

УДК 004.43
ББК 32.973.26-018.2
В19

Васильев, Алексей Николаевич.
В19 JavaScript в примерах и задачах / Алексей Васильев. – Москва :
Издательство «Э», 2017. – 720 с. – (Российский компьютерный бес-
тселлер).

ISBN 978-5-699-95459-9

Простой и интересный самоучитель по JavaScript, наиболее популярному сегодня языку программирования во всем мире. Полный спектр сведений о языке JavaScript с примерами и разбором задач от автора учебников-бестселлеров по языкам программирования Алексея Васильева. С помощью этой книги освоить язык JavaScript сможет каждый желающий – от новичка до специалиста.

УДК 004.43
ББК 32.973.26-018.2

ISBN 978-5-699-95459-9

© **Васильев А.Н., 2017**
© **Оформление. ООО «Издательство «Э», 2017**

ОГЛАВЛЕНИЕ

Вступление. Книга о JavaScript	7
Сценарии и программы	8
Веб-документы и язык HTML	11
Добавление сценария в веб-документ	18
Концепция книги	25
Тестирование сценариев и программное обеспечение	28
Обратная связь	39
Об авторе	39
Благодарности	40

ЧАСТЬ I. ОСНОВЫ JAVASCRIPT

Глава 1. Знакомство с JavaScript	42
Отображение текста в рабочем документе	43
Способы реализации сценария	46
Знакомство с переменными	49
Сценарий с одной переменной	51
Сценарий с двумя переменными	53
Присваивание переменной значений разных типов	54
Вычисление выражений	56
Основные операторы	58
Арифметические операторы	58
Операторы сравнения	62
Логические операторы	63
Побитовые операторы	67
Сокращенные формы оператора присваивания	73
Тернарный оператор	74
Преобразование типов	75
Приоритет операторов	77
Резюме	79
Глава 2. Управляющие инструкции	80
Условный оператор	80
Общий синтаксис условного оператора	80
Пример с условным оператором	82
Упрощенная форма условного оператора	87
Пример с упрощенной формой условного оператора	88
Вложенные условные операторы	90
Оператор цикла while	98
Синтаксис оператора	98
Пример использования оператора цикла	99

Оператор цикла do-while	102
Синтаксис оператора	102
Пример с использованием оператора цикла	104
Оператор цикла for	106
Синтаксис оператора	106
Пример использования оператора	108
Оператор выбора switch	113
Синтаксис оператора выбора	113
Примеры использования оператора выбора	115
Резюме	123
Глава 3. Функции	125
Знакомство с функциями	125
Описание функции	126
Примеры объявления функций	127
Локальные и глобальные переменные	131
Область видимости и локальные переменные в функции	131
Обращение к глобальной переменной в функции	135
Создание глобальной переменной в функции	137
Глобальная переменная как свойство объекта окна	139
Аргументы функции	146
Аргумент как локальная переменная	146
Механизм передачи аргументов функции	149
Проверка типа аргумента	152
Количество аргументов функции	156
Передача функции аргументом	160
Рекурсия	164
Внутренние функции	167
Присваивание функций	174
Анонимные функции	179
Функция как результат	186
Резюме	193

ЧАСТЬ II. JAVASCRIPT И ООП

Глава 4. Знакомство с объектами и принципы ООП	196
Концепция ООП	196
Создание объектов	198
Литерал объекта	198
Объект с методом	203
Присваивание объектов	206
Добавление свойств и методов в объект	209
Конструктор объектов	211
Утилиты для работы с объектами	215
Оператор with	215
Оператор for-in	217

Оператор in	220
Оператор delete и удаление свойств и методов	223
Прототипы	227
Механизм доступа к свойствам и создание объекта на основе прототипа	228
Получение доступа к прототипу	238
Создание объектов без прототипа	247
Конструкторы и прототипы	250
Свойства и методы	258
Перечисляемые и неперечисляемые свойства	258
Свойства с режимом доступа	269
Резюме	280
Глава 5. Знакомство с массивами	283
Создание массива	283
Явное указание элементов массива	283
Добавление элементов в массив	286
Использование объекта-конструктора Array	288
Операции с массивами	295
Методы для работы с массивами	295
Присваивание и копирование массивов	311
Методы toString() и valueOf()	323
Двумерные массивы	328
Резюме	333
Глава 6. Использование объектов	334
Обработка исключительных ситуаций	334
Инструкция try-catch	335
Объект ошибки	339
Генерирование ошибок	346
Вложенные try-catch блоки и блок finally	351
Создание ошибки пользовательского типа	357
Объекты и массивы	362
Объект как ассоциативный массив	362
Методы toString() и valueOf() для объектов	367
Массивы и объекты как свойства объекта	374
Функция как объект	383
Количество аргументов функции	384
Функция с произвольным количеством аргументов	388
Передача контекста функции	391
Встроенные объекты	405
Объект Math	405
Объект Number	407
Объект Boolean	410
Объект String	412
Объект Date	417
Резюме	427

