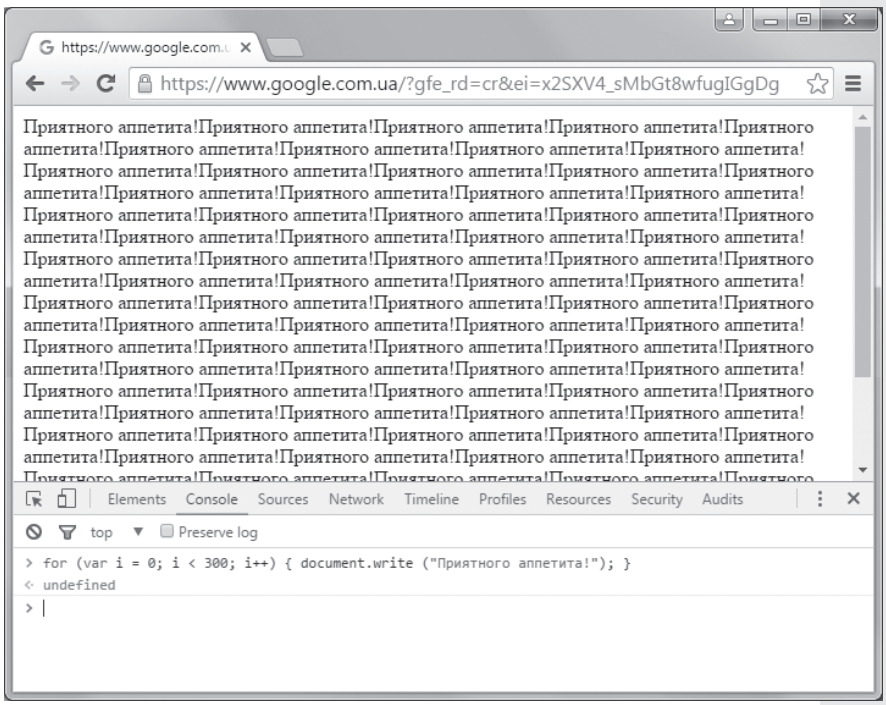
ГЛАВА

2

Синтаксис JavaScript

Как и языки, на которых мы разговариваем, языки программирования имеют свой синтаксис, определяющий правила написания на них программ. Как только вы освоите принципы написания программ на JavaScript, он перестанет казаться необычным, поскольку синтаксически подобен английскому языку.



Если вам кажется, что учитель английского языка придирается к вам из-за каждой погрешности в написании слов, то не спешите с выводами. Это вы еще не знакомы с требованиями к синтаксису JavaScript. Синтаксические ошибки в коде JavaScript непростительны, так как вызывают сбои в работе программы.

В этой главе вы познакомитесь с синтаксисом языка программирования JavaScript и научитесь избегать опечаток при написании кода программ.

Точность — вежливость королей

Поскольку компьютеры не приемлют двусмысленности, инструкции для них должны быть предельно четкими и однозначными.

В главе 1 было введено понятие программы и описано, как набор инструкций преобразуется в машинный код, понятный компьютеру. Как вы уже знаете, эта операция называется компиляцией.

В задачи программиста, кроме всего прочего, входит разбиение кода на отдельные фрагменты, каждый из которых предельно просто проверить на наличие ошибок. Например, являясь счастливым обладателем домашнего робота, можно дать ему указание спуститься на первый этаж и приготовить кофе. Чтобы удостовериться в правильности выполнения роботом программы по приготовлению бодрящего напитка, все его действия нужно представить точными инструкциями. Ниже приведен один из возможных вариантов.

1. Повернуть голову в направлении лестницы.
2. С помощью датчиков распознать объекты, преграждающие движение в выбранном направлении.
3. Оценить преграды.
4. Если преграды живые (домашние животные), то согнать их с пути следования.
   - Прогнать звуковым сигналом.
   - Бросить любимую игрушку в сторону от пути следования.
   - Подтолкнуть преграду в требуемом направлении рукой-манипулятором.
5. Если преград нет или они устранены, то начать движение в направлении лестницы.
6. Сделав шаг, еще раз оценить преграды на пути.
7. Если преград нет, сделать еще один шаг.
8. Снова оценить, свободен ли путь.
9. Дойти до начала лестницы.

