А. Н. МАСЛОБОЕВ

WEB-СТРАНИЦЫ ИСПОЛЬЗОВАНИЕ КАСКАДНЫХ ТАБЛИЦ СТИЛЕЙ ДЛЯ ОФОРМЛЕНИЯ WEB-СТРАНИЦ

Учебное пособие

Санкт-Петербург
2021

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»
Высшая школа технологии и энергетики

А. Н. Маслобоев

WEB-СТРАНИЦЫ ИСПОЛЬЗОВАНИЕ КАСКАДНЫХ ТАБЛИЦ СТИЛЕЙ ДЛЯ ОФОРМЛЕНИЯ WEB-СТРАНИЦ

Учебное пособие

Утверждено Редакционно-издательским советом ВШТЭ СПбГУПТД

Санкт-Петербург
2021

УДК 681.3 (075)
ББК 22.18я7
М 316

Рецензенты:
доктор технических наук, профессор кафедры аудиовизуальных систем и технологий факультета телевидения, дизайна и фотографии Санкт-Петербургского государственного института кино и телевидения (СПбГИКиТ)
В. М. Пестриков;
кандидат технических наук, профессор, заведующий кафедрой информационно-измерительных технологий и систем управления ВШТЭ СПбГУПТД
В. И. Сидельников

Маслобоев, А. Н.
М316 Web-страницы. Использование каскадных таблиц стилей для оформления web-страниц / А. Н. Маслобоев. – СПб.: ВШТЭ СПбГУПТД, 2021. – 66 с.
ISBN 978-5-91646-275-3

Учебное пособие соответствует программам и учебным планам дисциплины «Web-страницы» для студентов, обучающихся по направлению подготовки 01.03.02 «Прикладная математика и информатика». В настоящем учебном пособии рассматриваются вопросы использования каскадных таблиц стилей CSS для оформления web-страниц. Пособие содержит необходимые теоретические сведения и примеры каскадных таблиц стилей. Также рассматривается их взаимодействие с web-страницами, созданными средствами языка HTML. С целью лучшего усвоения студентами изложенного материала предлагаются контрольные вопросы.

Пособие предназначено для подготовки бакалавров очной формы обучения.

УДК 681.3 (075)
ББК 22.18я7

ISBN 978-5-91646-275-3

ОГЛАВЛЕНИЕ

Введение 4
Глава 1. Форматирование простого текста 6
Контрольные вопросы 13
Глава 2. Использование классов 14
Контрольные вопросы 27
Глава 3. Оформление списков различных видов 28
Контрольные вопросы 38
Глава 4. Оформление таблиц 39
Контрольные вопросы 51
Глава 5. Работа с блоками 52
Контрольные вопросы 62
Заключение 63
Библиографический список 64

ВВЕДЕНИЕ

В настоящее время неотъемлемой частью подготовки специалистов по информатике и информационным технологиям является их обучение созданию и оформлению web-страниц. При этом современный подход к их разработке подразумевает, что создание web-страниц производится в два этапа с использованием различных средств.

На первом этапе средствами языка HTML создается сама web-страница. Язык HTML описывает основную структуру данной страницы. С помощью тегов (команд) данного языка текст страницы разбивается на абзацы, в тексте создаются заголовки различных уровней, где это необходимо для лучшего восприятия, фрагменты текста представляются в виде списков. Для более наглядного представления информации на странице могут использоваться таблицы. Web-страницы могут связываться с другими страницами с помощью гиперссылок. В текст страницы тегами HTML могут вставляться элементы компьютерной графики, а также аудио- и видеофайлы.

Но следует помнить о том, что вся информация, представленная на web-странице, должна быть не просто структурирована, но и соответствующим образом оформлена. Язык HTML обладает определенным набором средств для оформления страниц, но следует отметить, что эти возможности являются в достаточной степени ограниченными. В частности, в языке HTML отсутствуют штатные средства, которые позволяли бы создать в начале абзаца красную строку заданной величины, нет возможности задать в документе поля нужного размера, указать размеры символов в тексте не в относительном виде, а в абсолютных единицах (пунктах, миллиметрах, сантиметрах), определить цвет рамки для таблицы, отличающийся от стандартного. И это только часть тех возможностей, которые необходимы web-разработчику для создания грамотно и эффектно оформленной web-страницы.

Выходом из данной ситуации является использование языка каскадных таблиц стилей (часто для его обозначения используется англоязычная аббревиатура CSS, что расшифровывается как Cascading Style Sheets). CSS является формальным языком описания внешнего вида документа (как правило, таким документом является web-страница), написанного с использованием языка разметки HTML. CSS может регулировать внешний вид документа, включая цвет и гарнитуру шрифта, фоновый цвет страницы, расположение различных блоков, составляющих документ, и многие другие параметры с помощью набора правил, о которых будет подробно рассказано в главе 1 данного пособия.

Набор правил может находиться внутри HTML-документа или представлять собой отдельный текстовый файл с расширением **css**. Последний способ является более предпочтительным по причинам, о которых будет сказано ниже. Таким образом, описание логической структуры web-документа отделяется от описания его внешнего вида, что является более удобным и

рациональным, обеспечивает большую гибкость в представлении документа, уменьшает объем и сложность работы web-программиста и значительно повышает ее производительность.

Исходя из всего вышеизложенного, знание языка CSS является необходимостью для современного специалиста по информационным технологиям. Рассмотрению основ данного языка и вопросам его практического применения и посвящено данное учебное пособие.

ГЛАВА 1. ФОРМАТИРОВАНИЕ ПРОСТОГО ТЕКСТА

Для того чтобы начать использование каскадных таблиц стилей, необходимо прежде всего изучить внутреннюю структуру такой таблицы. Любая каскадная таблица стилей фактически представляет собой набор правил по оформлению web-страницы. Общий вид правила представлен ниже.

Селектор {свойство 1: значение 1; свойство 2: значение 2; …, свойство n: значение n},

где **селектор** представляет собой перечень элементов, для которых сформулировано данное правило. В простейшем случае элемент является одним из тегов языка HTML. Таких элементов может быть один или несколько. В случае, если элементов несколько, они перечисляются через запятую. Далее в фигурных скобках указывается, собственно, само правило. Оно состоит из перечня **свойств**, для каждого из которых указывается после двоеточия его **значение**. Каждая такая пара **свойство – значение** отделяется от другой точкой с запятой.

При создании каскадной таблицы стилей следует помнить, что названия свойств в языке CSS могут отличаться от названий аналогичных свойств в языке HTML. Например, в HTML свойство, которое отвечает за выравнивание текста внутри параграфа или абзаца, называется **align**, а в CSS такое же свойство называется **text-align**. При этом значения свойств, как правило, остаются теми же самыми, т. е. и в HTML, и в CSS можно выравнивать текст внутри одного из указанных структурных элементов четырьмя различными способами: по левому краю, по правому краю, по центру и по ширине. При этом соответствующие значения называются одинаково в обоих языках: **left, right, center и justify**.

Приведем некоторые свойства, которые нам понадобятся в дальнейшем, с указанием их возможных значений:

font-family – определяет гарнитуру одного шрифта или семейства шрифтов; в качестве значения указывается название соответствующего шрифта. Если шрифтов несколько, то они перечисляются через запятую. Если название шрифта состоит из нескольких слов (например, «Times New Roman») то его лучше заключать в кавычки;

font-size – определяет размер шрифта, который может выражаться в различных единицах: пунктах, пикселях или сантиметрах, т. е. в абсолютных единицах. В этом заключается отличие от свойства **size** в HTML, которое позволяет задать только относительные размеры шрифта;

color – задает цвет шрифта, который может быть выражен или в виде текстовой константы (одного из названий цветов на английском языке), или в виде шестнадцатеричного числового кода;

text-indent – задает размер красной строки (т. е. отступа в начале абзаца). Величина этого отступа может задаваться в тех же единицах, что и для свойства **font-size**. Это свойство предоставляет дополнительную возможность, которая вообще отсутствует в HTML;

margin-left – задает размер левого поля web-документа, т. е. расстояние от левой границы окна браузера до границы текста. Измеряется в тех же единицах, что и для свойства **font-size**;

margin-right – задает размер правого поля web-документа, т. е. расстояние от правой границы окна браузера до границы текста. Измеряется в тех же единицах, что и для свойства **font-size**;

margin-top – задает размер верхнего поля web-документа, т. е. расстояние от верхней границы окна браузера до границы текста. Измеряется в тех же единицах, что и для свойства **font-size;**

margin-bottom – задает размер нижнего поля web-документа, т. е. расстояние от нижней границы окна браузера до границы текста. Измеряется в тех же единицах, что и для свойства **font-size.**

Как уже было сказано во введении, существует два основных способа использования каскадных таблиц стилей.

Первый заключается в том, что каскадная таблица стилей встраивается непосредственно в web-документ. Для этого внутри тега заголовка страницы **<HEAD>…</HEAD>** помещается еще один парный тег **<STYLE>… </STYLE>**. Внутри данного тега и располагается каскадная таблица стилей. Недостатком этого способа является то, что такая таблица стилей может быть использована только для форматирования той web-страницы, в заголовке которой она находится.

Другой, альтернативный, способ применения таблиц стилей состоит в том, что таблица стилей создается и сохраняется на компьютере в виде отдельного файла. Этот файл создается как обычный текстовый документ (например, в редакторе «Блокнот», который является стандартным приложением операционной системы Windows). Далее этот файл сохраняется на компьютере, но не с расширением **txt**, а с расширением **css**. Подобный файл будет интерпретироваться всеми современными web-браузерами как каскадная таблица стилей. Далее этот файл подключается к web-странице, написанной на языке HTML, с помощью одиночного тега **LINK**, который также помещается в заголовочной части web-страницы. Этот способ является предпочтительным потому, что одну и ту же таблицу стилей можно подключать к разным страницам. Это позволяет, в частности, без дополнительных затрат времени и сил создать единый стиль оформления для целого web-сайта. Поэтому в следующих примерах мы будет использовать только второй способ.

Рассмотрим следующий пример использования каскадной таблицы стилей для оформления простой текстовой web-страницы. Имеется страница, созданная средствами языка HTML (пока без использования каскадной таблицы стилей). Она содержит начало знаменитой сказочной повести итальянского

писателя Карло Коллоди «Приключения Пиноккио». Ниже приводится ее исходный код, сохраненный в текстовом файле **Primer1.html**.

```
<HTML>

<HEAD>

<TITLE>
Пример 1
</TITLE>

</HEAD>

<BODY>

<H1>
КАРЛО КОЛЛОДИ
ПРИКЛЮЧЕНИЯ ПИНОККИО
</H1>

<H2>
1. КАК МАСТЕРУ ВИШНЕ ПОПАЛСЯ КУСОК ДЕРЕВА, КОТОРЫЙ ПЛАКАЛ И СМЕЯЛСЯ, КАК РЕБЁНОК
</H2>

<P>
Жил-был…
</P>

<P>
«Король!» – немедленно воскликнут мои маленькие читатели.
</P>

<P>
Нет, дети, вы не угадали. Жил-был кусок дерева.
</P>

<P>
То было не какое-нибудь благородное дерево, а самое обыкновенное полено, из тех, которыми в зимнюю пору топят печи и камины, чтобы обогреть комнату.
</P>

<P>
Не знаю уж, какими путями, но в один прекрасный день этот кусок дерева оказался в мастерской старого столяра. Старика звали мастер Антонио, но весь свет именовал его «мастер Вишня», так как кончик его носа был подобен спелой вишне – вечно блестящий и сизо-красный.
</P>
```

<P>

Мастер Вишня страшно обрадовался, обнаружив полено, и, весело потирая руки, пробормотал:

</P>

<P>

– Этот кусок дерева попался мне довольно кстати. Смастерю-ка я из него ножку для стола.