ЧАСТЬ III. ИСПОЛЬЗОВАНИЕ JAVASCRIPT

Глава 7. Веб-документы и сценарии	430
Место и роль сценария в веб-документе	430
Размещение сценария в документе	430
Обработка событий	438
Объект окна window	440
Объектные модели	440
Диалоговые окна	442
Открытие и закрытие окна	448
Загрузка документа	455
Свойства и методы объекта window	461
Таймеры	481
Объект документа document	502
Свойства и методы объекта document	502
Настройки цвета	512
Методы write() и writeln()	515
Программное создание документа	518
Резюме	529
Глава 8. Элементы управления и обработка событий	530
Элементы управления в веб-документе	530
Кнопки и поля ввода	531
Опции, переключатели и списки	553
Работа с изображениями	574
Просмотр изображений	575
Рисование изображения и текста	585
Создание изображений в сценарии	595
События	606
Объект события	606
Диспетчеризация событий	620
Резюме	630
Глава 9. Различные примеры	632
Триадная кривая Коха	632
Калькулятор	646
Бегущий текст	667
Игра «Жизнь»	674
Динамические рисунки	696
Резюме	714
Заключение. Немного о веб-программировании	715

Вступление

КНИГА О JAVASCRIPT

Видел чудеса техники, но такого...

из к/ф «Иван Васильевич меняет профессию»

Вниманию читателя предлагается книга о JavaScript. На сегодня JavaScript является одним из наиболее популярных языков программирования. Причем не просто программирования, а веб-программирования. Вместе с тем язык JavaScript в некоторых аспектах особенный. Поэтому, прежде чем приступить к его изучению, имеет смысл прокомментировать общее положение дел.

Как правило, JavaScript упоминают как *сценарный* язык. Другими словами, обычно на JavaScript создаются *сценарии*. Сценарий — это та же программа. Различие между программой и сценарием в том, под управлением какой среды они выполняются. Обычная программа чаще всего выполняется под управлением операционной системы. Сценарий выполняется под управлением *браузера*. Браузер, в свою очередь, представляет собой специальную программу, с помощью которой просматриваются веб-документы.



НА ЗАМЕТКУ

Браузеров существует достаточно много. Наиболее популярными (на момент написания книги) являются *Internet Explorer*, *Mozilla Firefox*, *Opera* и *Google Chrome*. Среди браузеров существует достаточно сильная конкуренция. Поэтому лидеры списка время от времени меняются, появляются новые браузеры, а уже существующие уходят со сцены. В общем, вопрос выбора браузера непростой и достаточно динамичный в плане предпочтений пользователей.

Указанное различие между программой и сценарием во многом техническое. Тем не менее оно имеет последствия. Собственно, об этих последствиях и поговорим далее.



НА ЗАМЕТКУ

То, что описывается далее, имеет некоторый привкус «технических подробностей». Эти подробности важные, но не критичные. Поэтому, даже если что-то покажется малопонятным, отчаиваться не стоит. Данный материал скорее для тех, кому интересно знать больше, чем необходимо.

Сценарии и программы

Меня не проведешь. Приемы сыщиков я вижу на пять футов вглубь.

*из к/ф «Приключения Шерлока Холмса
и доктора Ватсона»*

Что сценарий, что программа представляют собой набор инструкций, которые следует выполнить. Вопрос только в том, кто или что данные инструкции будет выполнять. Мы уже знаем, что программа выполняется под управлением операционной системы. Как это происходит? Достаточно просто и прозаично. Общая последовательность действий при создании и выполнении программы выглядит примерно следующим образом.