Как видите, только на описание действий робота, необходимых для перемещения к началу лестницы, потребовалось 9 инструкций. На самостоятельное приготовление кофе уходит намного меньше времени!

Настоящая компьютерная программа, отвечающая за перемещение робота к лестнице, содержит намного больше инструкций, чем представлено в приведенном выше списке. На каждом этапе необходимо указывать продолжительность работы каждого из сервоприводов, обрабатывать данные, поступающие с датчиков, а также принимать решения о том, как поступить с найденными преградами. И это далеко не все операции! Детализации действий нет предела.

Все указания для робота в программе JavaScript записываются в виде специальных команд, именуемых *инструкциями*.

Детально примеры управления робототехническими устройствами с помощью JavaScript рассмотрены на сайте `http://nodebots.io`.

Инструкции

В нашем языке речь записывается в виде предложений. В языке программирования JavaScript предложениям обычного языка сопоставляются *инструкции*. Как и предложения, инструкции состоят из отдельных элементов, расположение которых регулируется специальными правилами. Если их не придерживаться, то компьютер попросту не поймет программу.

Пример инструкции приведен в листинге 2.1.

Листинг 2.1. Пример инструкции JavaScript

```
alert("Программировать весело!");
```

Эта инструкция указывает браузеру вывести на экран сообщение, содержащее текст `Программировать весело!`. Если ввести эту инструкцию на консоли JavaScript браузера Chrome, то можно увидеть извещение, показанное на рис. 2.1.

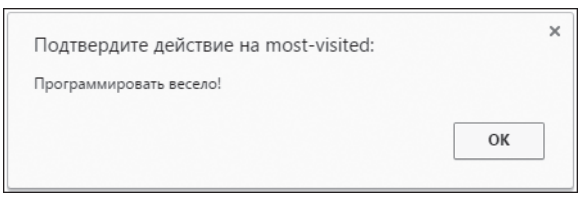


Рис. 2.1. Результат выполнения одной из инструкций JavaScript

Легко заметить, что инструкция из листинга 2.1 состоит из ключевого слова, специальных символов (кавычек и скобок) и произвольного текста (`Программировать весело!`). Заканчивается инструкция точкой с запятой.

В JavaScript вас никто не ограничивает количеством инструкций в программе. Как и в художественном произведении, написанном на обычном языке, в программе JavaScript может содержаться произвольное количество строк кода.

Приведенная выше инструкция начинается с ключевого слова `alert`, известного JavaScript. На самом деле многие инструкции начинаются с ключевых слов, но это не обязательное правило.

Инструкции разделяются точкой с запятой. В обычных текстах роль точки с запятой играет точка, которой заканчивается большинство предложений. В языке программирования JavaScript точкой с запятой заканчиваются все без исключения инструкции.

Синтаксические правила

Чтобы компьютер понимал введенный вами код JavaScript, при его написании необходимо соблюдать определенные правила. Ниже указаны наиболее важные из них.

- ✓ Ключевые слова JavaScript строго заданы.
- ✓ Пробелы не учитываются.

Давайте детально рассмотрим каждое из них. Проще всего это сделать на примере программы, выводящей на экран 300 уже известных вам извещений `Программировать весело!`. Ее код показан в листинге 2.2.

Листинг 2.2. Программа вывода на экран 300 сообщений

```
for (var i = 0; i < 300; i++) { document.write ("Программировать
   ↳весело!"); }
```

Чтобы запустить эту простую программу, выполните следующие действия.

1. Запустите браузер Google Chrome.

2. Отобразите в окне браузера консоль JavaScript (воспользовавшись панелью разработки).

   Для быстрого добавления в окно Chrome консоли JavaScript нажмите комбинацию клавиш <⌘+Option+J> в Mac или <Ctrl+Shift+J> в Windows.

3. Введите в одну строку код программы, показанной в листинге 2.2. Нажмите <Return> в Mac или <Enter> в Windows.

   Если все введено правильно, то в окне браузера появится множество (точнее — 300) одинаковых сообщений, как показано на рис. 2.2.

В данном коде основная часть работы выполняется в специальной программной структуре, известной под названием *цикл* `for`. Она позволяет выполнить одни и те же действия строго заданное количество раз. Подробно о возможностях циклических структур в JavaScript рассказывается в главах 17 и 18.