</P>

<P>

Сказано – сделано. Не мешкая, он взял острый топор, чтобы очистить кору и придать дереву форму ножки. Но не успел он занести топор, как рука его так и повисла в воздухе – из полена послышался тонкий, умоляющий голосок:

</P>

<P>

– Не бейте меня!

</P>

<P>

Можете себе представить, какое сделалось лицо у доброго старого мастера Вишни.

Изумлённый в высшей степени, он начал водить глазами по мастерской, чтобы узнать, откуда взялся этот голосок. Но в комнате никого не было. Он заглянул под верстак – никого. Посмотрел в шкаф, который обычно держал запертым, – никого. Сунул голову в корзину с опилками и стружками – никого. Наконец открыл ставню и поглядел на улицу – тоже никого. Может быть…

</P>

<P>

– Я все понял, – захихикал он и почесал под париком. – Голосок мне просто померещился. Значит, снова за работу! И он опять взялся за топор и нанёс превосходнейший удар по деревяшке.

</P>

<P>

– Ой, ты мне сделал больно! – завопил знакомый голосок.

</P>

<P>

Для мастера Вишни это было уже слишком. Глаза у него от страха полезли на лоб, рот раскрылся, язык свесился до подбородка, так что старик стал похож на одну из тех удивительных статуй, какими в старину украшали фонтаны.

</P>

</BODY>

</HTML>

На рис. 1 показан внешний вид данной web-страницы при просмотре ее в браузере Firefox.

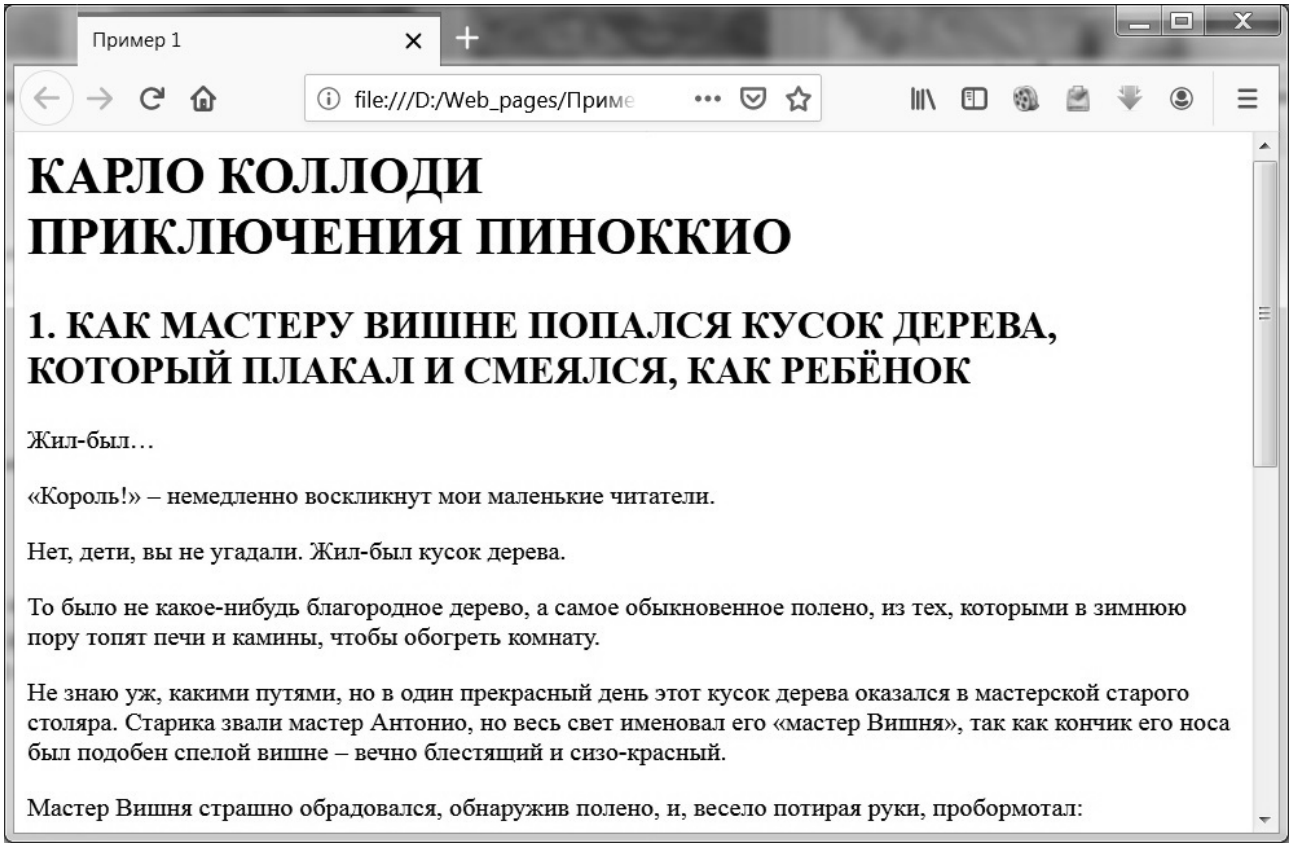

Рис. 1. Простая текстовая страница (используется только язык HTML)

Данная страница содержит заголовок первого уровня, включающий имя автора и название книги, заголовок второго уровня с названием главы и ряд текстовых параграфов. HTML-файл не имеет никаких элементов форматирования: текст отображается черным цветом на белом фоне, гарнитура шрифта и его размер представлены значениями по умолчанию, красной строки в начале абзацев нет, поля в тексте документа отсутствуют и текст расположен вплотную у края страницы.

Теперь преобразуем внешний вид данного web-документа при помощи каскадной таблицы стилей, набранной и сохраненной в отдельном текстовом файле **Style1.css**. Вот исходный код этой таблицы:

BODY {background-color:#CCFFCC; color:Blue}
H1 {font-family:Arial; color:Red; font-size:28pt; text-align:Center}

H2 {font-family:Arial; color:Green; font-size:18pt; text-align:Center}

P{font-family:"Comic Sans MS"; font-size:16pt; Text-indent:20pt; Text-Align:justify; margin-left:20pt; margin-right:20pt}

Для подключения каскадной таблицы стилей к web-странице нужно установить между ними связь. Для этой цели в заголовочную часть web-страницы (т. е. внутри парного тега **<HEAD> … </HEAD>**) вставляется одиночный тег **<LINK>**. Этот тег имеет следующие обязательные атрибуты:

HREF – указывает имя каскадной таблицы стилей. В случае, если таблица стилей находится в той же папке, что и Web-страница, достаточно указать только ее собственное имя. Если же таблица находится в другой папке, то необходимо указать путь к ней.

REL – указывает отношение таблицы стилей к web-странице. Этот атрибут должен иметь значение **STYLESHEET**, что в переводе с английского и означает таблица стилей.

TYPE – указывает тип файла, в котором находится подключаемая таблица стилей. Этот атрибут должен иметь значение **TEXT/CSS**.

Таким образом, в целом команда подключения таблицы стилей для данного примера выглядит так:

<LINK HREF="Style1.css" REL="Stylesheet" TYPE="Text/CSS">

Внешний вид данной страницы после подключения каскадной таблицы стилей при просмотре ее в браузере Firefox приведен на рис. 2.

При сравнении с предыдущим вариантом той же страницы (см. рис. 1) можно заметить следующие отличия:

а) изменился шрифт заголовка первого уровня (теперь этот шрифт Arial);

б) изменился шрифт заголовка второго уровня (также на шрифт Arial);

в) изменился шрифт заголовка для основного текста web-страницы (на шрифт Comic Sans MS);

г) увеличился размер шрифта для абзацев, содержащих основной текст данной страницы;

д) текст теперь не прилегает вплотную к границам окна; слева и справа от текста имеются поля;

е) в начале текстовых абзацев появились отступы.

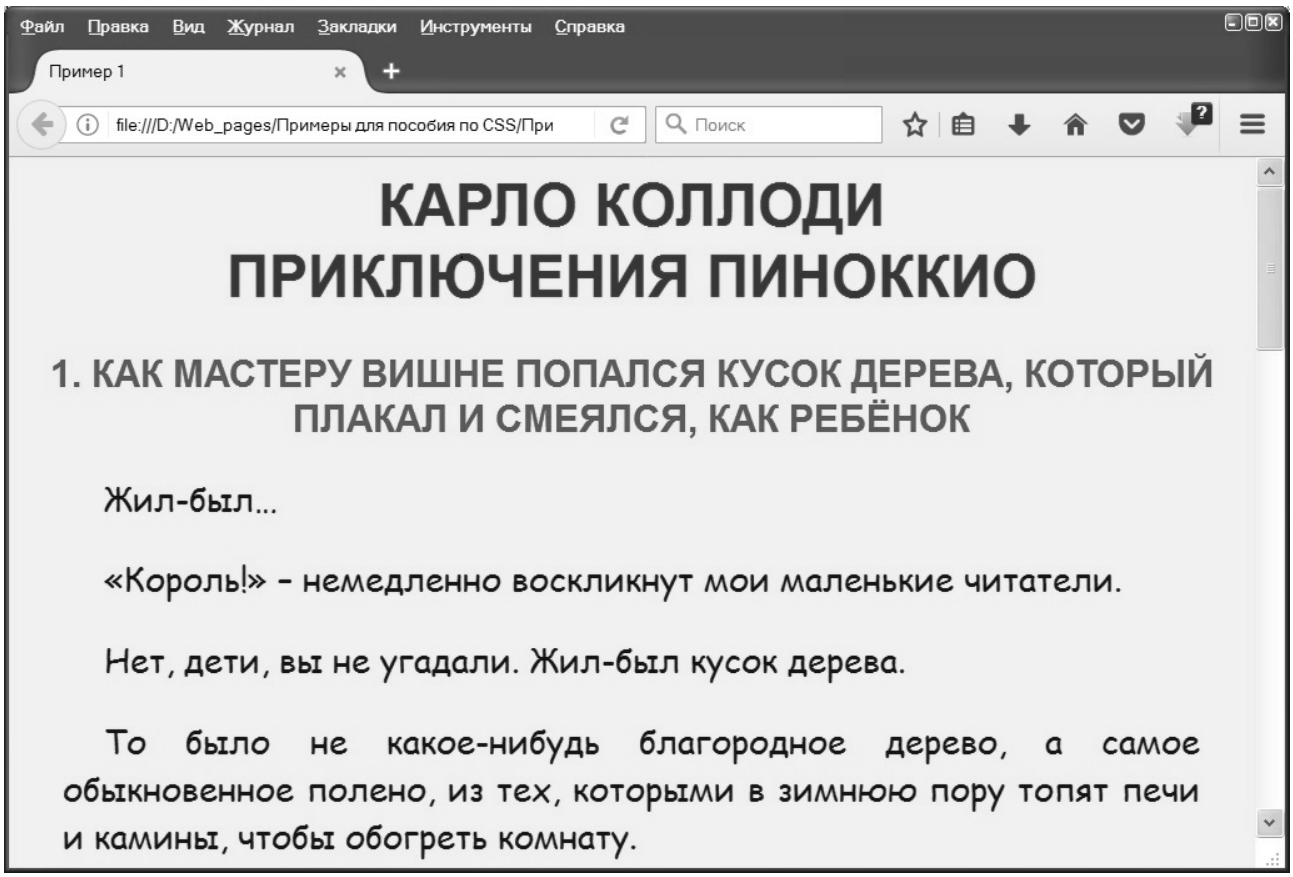

Рис. 2. Текстовая страница после подключения к ней каскадной таблицы стилей

Иллюстрации в данном пособии являются черно-белыми. Но если набрать текст данной web-страницы и соответствующей ей каскадной таблицы стилей на компьютере, а затем подключить каскадную таблицу стилей, то будет видно, что поменялась и цветовая палитра страницы. Заголовок первого уровня окрасился в красный цвет, второго уровня – стал темно-зеленым, а основной цвет текста – синим. Кроме того, фон web-страницы стал светло-зеленым.

