читать программу стало бы труднее, поскольку неясно, где в этом имени заканчиваются отдельные слова.

Один из обычных способов решения этой проблемы — писать каждое слово с заглавной буквы: `NumberOfCandies`. Такую манеру именования