Еще раз внимательно изучите код листинга 2.2. Легко заметить, что текст сообщения `Программировать весело!` заключен в кавычки. Они указывают на то, что текст в них стоит рассматривать именно как текст, а не как ключевые слова JavaScript.

Текстовые строки

В программировании текст, заключенный в кавычки, называется строкой или строковым значением. Справедливое название, если учесть, что все символы, заключенные в кавычки, выводятся на экран последовательно и в том порядке, в котором они введены.

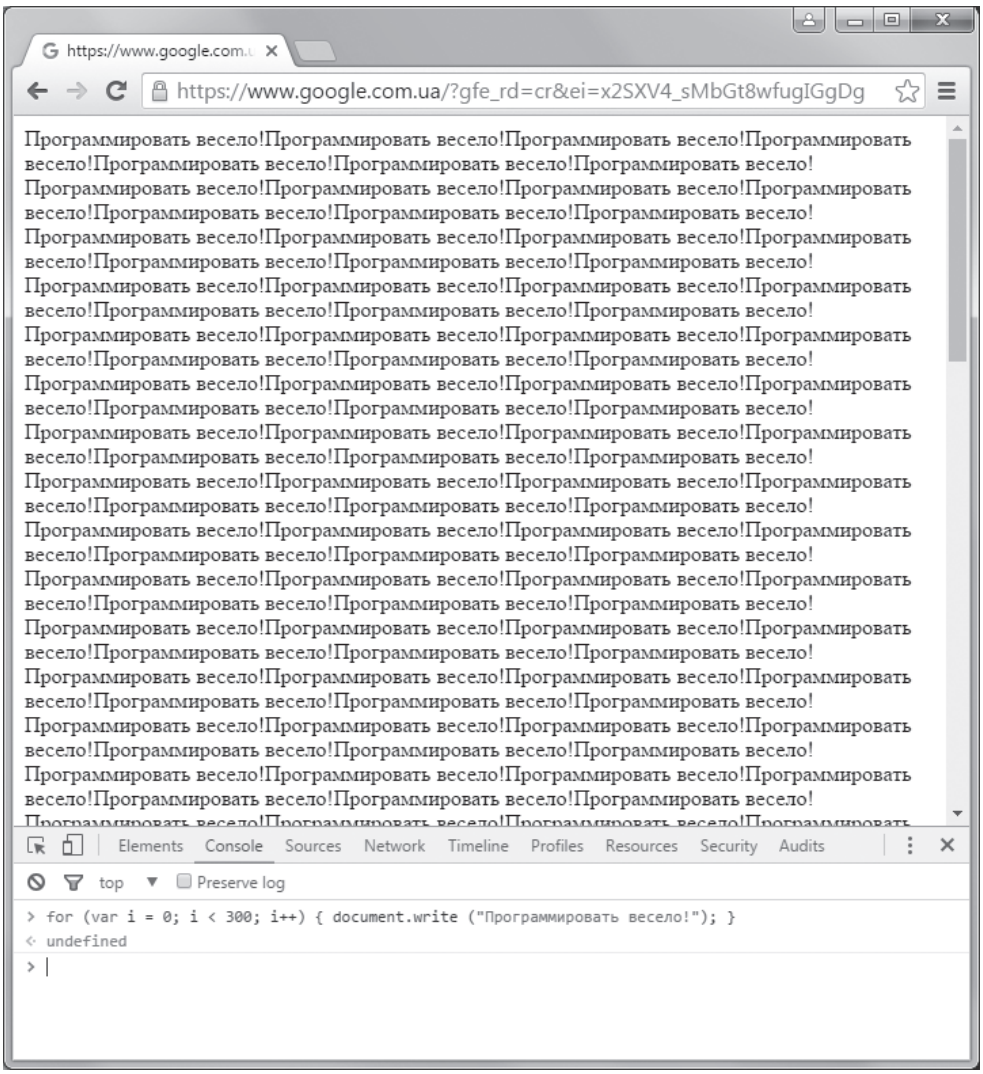


Рис. 2.2. Результат выполнения кода, приведенного в листинге 2.2

Изменить строковое значение в JavaScript очень просто. Попробуйте в примере, приведенном в листинге 2.2, заменить строковое значение `Программировать весело!` любым другим текстом, например пожеланием приятного аппетита. Выполнив измененный указанным образом код, вы получите на экране новые сообщения.