Из представленной на этой странице иллюстрации видно, что даже использование приведенной выше небольшой по объему и очень простой по содержанию каскадной таблицы стилей может существенно изменить в лучшую сторону внешний вид web-страницы.

Контрольные вопросы

1. Для чего нужны каскадные таблицы стилей?
2. Назовите два основных способа использования каскадных таблиц стилей.
3. Какова внутренняя структура каскадной таблицы стилей?
4. Какова структура правила каскадной таблицы стилей?
5. Что такое селектор применительно к каскадным таблицам стилей?
6. В какие скобки заключается перечень свойств в каскадных таблицах стилей?
7. Для чего в каскадных таблицах стилей используется двоеточие?
8. Какое свойство каскадных таблиц стилей определяет гарнитуру используемого шрифта?
9. Какое свойство каскадных таблиц стилей определяет размер используемого шрифта?
10. Какое свойство каскадных таблиц стилей задает размер полей для страницы?
11. Какое свойство каскадных таблиц стилей определяет цвет шрифта?
12. Какое свойство каскадных таблиц стилей определяет фоновый цвет документа?
13. Какими двумя основными способами можно задавать цвет шрифта или фона в каскадных таблицах стилей?
14. Какой тег языка HTML позволяет подключить к web-странице каскадную таблицу стилей?

ГЛАВА 2. ИСПОЛЬЗОВАНИЕ КЛАССОВ

Дополнительным средством оформления, которое часто используется в каскадных таблицах стилей, является класс. Описание класса позволяет задать указанный в нем стиль оформления для каждого элемента, относящего к данному классу.

Общий вид описания класса в каскадной таблице стилей следующий:

Селектор. Название_класса {свойство 1: значение 1; свойство 2: значение 2; …, свойство n: значение n}

Если посмотреть на данное описание, то видно, что оно похоже на обычное правило каскадной таблицы стилей за одним исключением. В это правило добавляется еще один элемент – название класса, которое пишется через точку после селектора. В остальном принципиальных отличий нет, т. е. дальше в фигурных скобках указывается список свойств и их значений.

Для того чтобы в HTML-коде какой-либо страницы указать, что она оформляется с помощью определенного класса, нужно прописать это в соответствующем теге. После собственно имени тега необходимо указать следующую команду:

CLASS = ”Название класса”

Тогда все, что находится внутри данного тега, будет отформатировано в соответствии с параметрами, которые указаны в описании класса, находящемся в таблице стилей.

Рассмотрим использование классов на следующем примере. Имеется страница сайта, посвященного трем различным языкам программирования: Пайтону, Бейсику и Си. Для каждого из перечисленных языков приводятся как краткие сведения о них, так и фрагменты программ на этих языках. Ниже приводится «чистый» (без подключения каскадной таблицы стилей) HTML-код этой страницы.

```
<HTML>
<HEAD>
<TITLE>
Языки программирования
</TITLE>
</HEAD>
<BODY>

<H1>
Языки программирования
</H1>
```

<P>

Эта страница содержит справочную информацию по языкам программирования Бейсику, Пайтону и Си. Каждому языку на странице посвящен отдельный раздел.

<P>

<H2> Бейсик </H2>

<P>

Язык Бейсик изначально создавался как средство для обучения студентов основам программирования. Бейсик был разработан преподавателями Дартмутского Колледжа в Канаде Джоном Кемени и Томасом Курцем в 1964 г. Название языка - сокращение от английского названия Beginner's All-purpose Symbolic Instruction Code — универсальный код символических инструкций для начинающих.

</P>

<P>

Бейсик - это первый язык программирования, который получил широкое распространение на персональных компьютерах. Язык изначально создавался как алгоритмический, а затем стал процедурным и объектно-ориентированным.

</P>

<P>

**Пример программы на языке Бейсик
**
**CLS
**
**PRINT "Решение квадратного уравнения"
**
**PRINT "Введите коэффициенты уравнения"
**
**INPUT a,b,c
**
**d=b^2-4*a*c
**
**IF d>=0 THEN
**
x1=(-b-SQR(d))/(2*a)
x2=(-b+SQR(d))/(2*a)
**
**
PRINT "Корни уравнения ",x1,x2
**
**
ELSE
**
**
PRINT "Уравнение не имеет решения"
**
**
END IF
</P>

<H2> Пайтон </H2>

<P>

Высокоуровневый язык программирования Пайтон был разработан в 1991 году нидерландским программистом Гвидо ван Россумом. Название язык получил в честь известного британского комедийного сериала «Летающий цирк Монти Пайтона», который будущий автор языка часто смотрел, еще будучи школьником и студентом.

</P>

<P>

Популярность языка Пайтон объясняется следующими обстоятельствами. С одной стороны, этот язык имеет простой и интуитивно понятный синтаксис. Поэтому Пайтон достаточно легко и быстро может даже начинающий пользователь. С другой стороны – это современный мультипарадигмальный язык, который обеспечивает все возможности для применения современных передовых технологий, включая объектно-ориентированное программирование.</P>

<P>

**Пример программы на языке Пайтон
**

**# Вычисление суммы цифр трехзначного числа
**

**# с проверкой правильности ввода исходных данных
**

**print("Введите трехзначное число")
**

**n=int(input())
**

**if (n>=100)and (n<=999):
**

**   r1=n//100;
**

**   t=n%100
**

**   r2=t//10
**

**   r3=t%10
**

**   x=r1+r2+r3
**

**   print("Сумма цифр числа равна ",x)
**

**else:
**

**   print("Ошибка ввода")
**

**   print ("Число должно быть в диапазоне от 100 до 999")
**

</P>

<H2> Си </H2>

<P>

Язык Си был создан в 1972 г. сотрудником американской корпорации Bell Laboratories Деннисом Ритчи. Первоначально этот язык был разработан для реализации операционной системы UNIX, но впоследствии был перенесён на множество других платформ.

</P>

<P>

Язык Си получил широкое применение при создании системного программного обеспечения и прикладного программного обеспечения для решения широкого круга задач. Синтаксис языка Си в дальнейшем стал основой для таких популярных языков программирования, как C++, C#, Java

</P>

<P>
**Пример программы на языке Си:
**
**/*Прогнозируется поведение ракеты,
**
**в зависимости от ее начальной скорости*/
**
#include <stdio.h>
**
**
main()
**
**
{
**
**
float V;
**
**
printf("Введите V\n");
**
**
scanf("%f",&V);
**
**
if (V<7.8)
**
**
printf("Ракета упадет на Землю"\n");
**
**
else
**
**
if (V<11.2)
**
**
printf("Ракета станет спутником Земли\n");
**
**
else
**
**
if (V<16.4)
**
**
printf("Ракета станет спутником Солнца\n");
**
**
else
**
**
printf("Ракета покинет Солнечную систему\n");
}
</P>

</BODY>
</HTML>

Внешний вид данной web-страницы при просмотре ее в браузере Firefox приведен на рис. 3.

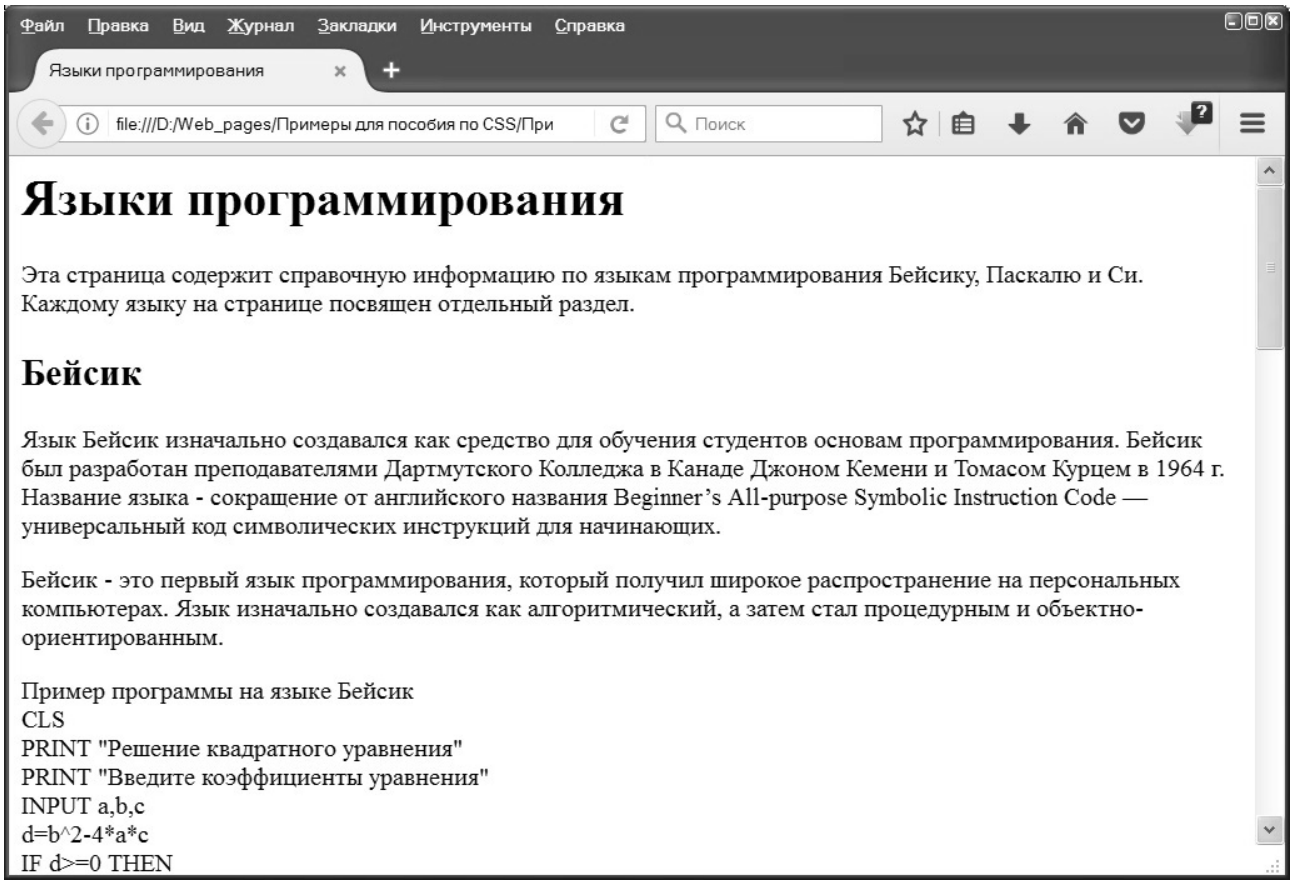

Рис. 3. Страница, посвященная различным языкам программирования (таблица стилей не подключена)

Теперь создадим для оформления данной страницы каскадную таблицу стилей **Style2.css**, текст которой представлен ниже:

BODY {background-color:#CCFFCC; color:Blue; margin-left:20pt; margin-right:20pt}
H1 {font-family:Arial; Color:Red; font-size:28pt; text-align:Center}
H2 {font-family:Arial; Color:Green; font-size:18pt; text-align:Center}
P{font-family:"Times New Roman"; font-size: 16pt; Text-indent:20pt; Text-Align:justify}
P.PROG{font-family:"Courier New"; font-size:16pt; Text-Align:justify; Text-indent:0pt; font-weight:bold}

P.ZAG{font-family:Calibri,Verdana; font-size: 18pt; Text-Align: justify; Text-decoration: underline; font-weight:bold}

Обратим внимание на некоторые изменения и дополнения, которые появились в данной таблице стилей по сравнению с той, которая приведена в главе 1 данного пособия.

Во-первых, в данной таблице стилей появились два новых свойства, которые можно использовать для оформления шрифта: **font-weight** и **font-decoration**. Свойство font-weight определяет жирность шрифта. Его значения могут быть выражены как служебными словами, так и в численном виде. Для задания этого параметра могут быть использованы следующие слова:

normal – обычное начертание;
lighter – тонкое начертание;
bold – полужирное начертание;
bolder – жирное начертание.