- Сначала набирается программный код. Это как раз та последовательность команд, которая должна быть выполнена. Программный код набирается в соответствии с правилами языка программирования, на котором пишется программа. Языков программирования очень много — например, C++, C#, Java или Python (этот перечень языков программирования далеко не полный). У каждого из них свои правила составления кодов.
- После того как код набран и сохранен, его следует выполнить. Проблема в том, что набранный нами код понятен для нас, но непонятен для компьютера. Необходима программа-посредник, которая переведет понятный для человека код в команды, понятные для компьютера (или, точнее, операционной системы, установленной на компьютере). В качестве такого посредника выступает или *компилятор*, или *интерпретатор*. Для каждого языка программирования предназначен свой персональный компилятор или интерпретатор.
- Если используется программа-компилятор, то исходный программный код компилируется и в результате получается набор

команд, готовых к выполнению операционной системой. Фактически при компиляции на основе набора инструкций, написанных на каком-то определенном языке программирования, создается набор команд машинного или квазимашинного уровня. Эти команды и выполняются при запуске программы на выполнение.

- Принципиальное отличие интерпретатора от компилятора состоит в том, что интерпретатор выполняет трансляцию исходного кода в исполняемый код, так сказать, в процессе выполнения, инструкция за инструкцией. Если компилятором конвертируется вся программа, а затем начинается ее выполнение, то интерпретатор конвертирует и исполняет сначала одну инструкцию, затем вторую, третью и так далее. То есть конвертирование программного кода происходит по мере необходимости.

Описанная последовательность действий достаточно условная, но вместе с тем показательная. Она дает ответ на вопрос, что же нам на самом деле нужно, чтобы написать программу на том или ином языке программирования. Во всяком случае, становится понятно, что без компилятора (или интерпретатора — все зависит от конкретного языка программирования) не обойтись.

В общем и целом получается так: имеется некоторый язык программирования, на котором мы собираемся писать программные коды. Язык программирования — это в широком смысле набор правил. Поэтому формально программный код можно написать, имея под рукой лишь обычный текстовый редактор. Но затем нам понадобится программа, способная не просто «понять» этот код, но и преобразовать его в форму, приемлемую для восприятия операционной системой. Здесь на сцену выходит компилятор или интерпретатор. Компиляторы и интерпретаторы предназначены для компилирования и интерпретации программ. Для этого они создаются фирмами-разработчиками. Это их основная задача. Что здесь важно? Важно то, что написание кода и его компиляция/интерпретация разнесены во времени и в пространстве. Другими словами, создавая программный код, мы можем особо не заботиться, как затем его компилировать или интерпретировать.

Еще один важный момент связан непосредственно с компиляторами и интерпретаторами. Это программы, которые *устанавливаются* на компьютер. Проще говоря, если мы хотим писать (и запускать) программы на языке C++, то нам необходимо будет установить компилятор для данного языка (и для данной операционной системы).



ДЕТАЛИ

Откровенно говоря, очень редко компилятор устанавливается отдельно. Обычно устанавливается *интегрированная среда разработки* (обычно упоминается как *IDE*, что является сокращением от английского *Integrated Development Environment*), в состав которой входит не только компилятор, но и редактор кодов, встроенный отладчик и многие другие полезные утилиты. Со средой разработки написание программы превращается в процесс комфортный и где-то даже приятный. Но в данном случае это не важно. Даже если используется среда разработки, компилятор все равно устанавливается и используется.

Все сказанное, напомним, относится к программам. Теперь вспомним о сценариях. Что принципиально меняется при написании сценариев? На самом деле, мало что. Как и в случае с программой, при написании сценария в соответствии с правилами языка (в данном случае имеется в виду язык JavaScript) набирается программный код. Код будет выполняться при выполнении сценария. Нас интересует создание сценариев на языке JavaScript. Сценарии на языке JavaScript *интерпретируются*. Но интерпретация выполняется не в явном виде, а, как отмечалось, средствами браузера. Беды в том нет, но есть определенное неудобство, связанное с необходимостью «инкапсуляции» сценария в веб-документ. Дело в том, что при интерпретации программы, с одной стороны, имеется программный код, а с другой — интерпретатор. Программный код интерпретируется интерпретатором. А вот если речь идет о сценарии, то этот сценарий должен быть включен в веб-документ. При открытии веб-документа браузером сценарий выполняется встроенными средствами браузера.