Как видно на рис. 2.3, теперь программа из листинга 2.2 выводит в окне браузера извещения `Приятного аппетита!`.

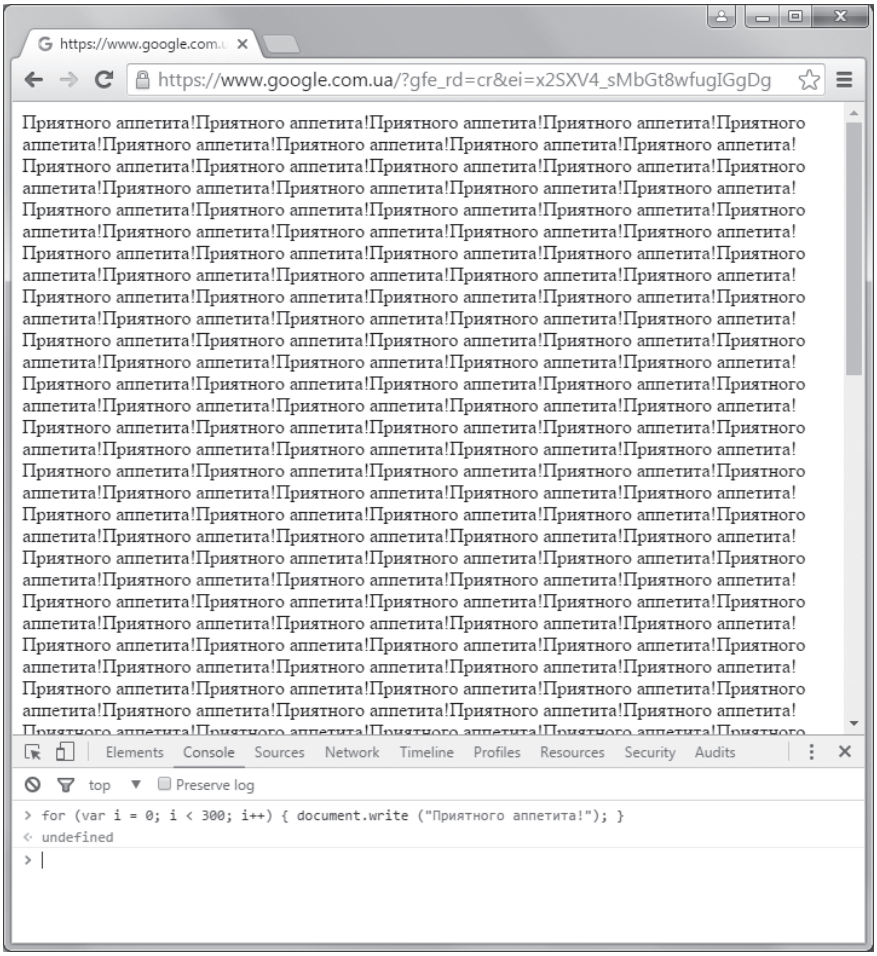


Рис. 2.3. Изменение текста строки приводит к выводу совершенно иного сообщения

В строковом значении сохраняются все символы, заключенные в кавычки. В подобном синтаксисе команды существует определенное противоречие: как поступить, если кавычки нужно включить в выводимое на экран сообщение? Должен же быть способ разграничения кавычек, включаемых в сообщение, и символов, определяющих его границы!

Для обозначения кавычек, выступающих в качестве части строкового значения, используется символ обратной косой черты (\). Он ставится перед кавычками и указывает на то, что следующий символ относится к тексту сообщения, а не к ключевому слову JavaScript. Существует целый ряд специальных знаков, включение которых в строковое значение требует добавления перед ними символа обратной косой черты. Кавычки — всего лишь один из них.

Например, для вывода на экран сообщения

```
При встрече мы говорим "Привет!"
```

в рассмотренном выше коде необходимо ввести следующее строковое значение.

```
"При встрече мы говорим \"Привет!\""
```

Полный код инструкции приведен в листинге 2.3.