Кроме того, можно устанавливать для жирности числовые значения от 100 до 900 с шагом 100. Значение 100 соответствует сверхтонкому начертанию, а 900 – сверхжирному. Нормальная же толщина шрифта (которая устанавливается по умолчанию) имеет значение 400.

Свойство **text-decoration** отвечает за различные специальные эффекты, которые могут применяться при оформлении текста. Оно может принимать следующие фиксированные значения:

underline – подчеркнутый текст;
line-through – перечеркнутый текст;
overline – линия над текстом;
blink – мигающий текст;
none – отсутствие спецэффектов.

Во-вторых, описание свойств полей (**margin-left** и **margin-right**) перенесено в правило для тега **BODY**. Таким образом, поля будут назначены не только для параграфов, но и для всей страницы в целом. Это показывает, что одни и те же свойства могут фигурировать в правилах для разных элементов.

В-третьих, в данной таблице появился новый элемент в виде двух классов. Класс **P.PROG** применяется к тем параграфам web-страницы, в которых содержатся программные коды. Обратите внимание, что имя класса всегда начинается с точки. Эта точка отделяет название элемента, к которому относится класс, от названия самого класса.

Рассмотрим, в чем состоит отличие параграфов, в которых используется класс **P.PROG**, от всех прочих параграфов web-документа. Начнем с того, что они набраны шрифтом, отличным от основного текста документа. Основный текст отображается с помощью шрифта Times New Roman, в то время как для

текста программ используется моноширинный шрифт Courier New. Кроме того, этот шрифт является полужирным, а красная строка в начале абзаца с программными кодами отсутствует.

Второй класс используется для заголовков, предшествующих тексту программ, и носит соответствующее название **P.ZAG**. Этот класс также применяется к параграфам и имеет следующие отличительные черты: для текста используется шрифт Calibri или Verdana, размер шрифта составляет 18 пунктов, шрифт полужирный, используется подчеркивание.

Для того чтобы данные классы можно было применить для оформления web-страницы, требуется ввести коррективы в содержание самой этой страницы. Для тех параграфов, в которых используются классы, это необходимо прописать в соответствующих тегах. В результате HTML-код данной страницы несколько изменится, и будет выглядеть так:

```
<HTML>
<HEAD>
<TITLE>
Языки программирования
</TITLE>
<LINK HREF="STYLE2.CSS" REL="STYLESHEET" TYPE="TEXT/CSS">
</HEAD>

<BODY>
<H1>
 Языки программирования
</H1>
<P>
Эта страница содержит справочную информацию по языкам программирования Бейсику, Пайтону и Си. Каждому языку на странице посвящен отдельный раздел.
<P>
<H2> Бейсик </H2>
<P>
Язык Бейсик изначально создавался как средство для обучения студентов основам программирования. Бейсик был разработан преподавателями Дартмутского Колледжа в Канаде Джоном Кемени и Томасом Курцем в 1964 г. Название языка - сокращение от английского названия Beginner’s All-purpose Symbolic Instruction Code — универсальный код символических инструкций для начинающих.
</P>
<P>
Бейсик - это первый язык программирования, который получил широкое распространение на персональных компьютерах. Язык
```

изначально создавался как алгоритмический, а затем стал процедурным и объектно-ориентированным.

</P>

<P CLASS="ZAG">

**Пример программы на языке Бейсик:
**

</P>

<P CLASS="PROG">

**CLS
**

PRINT "Решение квадратного уравнения"

**
**

PRINT "Введите коэффициенты уравнения"

**
**

INPUT a,b,c

**
**

d=b^2-4*a*c

**
**

IF d>=0 THEN

**
**

x1=(-b-SQR(d))/(2*a)

**x2=(-b+SQR(d))/(2*a)
**

PRINT "Корни уравнения ",x1,x2

**
**

ELSE

**
**

**PRINT "Уравнение не имеет решения"
**

END IF

</P>

<H2> Пайтон </H2>

<P>

Высокоуровневый язык программирования Пайтон был разработан в 1991 году нидерландским программистом Гвидо ван Россумом. Название язык получил в честь известного британского комедийного сериала «Летающий цирк Монти Пайтона», который будущий автор языка часто смотрел, еще будучи школьником и студентом.</P>

<P>

Популярность языка Пайтон объясняется следующими обстоятельствами. С одной стороны, этот язык имеет простой и интуитивно понятный синтаксис. Поэтому Пайтон достаточно легко и быстро может даже начинающий пользователь. С другой стороны – это современный мультипарадигмальный язык, который обеспечивает все возможности для применения современных передовых технологий, включая объектно-ориентированное программирование.</P>

<P CLASS="ZAG">

**Пример программы на языке Пайтон
**

</P>
<P CLASS="PROG">
**# Вычисление суммы цифр трехзначного числа
**
**# с проверкой правильности ввода исходных данных
**
**print("Введите трехзначное число")
**
**n=int(input())
**
**if (n>=100)and (n<=999):
**
**   r1=n//100;
**
**   t=n%100
**
**   r2=t//10
**
**   r3=t%10
**
**   x=r1+r2+r3
**
**   print("Сумма цифр числа равна ",x)
**
**else:
**
**   print("Ошибка ввода")
**
**   print ("Число должно быть в диапазоне от 100 до 999")
**
</P>
<H2> Си </H2>
<P>
Язык Си был создан в 1972 г. сотрудником американской корпорации Bell Laboratories Деннисом Ритчи. Первоначально этот язык был разработан для реализации операционной системы UNIX, но впоследствии был перенесён на множество других платформ.
</P>
<P>
Язык Си получил широкое применение при создании системного программного обеспечения и прикладного программного обеспечения для решения широкого круга задач. Синтаксис языка Си в дальнейшем стал основой для таких языков программирования, как C++, C#, Java
</P>
<P CLASS="ZAG">
Пример программы на языке Си:
</P>
<P CLASS="PROG">
**/*Прогнозируется поведение ракеты,
**
**в зависимости от ее начальной скорости*/
**
#include <stdio.h>
**
**
main()
**
**
{
**
**

```
float V;
<BR>
printf("Введите V\n");
<BR>
scanf("%f",&V);
<BR>
if (V<7.8)<BR>
printf("Ракета упадет на Землю"\n");
<BR>
else
<BR>
 if (V<11.2)
<BR>
printf("Ракета станет спутником Земли\n");
<BR>
else
<BR>
if (V<16.4)
<BR>
printf("Ракета станет спутником Солнца\n");
<BR>
else
<BR>
printf("Ракета покинет Солнечную систему\n");
}
</P>
</BODY>
</HTML>
```

Если сравнить этот код web-страницы с предыдущей версией, то можно заметить, что изменения в самом коде являются сравнительно небольшими, но при просмотре в любом браузере внешний вид данной страницы изменится достаточно существенно. На рис. 4 показано, как будет выглядеть в браузере Firefox начало страницы.

На рис. 5, 6 и 7 показаны фрагменты этой же web-страницы, содержащие программные коды для языков Бейсик, Пайтон и Си.

При просмотре приведенных фрагментов web-страницы становится видно, что здесь использованы три разных стиля оформления параграфов. Один из них – это тот, который применяется к параграфам по умолчанию и прописан в правиле для селектора P, а два других определяются специальными классами, прописанными в каскадной таблице стилей.

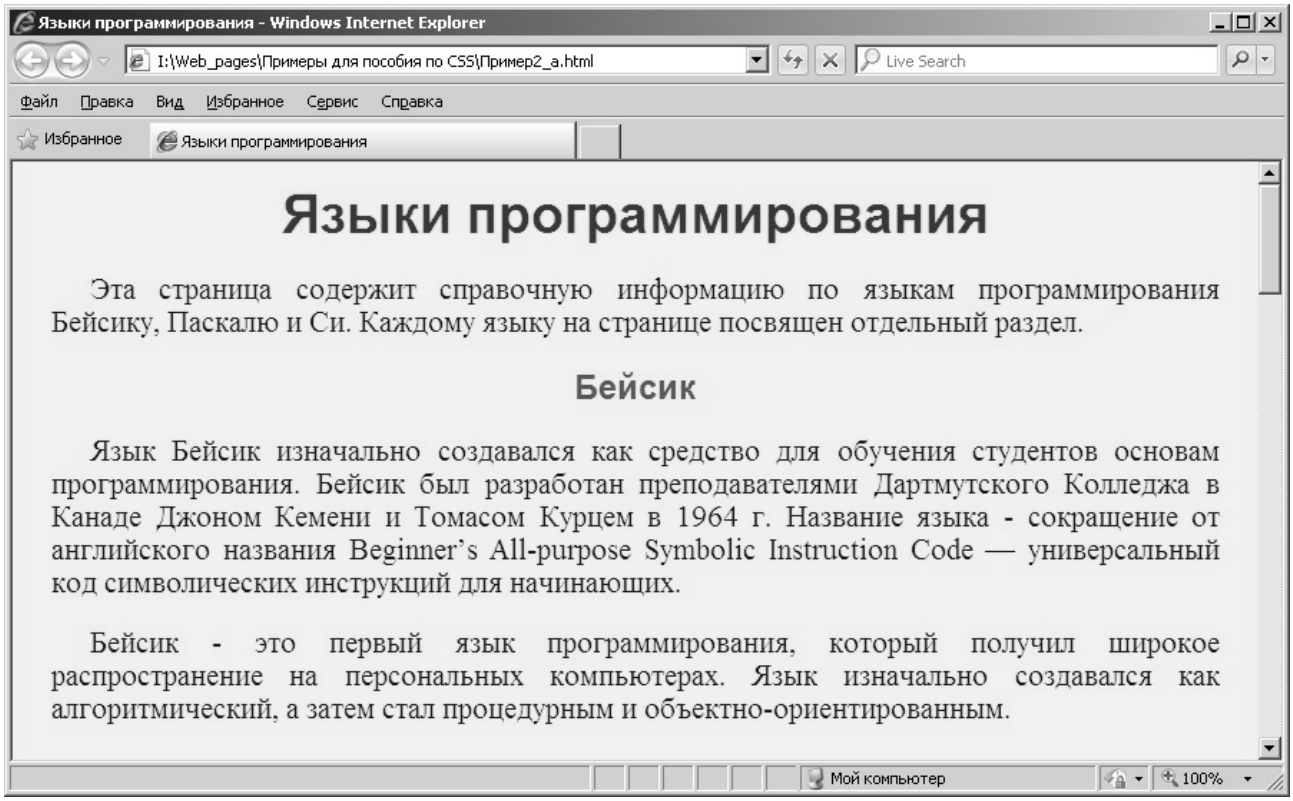

Рис. 4. Начало страницы, посвященной языкам программирования (после подключения каскадной таблицы стилей)