НА ЗАМЕТКУ

Здесь мы имеем в виду разработку с помощью JavaScript клиентских веб-приложений. То есть речь идет о приложениях, которые выполняются на компьютере клиента. С серверными приложениями ситуация несколько иная.

Фактически интерпретатор для сценария «спрятан» в браузере. Выполнить единственно сценарий при таком подходе проблематично. Сценарий приходится выполнять в контексте работы с веб-документом. Само по себе это не хорошо и не плохо. Но с методологической точки зрения при изучении языка JavaScript ситуация не самая луч-

шая, поскольку в «компании» с JavaScript-кодом сценария оказывается еще и код веб-документа. Причина в том, что основное назначение браузеров — работа с веб-документами. Выполнение сценариев браузерами — в известном смысле дополнительная функция.

НА ЗАМЕТКУ

Великий немецкий философ *Иммануил Кант* (1724–1804) утверждал, что к человеку должно относиться только как к цели, но не как к средству (*категорический императив Канта*). В рамках данной философской парадигмы, пожалуй, можно утверждать, что, тогда как программа для интерпретатора является целью, сценарий для браузера скорее является средством.



ДЕТАЛИ

Ситуация со сценариями (в плане выполнения кода под управлением не операционной системы, а другой программы — в данном случае браузера) не является уникальной. Например, существуют *макросы*. Скажем, при работе с приложением Excel из пакета Microsoft Office на языке VBA можно создавать программные коды, выполняемые под управлением приложения Excel. Это и есть макросы.

Здесь имеет смысл отметить, что при создании веб-документов используется специальный язык, который называется *языком гипертекстовой разметки*, или *HTML* (сокращение от английского *HyperText Markup Language*). Язык HTML не является предметом книги, но не упомянуть его совсем не удастся. Покоряясь неизбежному, предадимся рассуждениям о языке HTML.

Веб-документы и язык HTML

Он начинает новую жизнь, дайте ему возможность вспомнить все лучшее.

из к/ф «Покровские ворота»

Концепция языка гипертекстовой разметки HTML достаточно проста. В стандартный (обычный) текст добавляются специальные коды, которые принято называть *тегами* или *дескрипторами*. Данные коды представляют собой инструкции для браузера. Инструкции касаются способа отображения текста, помеченного дескрипторами. Поэто-

му на браузер в некотором приближении можно смотреть как на программу для просмотра текстов с гипертекстовой разметкой.



ДЕТАЛИ

Откровенно говоря, ситуация не такая простая, как может показаться на первый взгляд. Один и тот же документ с гипертекстовой разметкой в разных браузерах может отображаться по-разному. Более того, есть дескрипторы, которые «понимаются» одними браузерами и совершенно «не признаются» другими. Подход, базирующийся на создании HTML-документов, ориентированных на работу с одним определенным типом браузера, не проходит, поскольку концептуально веб-документы предназначены для использования в Интернете, что автоматически сводит на нет попытку ограничить пользователей документа браузером одного типа. То есть даже на уровне стандартного (не использующего сценарии) HTML-документа возникают неожиданные моменты. С другой стороны, для каждой проблемы обычно имеется более или менее удачное решение.

Создать HTML-документ достаточно просто. Из всех возможных простых способов мы здесь рассмотрим самый простой и наименее ресурсозатратный. Для этого понадобится простенький текстовый редактор вроде Notepad (*Блокнот*) и, естественно, браузер.



НА ЗАМЕТКУ

Браузер в принципе подойдет любой, но мы будем ориентироваться на группу лидеров: Internet Explorer, Mozilla Firefox, Google Chrome и Opera.

Сначала в текстовом редакторе необходимо набрать код документа. Под словом «код» имеется в виду HTML-код. Чтобы не быть голословными, поступим следующим образом: создадим пустой текстовый документ. Этот документ необходимо открыть и в окне текстового редактора ввести код, представленный в листинге В.1 (назначение инструкций кода поясняется позже).