Листинг 2.3. Вывод в сообщении символа кавычек

```
for (var i = 0; i < 300; i++) { document.write ("При встрече мы
   ↳говорим \"Привет!\""); }
```

А как поступить, если в текст сообщения необходимо включить обратную косую черту? Существует ли вообще возможность представления ее в качестве текстового символа? Конечно же, да! Выглядит странно, но для выполнения этой задачи перед символом обратной косой черты необходимо ввести еще один символ обратной косой черты (\\).

Как и многое другое в JavaScript, существует альтернативный способ включения символов кавычек в строку. Как вы знаете, кавычки бывают одинарные (') и двойные ("). В JavaScript разрешается использовать и те, и другие, главное, чтобы в начале строки и ее конце использовались кавычки одного типа.

Заключив строковое значение в одинарные кавычки, вы сможете включать в нее двойные кавычки, не прибегая к использованию символов обратной косой черты. Если при этом необходимо включить в текст сообщения одинарные кавычки, то символ обратной косой черты перед ними вводится в обязательном порядке.

И наоборот, обозначив строку двойными кавычками, можно не вводить обратную косую черту перед одинарными кавычками в самом сообщении, чего не скажешь о двойных кавычках.

Пример одновременного использования в коде JavaScript кавычек обоих типов приведен в листинге 2.4. Обратите внимание на то, что в тексте сообщения, выводимом на экран, содержатся двойные кавычки.

Листинг 2.4. Совместное применение кавычек обоих типов

```
for (var i = 0; i < 300; i++) { document.write ('При встрече мы
   ↳говорим "Привет!"'); }
```

Ввод ключевых слов

В отличие от строк, весь остальной код записывается по строгим правилам. Код, расположенный вне одинарных или двойных кавычек, относится к ключевым словам JavaScript и определяет операции, выполняемые программой.

При вводе ключевых слов JavaScript очень важно не допускать опечаток и сохранять исходный регистр символов. Как несложно заметить, следующие выражения распознаются в JavaScript по-разному.

```
FOR
for
For
```

Только второе слово относится к ключевым словам JavaScript. Если вы попытаетесь ввести в коде программы остальные два варианта написания, то получите сообщение об ошибке, подобное показанному на рис. 2.4.

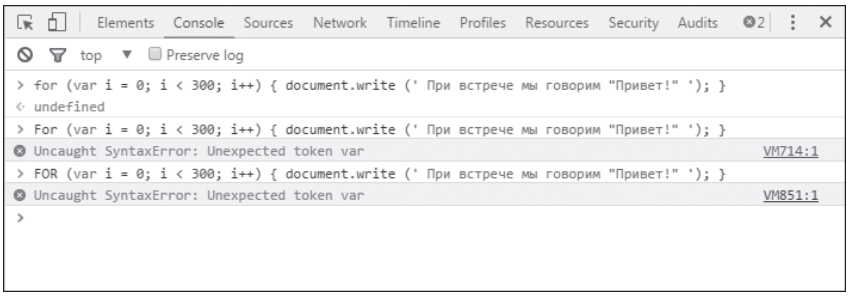


Рис. 2.4. Использование прописных символов во многих ключевых словах JavaScript приводит к возникновению ошибок

Подробно ключевое слово `for` и его назначение в коде JavaScript будут рассмотрены в главе 17.

При вводе кода JavaScript будьте предельно аккуратны. Во многих случаях синтаксические ошибки в программе вызваны опечатками в ключевых словах (чаще всего пропущена буква или буквы следуют в неправильном порядке).

Как и ошибки в школьных сочинениях, опечатки в коде JavaScript очень сложно отследить на этапе создания программы. Возьмите себе за правило никогда не торопиться при вводе ключевых слов. По окончании набора всегда проверяйте каждую инструкцию на наличие синтаксических ошибок. Тем самым вы сэкономите огромное количество времени и сил, которые еще понадобятся вам при поиске причин более серьезных неполадок.

Пробелы в коде

Давайте договоримся понимать под пробелами любые интервалы между словами и символами, в том числе отступы и разрывы строк. Как уже упоминалось, в коде JavaScript пробелы полностью игнорируются. В рассмотренном выше примере вывода сообщений в окне браузера использование отступов и пробелов в сочетании с разбивкой кода на отдельные строки позволяет лучше понять общую структуру программы (листинг 2.5).

Листинг 2.5. Пробелы и структурирование программы

```
for (var i = 0; i < 300; i++) {
   document.write ("Программировать весело!");
}
```

В листинге 2.5 показан только один из способов структурирования программы. Это предложенный нами вариант, не претендующий на абсолютную правильность.