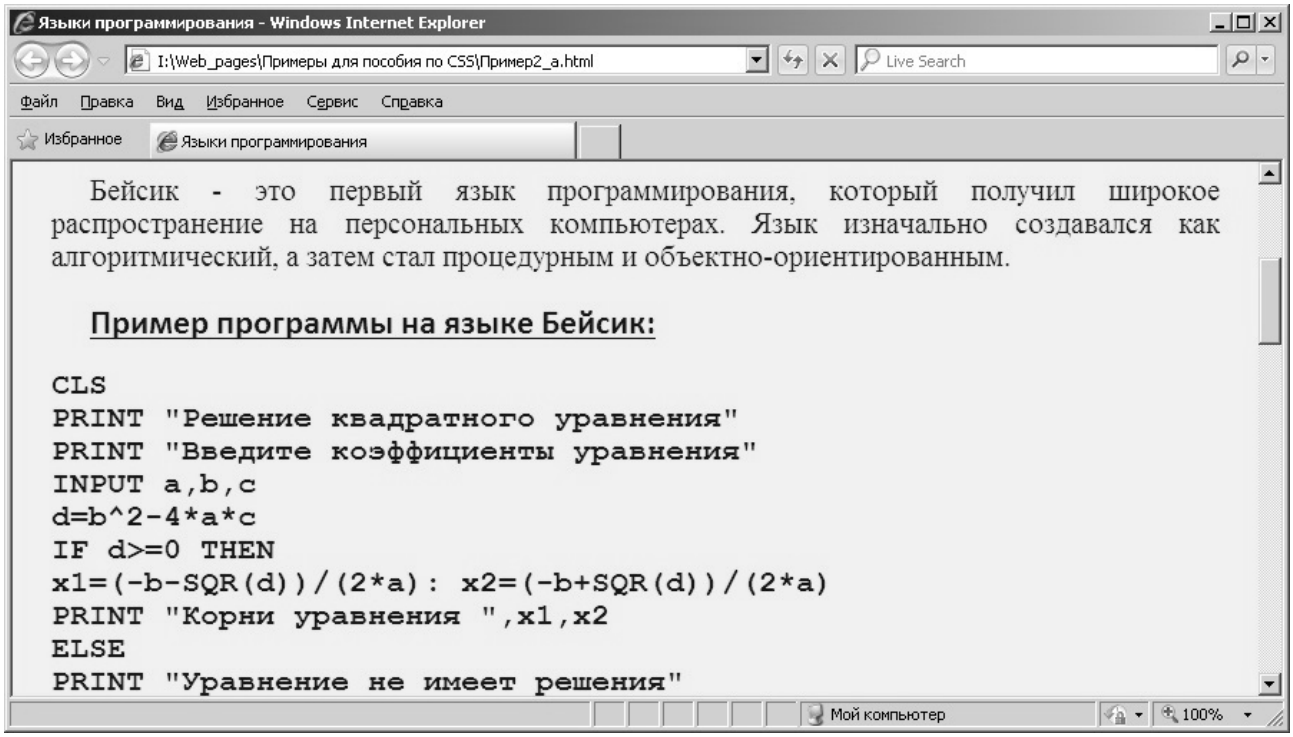

Рис. 5. Фрагмент страницы, содержащей текст программы на языке Бейсик (после подключения каскадной таблицы стилей)

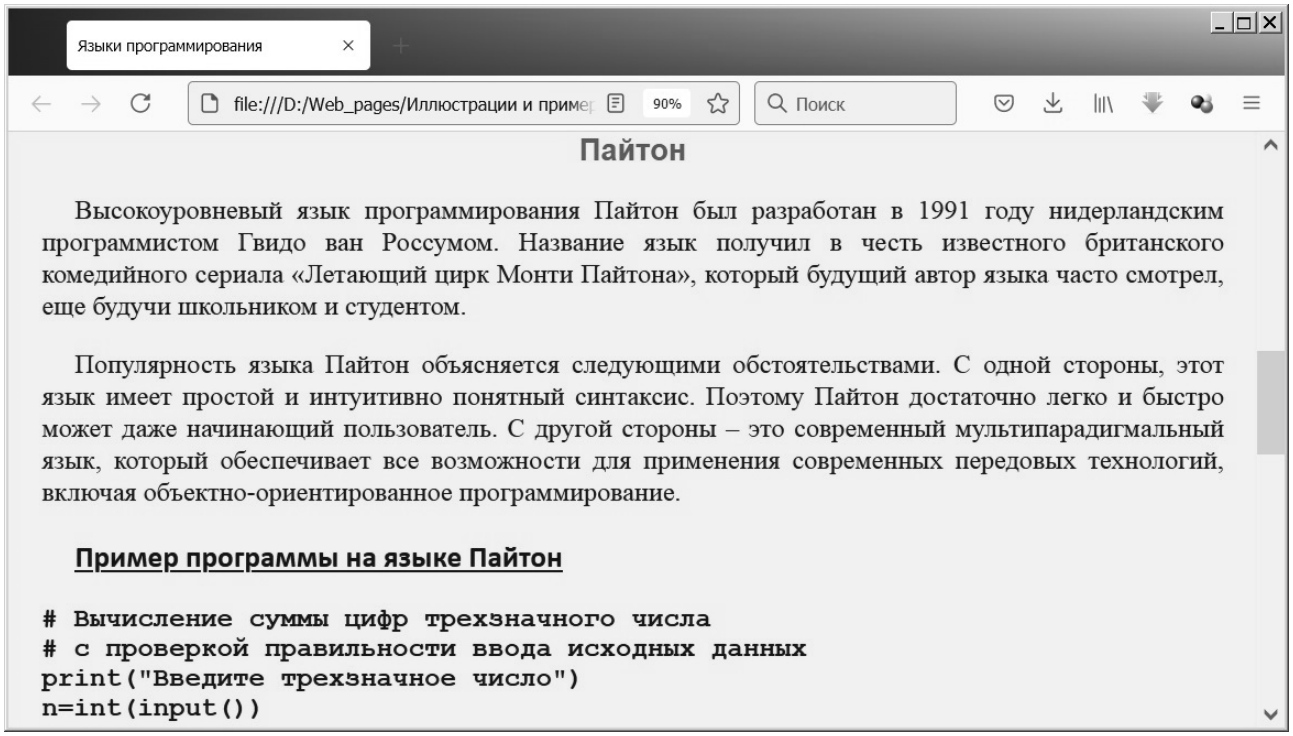

Рис. 6. Фрагмент страницы, содержащей текст программы на языке Пайтон (после подключения каскадной таблицы стилей)

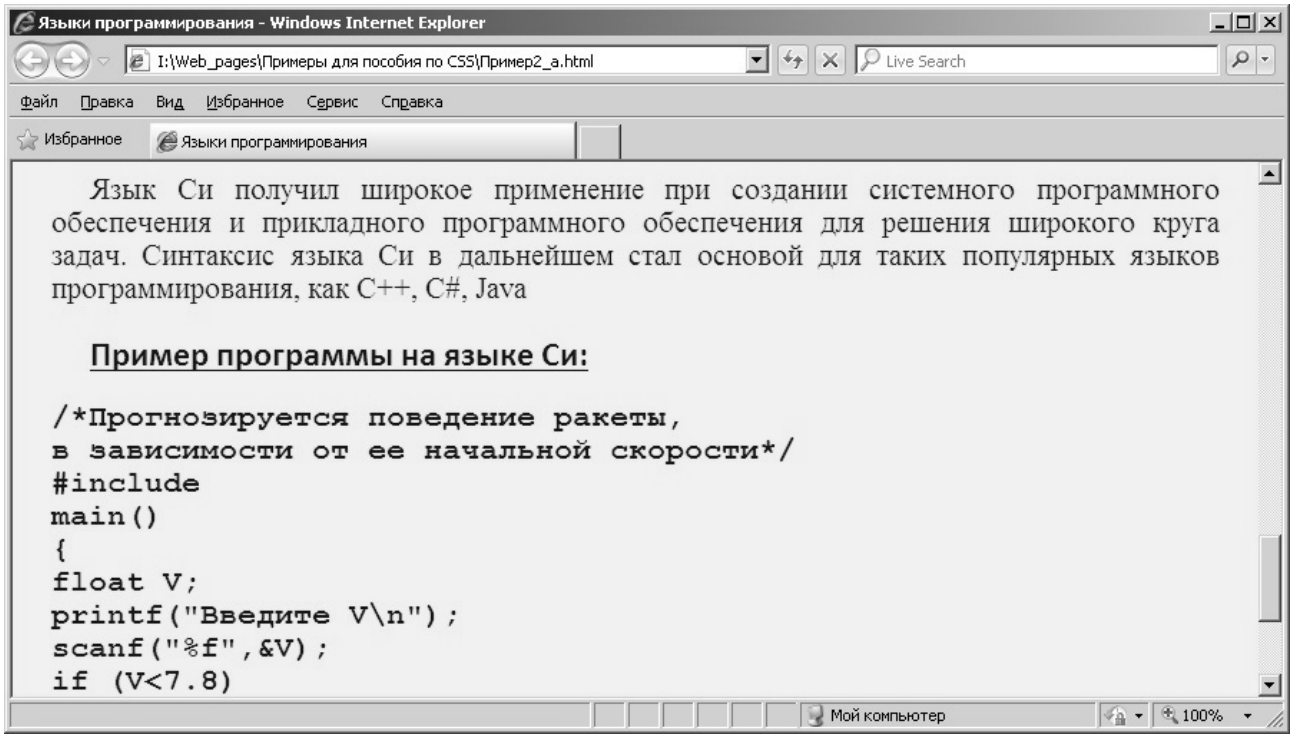

Рис. 7. Фрагмент страницы, содержащей текст программы на языке Си (после подключения каскадной таблицы стилей)

Это уже упомянутые ранее стили, используемые для оформления фрагментов программного кода на различных языках, которые приведены на web-странице, и заголовков к этому коду. Причем следует обратить внимание, что правило для класса имеет приоритет над обычным правилом, приведенным в таблице стилей, т. е. общее правило для элемента (в данном случае для параграфа) действует только там, где нет указания на использование конкретного класса.

Таким образом, очевидно, что использование классов существенно расширяет возможности каскадных таблиц стилей, позволяя задавать различные варианты оформления для элементов HTML, относящихся к одному и тому же типу.

Контрольные вопросы

1. Для каких целей в каскадных таблицах стилей применяются описания классов?
2. Для чего в названии класса используется точка?
3. Требует ли использование классов внесения дополнительных изменений в HTML-код web-страницы?
4. Могут ли в таблицах стилей для разных селекторов использоваться одни и те же свойства?
5. Какое свойство в каскадных таблицах стилей используется для изменения жирности шрифта?
6. Какое свойство в каскадных таблицах стилей используется для придания шрифтам специальных эффектов?
7. Можно ли в каскадных таблицах стилей создавать несколько классов для одного и того же HTML-элемента?
8. Что в каскадных таблицах стилей имеет приоритет: обычное правило или правило для какого-либо класса?

ГЛАВА 3. ОФОРМЛЕНИЕ СПИСКОВ РАЗЛИЧНЫХ ВИДОВ

Одним из распространенных способов упорядочения информации, представленной на web-страницах, являются списки. Списки могут быть нумерованными и маркированными, а также списками определений. В нумерованном списке каждый отдельный элемент обозначается цифрой или другим символом, а в маркированном списке для обозначения элементов применяются специальные значки-маркеры, которые могут иметь разную геометрическую форму. Также в качестве маркеров могут использоваться различные рисунки небольшого размера.

В языке HTML имеются необходимые теги для создания нумерованных и маркированных списков, а с помощью дополнительных атрибутов можно задать некоторые параметры оформления подобных списков. Но практика web-разработки показывает, что более эффективным способом оформления списков является использование каскадных таблиц стилей.

Во-первых, каскадные таблицы стилей предоставляют более обширный и разнообразный набор средств, которые можно применить для оформления различных списков.

Во-вторых, один раз разработав таблицу стилей, содержащую варианты оформления списков различных видов, можно ее в дальнейшем применять для оформления многих web-страниц.

Сначала рассмотрим основные свойства, применяемые в каскадных таблицах стилей для работы со списками, и их возможные значения.

Свойство **list-style-type** определяет внешний вид символа, применяемого для нумерации, или маркера. Это свойство может принимать следующие основные значения:

disk – в качестве маркера элемента списка выступает сплошной кружок;

circle – в качестве маркера элемента списка выступает кольцо, т. е. окружность с незакрашенной внутренней полостью;

decimal – элементы списка нумеруются обычными цифрами;

decimal-leading-zero – элементы списка нумеруются цифрами с ведущим нулем, т. е. 01, 02, 03, 04, 05 и т. д.;

lower-alpha – элементы списка обозначаются строчными буквами латинского алфавита;

lower-greek – элементы списка обозначаются строчными буквами греческого алфавита;

lower-roman – элементы списка нумеруются римскими цифрами в нижнем регистре, т. е. I, II, III, IV, V и т. д.;

none – номер элемента или маркер отсутствует;

square – в качестве маркера элемента списка используется квадрат;

upper-alpha – элементы списка обозначаются прописными буквами латинского алфавита;

upper-roman – элементы списка нумеруются римскими цифрами в верхнем регистре.