Листинг В.1. Код гипертекстовой разметки документа

```
<!DOCTYPE HTML>  
<html>  
  <head>
```

```
<title>
  Омар Хайям. Рубаи
</title>
</head>
<body>
  <h3>Рубаи</h3>
  Чтоб мудро жизнь прожить, знать надобно немало,<br>
  Два важных правила запомни для начала:<br>
  Ты лучше голодай, чем что попало есть,<br>
  И лучше будь один, чем вместе с кем попало.
  <hr>
  <b>Омар Хайям</b>
</body>
</html>
```

Далее необходимо сохранить изменения в документе, закрыть его и изменить расширение txt на html (или htm). Собственно, все: мы создали HTML-документ. Этот документ можно открыть с помощью браузера. Также можно его открыть с помощью текстового редактора. В последнем случае увидим содержимое документа, то есть его HTML-код. На рис. В.1 показан документ с HTML-кодом из листинга В.1, открытый в текстовом редакторе.

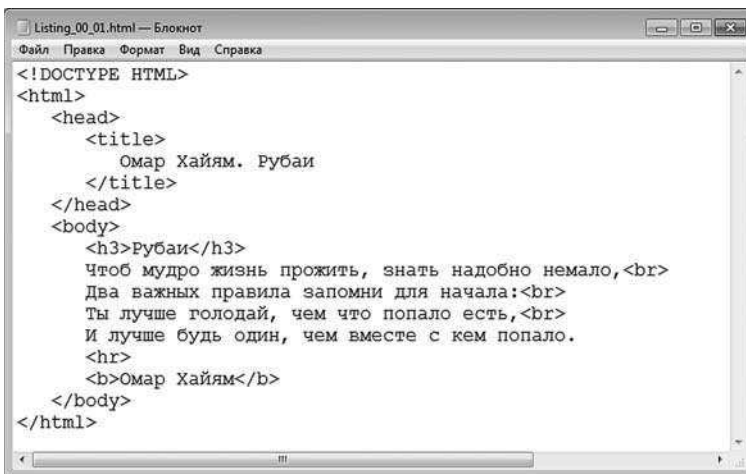


Рис. В.1. Документ с HTML-кодом открыт в текстовом редакторе

Если мы откроем документ с помощью браузера, получим несколько иной результат. На рис. В.2 показано, как будет выглядеть созданный нами документ в окне браузера Mozilla Firefox.

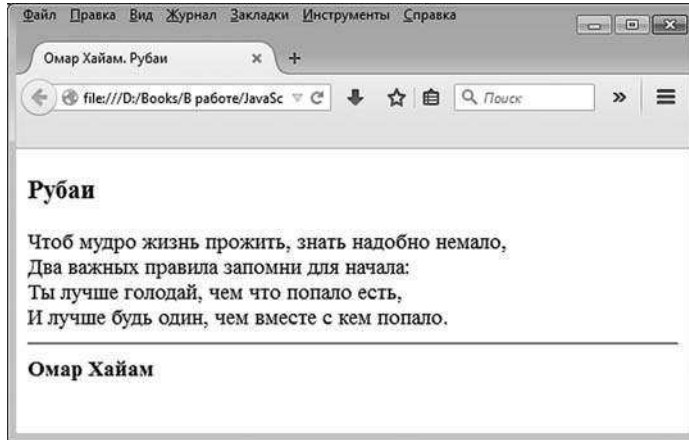


Рис. В.2. Документ открыт в окне браузера Mozilla Firefox

Для сравнения на рис. В.3 этот же документ открыт браузером Internet Explorer.

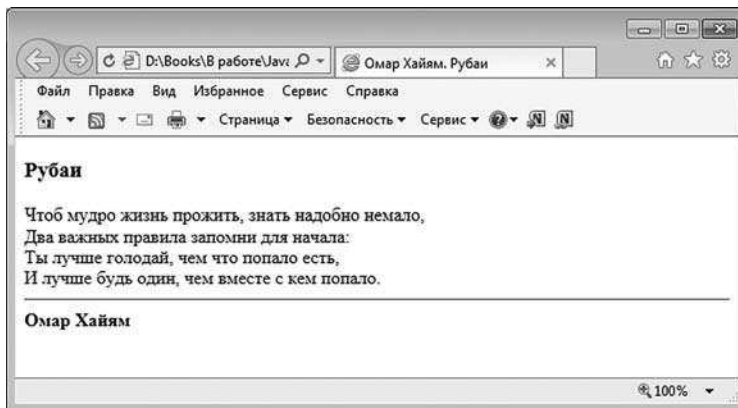


Рис. В.3. Документ открыт в окне браузера Internet Explorer

Также желающие могут взглянуть на рис. В.4 и рис. В.5, на которых показан документ с HTML-кодом, открытый соответственно с помощью браузеров Google Chrome и Opera (разработчик – компания Opera Software).