Обратите внимание на то, что после открывающей фигурной скобки ({) добавлен разрыв строки. Еще один разрыв строки находится перед закрывающей фигурной скобкой (}). Обычно фигурные скобки используются для обособления некой части кода (группы инструкций) в отдельный *блок*. В нашем случае в таком блоке содержится только одна инструкция, отвечающая за вывод на экран повторяющегося 300 раз сообщения.

Фигурные скобки всегда указывают на место в коде, где не лишне добавить пробел, отступ или разрыв строки. Еще один символ, после которого гарантированно ставится разрыв строки, — это точка с запятой, обозначающая конец инструкции.

Если начать вводить показанный в листинге 2.5 код на консоли JavaScript и в конце первой строки (после открывающей фигурной кавычки) нажать клавишу <Return> (в Mac) или <Enter> (в Windows), то будет возвращена ошибка. А все потому, что нажатие <Return> (<Enter>) указывает выполнить введенный выше код. Разумеется, выполнение только первой строки программы не позволяет получить значимый результат. Чтобы вместо выполнения строки добавить в ее конец разрыв, нужно вместе с <Enter> нажимать клавишу <Shift>. Этой же комбинацией клавиш воспользуйтесь и в конце второй строки.

Строки внутри блока кода, заключенного в фигурные кавычки, вводятся с отступом. Отступами в коде обычно выделяются инструкции, выполняемые внутри других инструкций. Поскольку инструкция `for` образует циклическую структуру, отступами в нем "отбиваются" все вложенные (повторно выполняемые) инструкции.

Для добавления в начале строки отступа лучше всего использовать клавишу пробела. Стандартный отступ образуется при нажатии клавиши пробела два или четыре раза. Некоторые программисты предпочитают добавлять отступы с помощью клавиши <Tab>. Здесь каждый решает сам. Главное, чтобы отступы были одинаковыми во всей программе. Если вы с первых строк устанавливаете отступы двумя символами пробела, то не стоит дальше в коде переходить на использование отступов шириной в четыре пробела или один символ табуляции.

Комментарии

Комментарии в программе не относятся ни к инструкциям, ни к выводимому на экран тексту. Это может показаться странным, но зачастую комментарии играют в программе более значимую роль, чем весь остальной код. Комментарии не содержат инструкций, что не мешает им представлять особую важность.

Обычно программисты снабжают свой код комментариями в следующих случаях.

- ✓ Программа подлежит дальнейшей модификации или исправлению. Тот, кому предстоит заниматься этим в дальнейшем, будет бесконечно благодарен тому, кто снабдил все выполняемые программой действия подробным описанием.
- ✓ В коде программы задействованы специальные или нестандартные методики выполнения операций.
- ✓ Программу все еще можно улучшить, несмотря на то что она выполняет возложенные на нее задачи. В комментарии добавляются сведения о том, как лучше всего совершенствовать имеющийся код.
- ✓ Отдельные инструкции нужно временно заблокировать для выполнения.

Комментарии бывают двух типов: вводимые в одну или в несколько строк.

- ✓ **Однострочный комментарий.** Комментарием в коде объявляется сразу вся строка. Для создания однострочного комментария в начало строки помещаются два подряд символа косой черты (//). Например, в листинге 2.6 однострочными комментариями объявлены первые три строки программы. Выполняющая действие инструкция содержится в единственной, четвертой по счету, строке.

Листинг 2.6. Однострочные комментарии

```
// Следующая инструкция не выполняется
// alert("Поберегись!");
// Приведенный далее код выполняется
alert("Хорошего настроения!"); // приятное пожелание
```

- ✓ **Многострочный комментарий.** Многострочным такой комментарий называется потому, что располагается сразу в нескольких (иногда во многих) строках кода. На начало многострочного комментария указывает оператор /*. Завершается такой комментарий оператором */. Пример многострочного комментария показан в листинге 2.7.

Листинг 2.7. Многострочный комментарий

```
/*
   Программа AlertMe разработана Крисом Минником и Евой Холланд.

   Программа извещения пользователей об использовании
   ими языка программирования JavaScript называется AlertMe.
   Она разработана Крисом Минником и Евой Холланд.
*/
```