Свойство **list-style-image** применяется для использования в качестве маркера какого-либо изображения. Это свойство должно иметь значение **url**(имя). Под именем понимается имя того графического файла (как правило, в «компактном» формате **gif** или **jpg**, подходящем к использованию во всемирной сети), в котором содержится изображение маркера. При этом, если графический файл находится в той же папке, что web-страница и каскадная таблица стилей, то достаточно будет указать собственное имя файла. В противном случае обязательно требуется также полное указание пути к данному графическому файлу.

Свойство **list-style-position** определяет расположение маркера относительно списка. Оно может иметь следующие основные значения:

outside – маркер располагается снаружи списка. Это значение назначается для списка по умолчанию;

inside – маркер располагается внутри списка, т. е. текст списка будет обтекаться маркером.

Теперь используем эти свойства для оформления web-страницы, содержащей информацию об устройстве компьютера и свойствах его различных компонентов. На начальном этапе создадим файл следующего вида, содержащий только «чистый» HTML-код.

```
<HTML>
<HEAD>
<TITLE>
Устройство персонального компьютера
</TITLE>
</HEAD>
<BODY>
<H3> В состав персонального компьютера входят </H3>
<OL>
<LI> Системный блок
<LI> Монитор или дисплей
<LI> Клавиатура
<LI> Координатное устройство
<LI> Принтер
<LI> Сканер
<LI> Другие устройства
</OL>

<H3> В системном блоке содержатся следующие устройства </H3>
<OL>
<LI> Материнская плата
<LI> Процессор
<LI> Оперативная память
```

<LI> Видеокарта
<LI> Блок питания
<LI> Жесткий диск (винчестер)
<LI> Дисковод для лазерных дисков
</OL>
<H3> Материнская плата содержит в своем составе </H3>
<OL>
<LI> сокет (разъем) для установки центрального процессора;
<LI> слоты для оперативного запоминающего устройства (ОЗУ);
<LI> постоянное запоминающее устройство (ПЗУ);
<LI> постоянное перепрограммируемое запоминающее устройство (ППЗУ);
<LI> слоты для видеокарты и других плат расширения;
<LI> коннекторы для подключения жесткого диска и привода для лазерных дисков;
<LI> разъемы для питания материнской платы
<LI> встроенные USB-порты
<LI> чипсет, в котором интегрированы северный и южный мост
<LI> другие устройства.
</OL>

<P> Скорость и работоспособность компьютера в значительной части определяются характеристиками центрального процессора</P>
<H3> Основными характеристиками центрального процессора ПК являются</H3>
<OL>
<LI> Внутренняя тактовая частота
<LI> Внешняя тактовая частота
<LI> Разрядность
<LI> Размер кэш-памяти
</OL>
<P> Важнейшим устройством для вывода информации, имеющейся в персональном компьютере, является монитор</P>
<H3> Основными характеристиками монитора персонального компьютера являются</H3>
<UL>
<LI> соотношение сторон экрана;
<LI> размер экрана по диагонали;
<LI> разрешение;
<LI> глубина цвета;
<LI> частота обновления экрана;
<LI> угол обзора;
</UL>
<P>Основным устройством для ввода текста в персональный компьютер является клавиатура</P>

<H3> На клавиатуре персонального компьютера можно выделить следующие блоки </H3>

<UL>

<LI> основная алфавитно-цифровая клавиатура;

<LI> дополнительная цифровая клавиатура;

<LI> клавиши управления курсором и редактирования;

<LI> функциональные клавиши;

<LI> клавиши-модификаторы;

<LI> специализированные клавиши;

</UL>

<P> Персональный компьютер часто работает во взаимодействии с различными периферийными устройствами. Одним из самых широко применяемых устройств является принтер.</P>

<H3> Основными характеристиками современного принтера являются</H3>

<UL>

<LI> скорость печати (в страницах в минуту)

<LI> разрешение принтера (в точках на дюйм)

<LI> объем памяти принтера

<LI> формат документа, который может быть напечатан на принтере

<LI> наличие цветной печати

</UL>

</BODY>

</HTML>

Данная web-страница содержит 7 списков – 4 нумерованных и 3 маркированных. Вид списков такой, какой используется в HTML по умолчанию, т. е. элементы нумерованных списков обозначены обычными цифрами, а маркированных – сплошными кружками.

Вид данной страницы при просмотре ее в браузере Firefox приведен на рис. 8 (нумерованные списки) и рис. 9 (маркированные списки).

Теперь создадим таблицу стилей **Style3.css**, в которой будут заданы следующие правила для оформления текста: основной цвет текста – синий, фоновый цвет – светло-зеленый, шрифт – Times New Roman, размер шрифта 16, начертание – полужирное. Для заголовков размер шрифта увеличим до 18 и будем использовать шрифт Arial. Правило для обычного параграфа предусматривает создание красной строки и выравнивание его по ширине. Для списков созданы семь различных классов, обеспечивающих их нестандартное оформление. Ниже приводится текст этой таблицы:

BODY {background: #CCFFCC; color: blue; font-size:16pt; font-weight:bold}

H3 {color: Red; font-family: Arial; font-size: 18pt}

P {font-family:Calibri,Verdana; Text-align: justify; Text-indent:20pt}

OL.ROME {list-style-type:upper-roman}

OL.UPP_LAT {list-style-type:upper-alpha}
OL.LOW_LAT {list-style-type:lower-alpha}
OL.GREEK{list-style-type:lower-greek}
UL.C {list-style-type: circle}
UL.S {list-style-type: square}
UL.P{list-style-image: url(pict.jpg)}

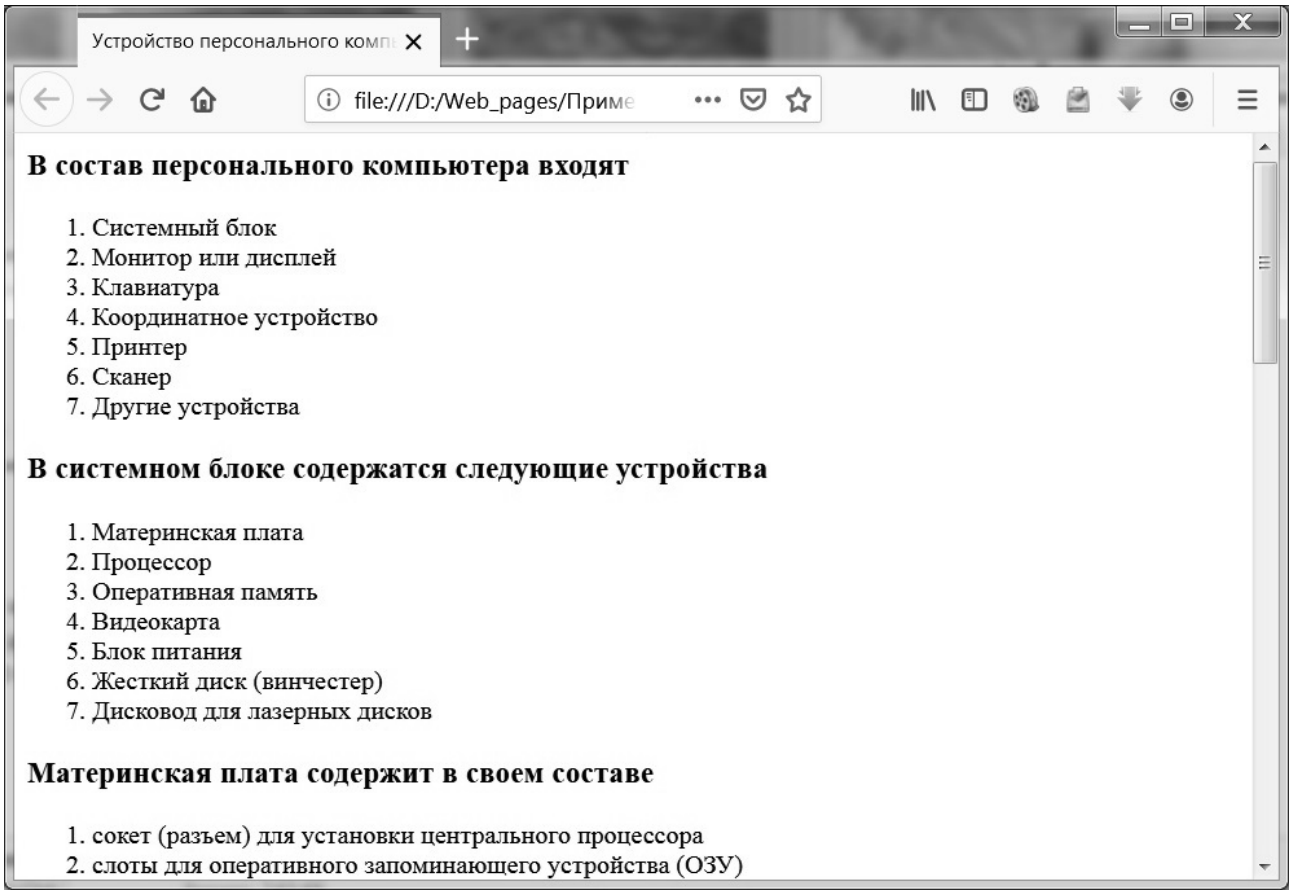

Рис. 8. Верхняя часть страницы, посвященной устройству компьютера, содержащая нумерованные списки

Далее внесем изменения в HTML-файл. В заголовочной части пропишем связь с вышеуказанной таблицей стилей с помощью тега **LINK**, а для каждого из семи списков, имеющихся в файле, сделаем ссылку на соответствующий класс из таблицы стилей. В итоге HTML-файл после необходимых дополнений будет выглядеть так, как показано ниже.