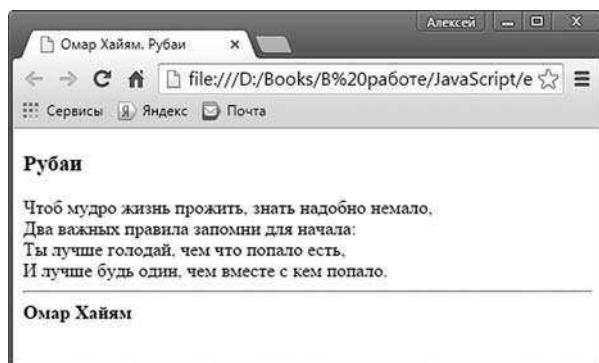


Рис. В.4. Документ открыт в окне браузера Google Chrome

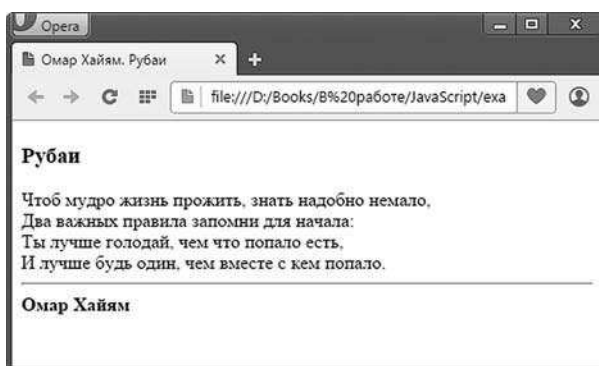


Рис. В.5. Документ открыт в окне браузера Opera

В принципе способ отображения документа разными браузерами идентичен. Так и должно быть. Теперь кратко обсудим назначение инструкций, использованных в HTML-коде документа.

НА ЗАМЕТКУ

В книге информация относительно HTML-кодировки обычно дается непосредственно при объяснении примеров или приводится в специальных врезках.

Сначала сделаем несколько общих замечаний относительно HTML-кодов. Во-первых, текст, не помеченный дескрипторами, отображается как обычно, то есть как текст. Во-вторых, дескрипторы бывают парными (таких большинство) и непарными. Дескриптор — это определенное ключевое слово, заключенное в угловые скобки. Если де-

скриптор парный, то второй (закрывающий) дескриптор отличается от первого (открывающего) дескриптора наличием обратной косой черты / перед ключевым словом в угловых скобках.



ЯЗЫК ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML

Дескриптор имеет вид <дескриптор>. Это открывающий дескриптор. Закрывающий дескриптор (если такой имеется) будет выглядеть как </дескриптор>. Например, пара дескрипторов и определяют фрагмент текста, который в браузере будет отображаться жирным шрифтом. Здесь речь идет о парном дескрипторе. Примеры непарных дескрипторов:

- дескриптор
 является инструкцией разрыва строки (в месте размещения данного дескриптора выполняется разрыв текстовой строки и осуществляется переход к новой строке для отображения текста);
- дескриптор <hr> является инструкцией отображения горизонтальной линии (по умолчанию на ширину окна браузера).

В-третьих, у HTML-документа должна быть определенная структура, которая формируется, как несложно догадаться, с помощью дескрипторов.



ЯЗЫК ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML

Откровенно говоря, браузеры очень демократичны в отношении правил оформления HTML-документа. Даже если взять самый обычный текстовый файл, добавить туда некоторые дескрипторы (например, выделить часть текста дескрипторами и), сохранить файл с расширением html и затем открыть его в браузере, браузер отобразит документ с учетом наличия в нем дескрипторов.

Вместе с тем существуют определенные правила создания документов с гипертекстовой разметкой, и их стоит придерживаться.

Использованный нами код содержит как парные, так и непарные дескрипторы. Начинается он с дескриптора <!DOCTYPE HTML>, являющегося стандартным началом HTML-документа, указывающим браузеру, с какого типа данными предстоит иметь дело. Весь фактический HTML-код размещается между дескрипторами <html> (открывает код документа) и </html> (закрывает код документа). Между дескрипторами <html> и </html> размещается несколько блоков кода (в данном случае блоков два). Первый блок выделен дескрипторами <head> и </head>.