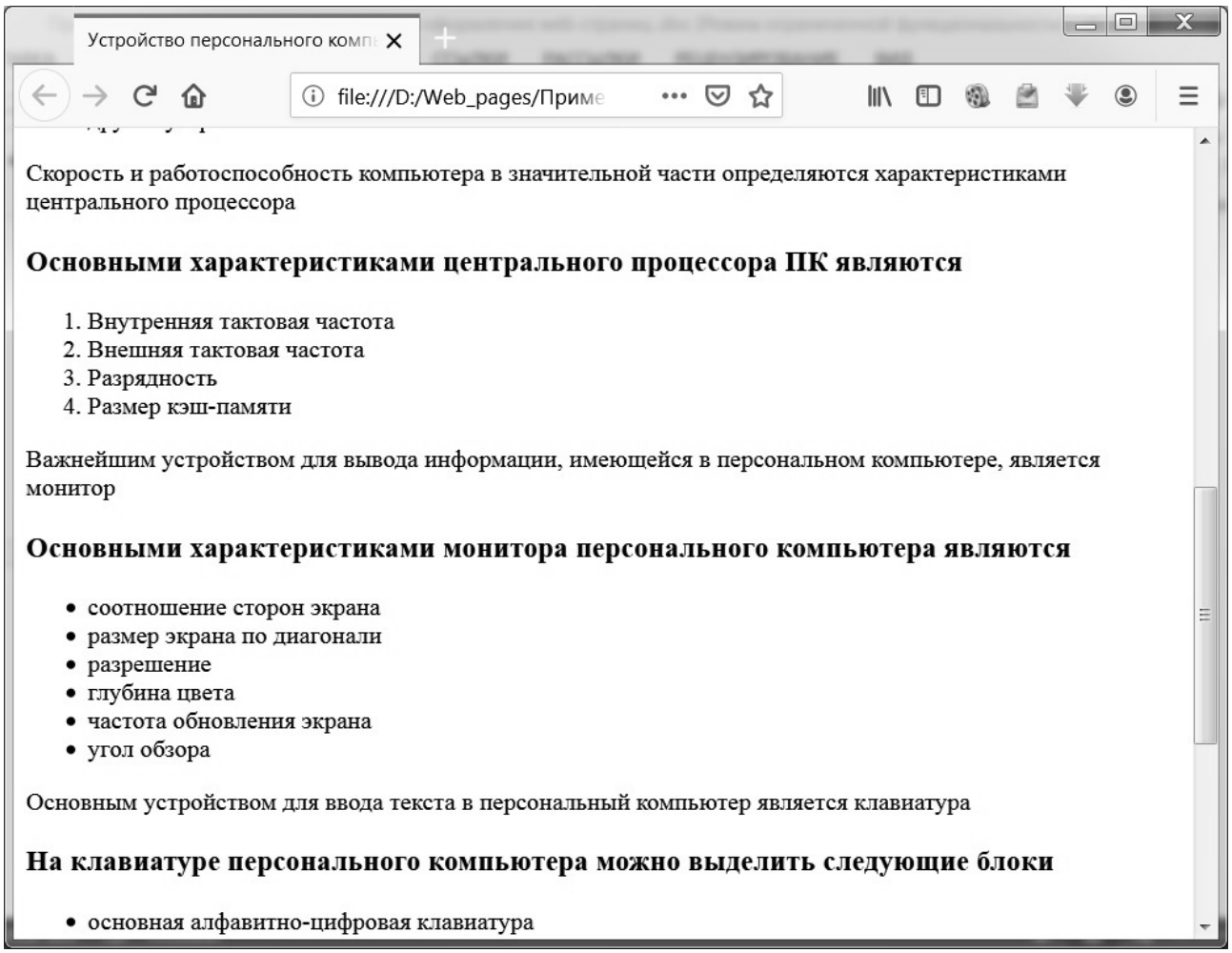

Рис. 9. Нижняя часть страницы, посвященной устройству компьютера, содержащая маркированные списки

```
<HTML>
<HEAD>
<TITLE>
Устройство персонального компьютера
</TITLE>
<LINK HREF="Style3.css" REL="STYLESHEET" TYPE="TEXT/CSS"
</HEAD>
<BODY>
<H3> В состав персонального компьютера входят </H3>
<OL CLASS="ROME">
<LI> Системный блок
<LI> Монитор или дисплей
<LI> Клавиатура
<LI> Координатное устройство
<LI> Принтер
```

<LI> Сканер
<LI> Другие устройства
</OL>
<H3>
В системном блоке содержатся следующие устройства
</H3>
<OL CLASS="UPP_LAT">
<LI> Материнская плата
<LI> Процессор
<LI> Оперативная память
<LI> Видеокарта
<LI> Блок питания
<LI> Жесткий диск (винчестер)
<LI> Дисковод для лазерных дисков
</OL>
<H3>
Материнская плата содержит в своем составе
</H3>
<OL CLASS="LOW_LAT">
<LI> сокет (разъем) для установки центрального процессора
<LI> слоты для оперативного запоминающего устройства (ОЗУ)
<LI> постоянное запоминающее устройство (ПЗУ)
<LI> постоянное перепрограммируемое запоминающее устройство (ППЗУ)
<LI> слоты для видеокарты и других плат расширения
<LI> коннекторы для подключения жесткого диска и привода для лазерных дисков
<LI> разъемы для питания материнской платы
<LI> встроенные USB-порты
<LI> чипсет, в котором интегрированы северный и южный мост
<LI> другие устройства
</OL>
<P>Скорость и работоспособность компьютера в значительной части определяются характеристиками центрального процессора</P>
<H3>
Основными характеристиками центрального процессора ПК являются
</H3>
<OL CLASS="GREEK">
<LI>Внутренняя тактовая частота
<LI>Внешняя тактовая частота
<LI>Разрядность
<LI>Размер кэш-памяти
</OL>

<P>Важнейшим устройством для вывода информации, имеющейся в персональном компьютере, является монитор</P>

<H3>

Основными характеристиками монитора персонального компьютера являются

</H3>

<UL CLASS="C">

<LI> соотношение сторон экрана

<LI> размер экрана по диагонали

<LI> разрешение

<LI> глубина цвета

<LI> частота обновления экрана

<LI> угол обзора

</UL>

<P>Основным устройством для ввода текста в персональный компьютер является клавиатура</P>

<H3>

На клавиатуре персонального компьютера можно выделить следующие блоки

</H3>

<UL CLASS="S">

<LI> основная алфавитно-цифровая клавиатура

<LI> дополнительная цифровая клавиатура

<LI> клавиши управления курсором и редактирования

<LI> функциональные клавиши

<LI> клавиши-модификаторы

<LI> специализированные клавиши

</UL>

<P> Персональный компьютер часто работает во взаимодействии с различными периферийными устройствами. Одним из самых широко применяемых устройств является принтер.</P>

<H3> Основными характеристиками современного принтера являются</H3>

<UL CLASS="P">

<LI> скорость печати (в страницах в минуту)

<LI> разрешение принтера (в точках на дюйм)

<LI> объем памяти принтера

<LI> формат документа, который может быть напечатан на принтере

<LI> наличие цветной печати

</UL>

</BODY>

</HTML>

В результате каждый из имеющихся на web-странице 4 видов нумерованных списков будет пронумерован своим, индивидуальным образом: первый список – римскими цифрами, второй – прописными латинскими буквами, третий – строчными латинскими буквами, четвертый – строчными греческими буквами. На рис. 10 показан внешний вид первого и второго списка при его просмотре в браузере Firefox. На рис. 11 представлен вид третьего и четвертого списка при просмотре в том же браузере.

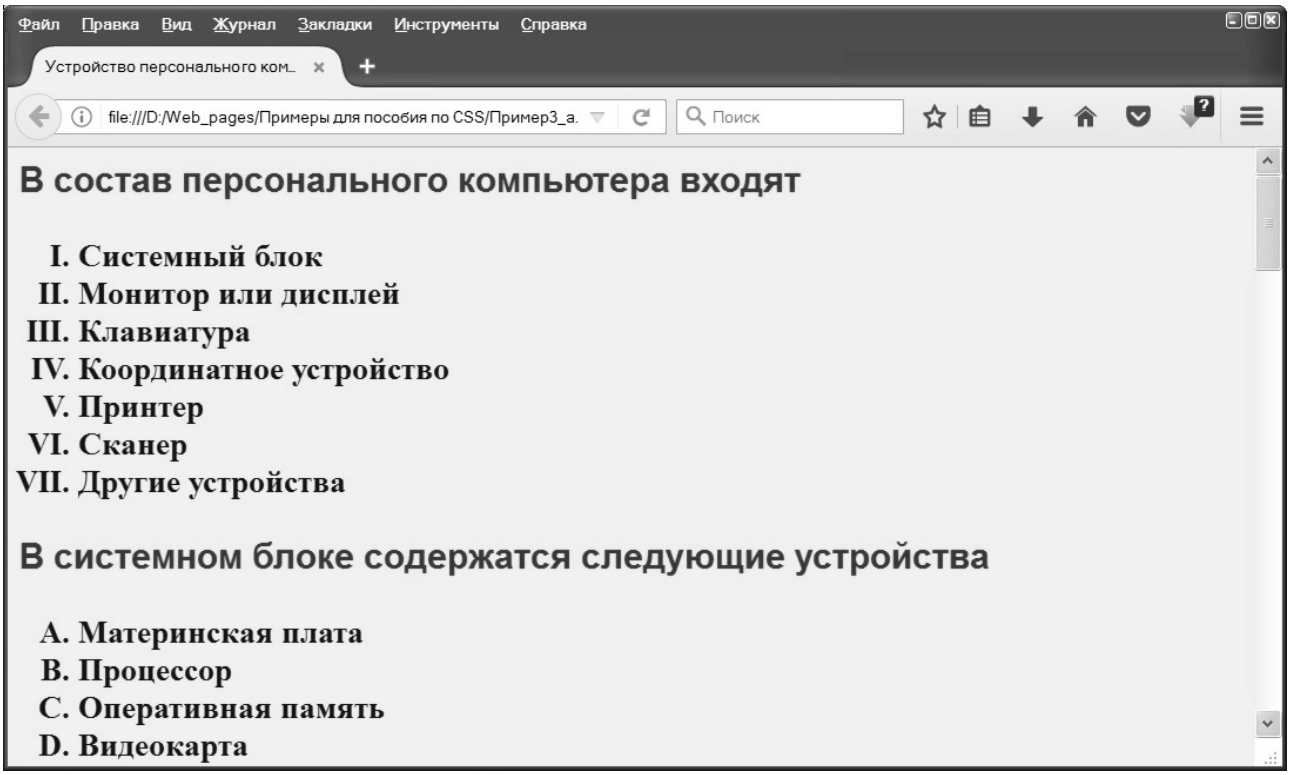

Рис. 10. Верхняя часть страницы, посвященной устройству персонального компьютера (после подключения каскадной таблицы стилей)

Также индивидуальным образом на странице оформлены все три нумерованных списка: первый список – маркерами в виде колец, второй – в виде квадратов, а для третьего в качестве маркеров использован рисунок в виде четырех ромбов. Рисунок для последнего вида маркеров сохранен в файле под названием **pict.jpg**. Этот файл находится в той же папке, что и web-страница, и поэтому указания полного пути к нему в свойстве **list-style-image** не требуется. На рис. 12 показан внешний вид маркированных списков при их просмотре в браузере.

Таким образом, приведенные в этой главе примеры показывают, что использование свойств каскадных таблиц стилей позволяет нестандартным и эстетически привлекательным образом оформить списки различных видов, которые имеются на web-страницах.

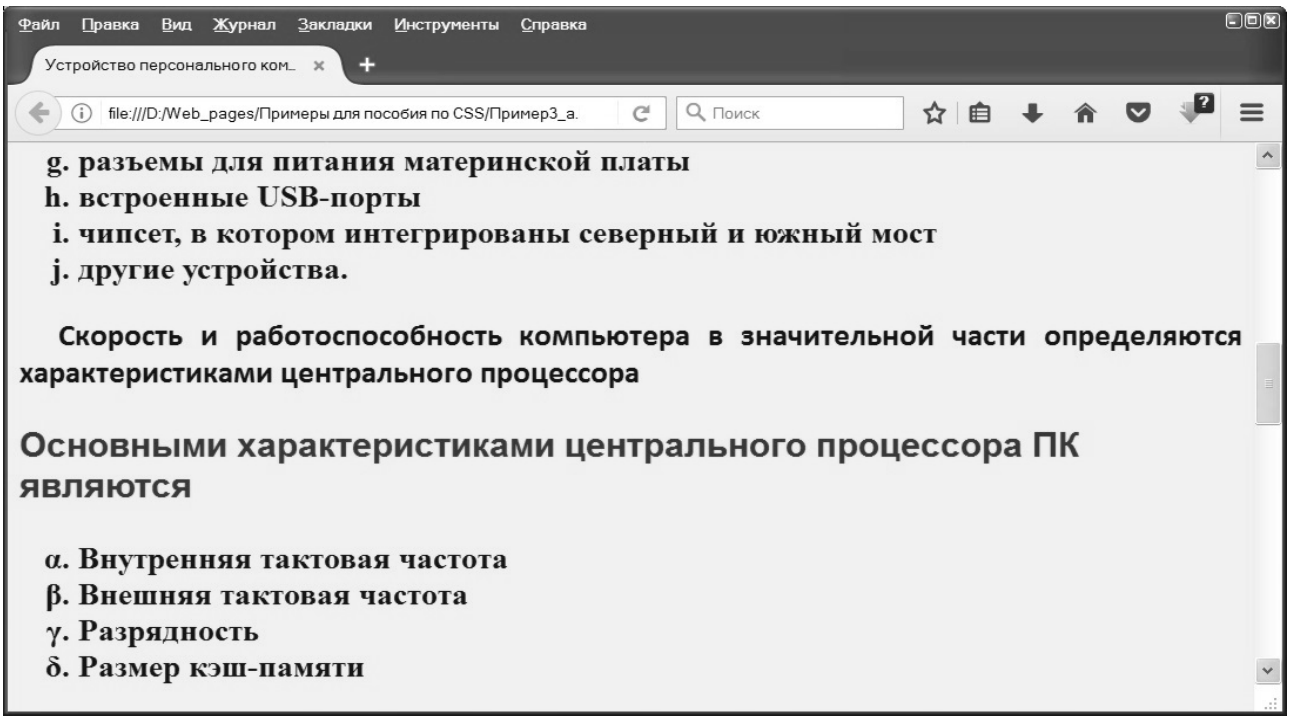

Рис. 11. Центральная часть страницы, посвященной устройству персонального компьютера (после подключения каскадной таблицы стилей)

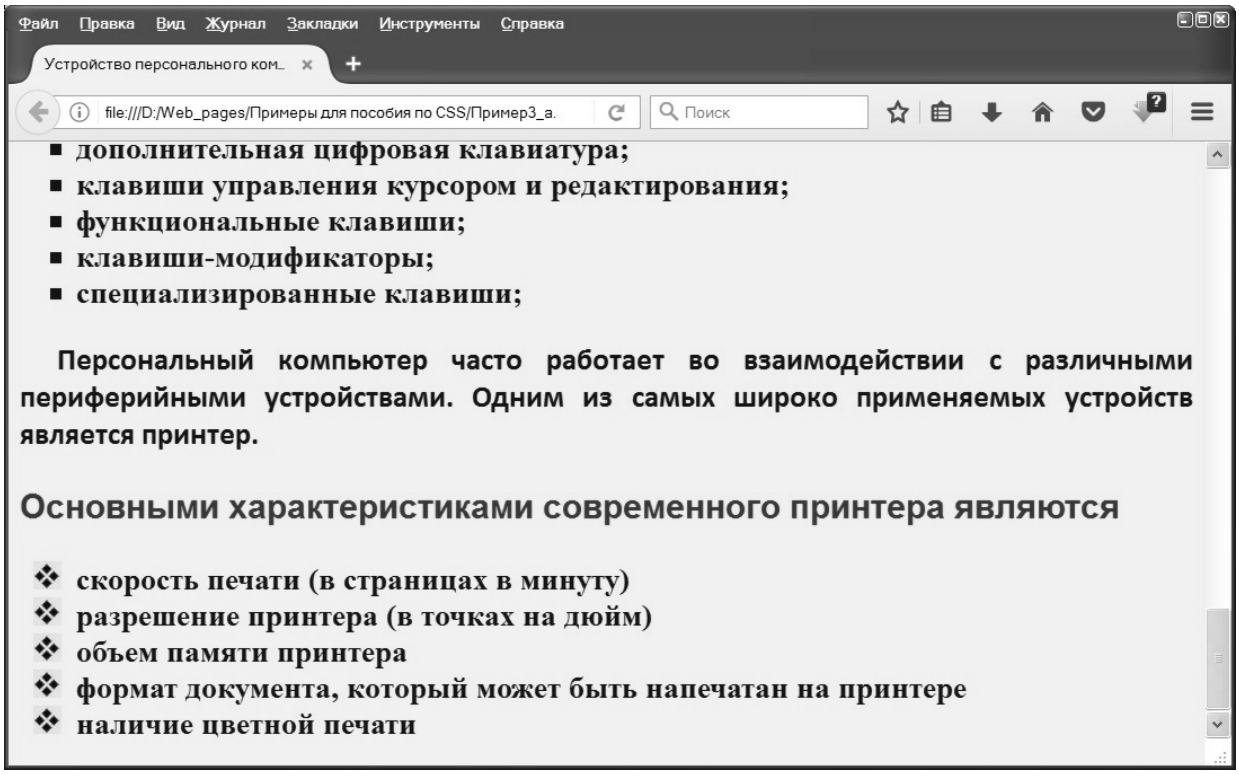

Рис. 12. Нижняя часть web-страницы, посвященной устройству персонального компьютера (после подключения каскадной таблицы стилей)

Контрольные вопросы

1. Какие основные виды списков могут оформляться при помощи каскадных таблиц стилей?
2. Какое свойство в таблице стилей отвечает за форму маркера списка?
3. Какое свойство в каскадной таблице стилей отвечает за вид символов в нумерованных списках?
4. Какое свойство в каскадной таблице стилей определяет положение маркера внутри или снаружи списка?
5. Какое свойство каскадной таблицы стилей позволяет использовать изображение в качестве маркера?
6. Какое значение соответствующего свойства в каскадной таблице стилей позволяет использовать при нумерации римские цифры?
7. Какое значение соответствующего свойства в каскадной таблице стилей позволяет использовать при нумерации прописные латинские буквы?
8. Какое значение соответствующего свойства в каскадной таблице стилей позволяет использовать при нумерации строчные латинские буквы?
9. Какое значение соответствующего свойства в каскадной таблице стилей позволяет использовать при нумерации строчные греческие буквы?
10. Какое значение соответствующего свойства в каскадной таблице стилей позволяет использовать маркеры в форме кольца?
11. Какое значение соответствующего свойства в каскадной таблице стилей позволяет использовать маркеры квадратной формы?

ГЛАВА 4. ОФОРМЛЕНИЕ ТАБЛИЦ

Одним из важнейших средств, часто применяемых на web-страницах, являются таблицы. Они не только позволяют представить в наглядном и упорядоченном виде различные данные, но и широко используются как средство для создания структуры страниц.

В языке HTML существует набор тегов, которые позволяют задать заголовок таблицы, определить количество строк и столбцов таблицы, задать размеры полей внутри ячеек, объединить несколько ячеек в одну и выполнить другие действия, определяющие основные характеристики таблицы. В то же время возможности HTML по оформлению внешнего вида таблиц в значительной степени ограничены. Например, нельзя задать для таблиц произвольный цвет рамки, отличный от стандартного.

Но возможности языка CSS по оформлению таблиц значительно шире. Перечислим основные свойства, которые позволяют дополнить возможности HTML, придав таблице красивый и эффектный внешний вид, а главное, облегчить и улучшить восприятие разнообразной информации, представленной в табличной форме.

Свойство **text-align** – используется для выравнивания текста таблицы по горизонтали; может принимать значения **center**, **left** и **right**. Применяется по отношению к элементам <**TD**> и <**TH**>, а также <**CAPTION**>, т. е. можно задавать способ выравнивания текста для ячеек, строк и для заголовка таблицы.

Свойство **vertical-align** позволяет выравнивать содержимое ячеек таблицы по вертикали. Это свойство может принимать следующие значения:

top – выравнивает содержимое ячейки по ее верхнему краю; это значение применяется по умолчанию;

middle – выравнивает содержимое ячейки по ее центру;

bottom – выравнивает содержимое ячейки по ее нижнему краю.

Свойство **border-style** определяет вид границы таблицы. Оно может принимать следующие основные значения:

none или **hidden** – рамка отсутствует; это значение указанное свойство имеет по умолчанию;

dotted – пунктирная линия;

dashed – штриховая линия;

solid – сплошная линия;

double – двойная линия.

Свойство **border-color** определяет цвет рамки таблицы или границ между ячейками. Это свойство следует описывать после свойства **border-style.** Значение данного свойства, подобно фоновому цвету страницы или цвету текста, может определяться двумя способами:

а) можно выбрать подходящую цветовую константу (название цвета на английском языке) из набора, поддерживаемого языком HTML;

б) можно указать значение в виде шестнадцатеричного числового кода.

Свойство **border-width** определяет ширину границы таблицы, а также границ между ячейками. Это свойство может принимать следующие фиксированные значения:

thin – создает тонкую рамку;

medium – создает рамку средней толщины;

thick – создает толстую рамку.

При разработке web-страниц следует понимать, что все эти величины являются относительными. Но существует и альтернативная возможность задавать толщину рамки явным образом, устанавливая это значение в пикселях. Тогда указание значения свойства может выглядеть, например, так:

border-width: **5px**

В данном случае задается рамка толщиной 5 пикселей.

Свойство **border-collapse** определяет стиль границ между ячейками в таблице. Оно используется по отношению ко всей таблице в целом и может принимать следующие значения:

separate – в этом случае каждая ячейка внутри таблицы окружается отдельной рамкой; это значение применяется в таблицах по умолчанию;

collapse – отдельной рамки для каждой ячейки нет, но внутри таблицы создаются разделительные линии, которые отделяют ячейки друг от друга по горизонтали и по вертикали. При использовании данного значения таблицы имеют более привычный для пользователя вид.

Свойство **padding** задает размеры полей внутри ячейки, т. е. расстояние между границами ячейки и текстом, находящимся внутри ячейки. Эта величина может измеряться в сантиметрах, пунктах или пикселях.

Свойство **table-layout** определяет, могут ли изменяться размеры таблицы и отдельных ячеек. Данное свойство применяется ко всей таблице в целом и может иметь следующие значения:

auto – в этом случае браузер может изменять размеры таблицы в целом и отдельных ее ячеек в зависимости от габаритов экрана, на котором таблица отображается. Это значение данное свойство имеет по умолчанию;

fixed – размеры таблицы и ее ячеек являются фиксированными и никогда не будут изменяться.

Свойство **empty-cells** определяет, каким образом на экран будут выводиться пустые ячейки, не имеющие содержания. Это свойство применяется по отношению ко всей таблице в целом и может принимать следующие значения:

hide – пустые ячейки не будут выводиться на экран компьютера;

show – пустые ячейки будут отображаться на экране компьютера. Причем, если эти ячейки имеют рамку и фоновый цвет, отличный от основного цвета web-страницы, то при просмотре таблицы в браузере данные рамки и цвет также будут видны на экране.

Кроме того, задавая правила для таблицы, ряда или ячейки можно использовать уже знакомые нам по главе 1 данного пособия свойства, определяющие параметры шрифта. Эти параметры могут применяться для отображения текста, содержащегося в таблице. К ним относятся такие свойства как **font-family**, **font-size**, **font-weight** и **color**. Также для определения цветового фона таблицы в целом или отдельной ячейки может использоваться рассмотренное ранее свойство **background-color.**

В качестве примера рассмотрим создание web-страницы, посвященной основным парадигмам (стилям) программирования. Эта страница содержит заголовок, два абзаца обычного текста и таблицу, содержащую два столбца и семь строк. Ниже приводится исходный HTML-код данной страницы:

```
<HTML>
<HEAD>
<TITLE>
Стили программирования
</TITLE>
</HEAD>
<BODY>
<H1>
Парадигмы программирования
</H1>
<P>
Данная страница посвящена рассмотрению существующих парадигм программирования. Парадигмой программирования называется совокупность идей и понятий, которые определяют стиль написания компьютерных программ. Знание основных парадигм, умение в них ориентироваться, выбрать подходящую парадигму и соответствующий язык программирования для решения конкретной задачи, безусловно, является необходимым для специалиста в области информационных технологий.
</P>
<P>
С целью облегчения данной задачи на странице представлена таблица, содержащая два столбца. В левом столбце содержится название парадигмы, а в правом ее краткая характеристика
</P>
<TABLE>
<TR>
<TD>
Название парадигмы
</TD>
<TD>
Краткая характеристика
</TD>
```

</TR>
<TR>
<TD>
Императивная
</TD>
<TD>
В исходном коде записываются команды, которые должны выполняться последовательно. Императивная программа похожа на приказы (англ. imperative — приказ, повелительное наклонение), выражаемые повелительным наклонением в естественных языках
</TD>
</TR>
<TR>
<TD>
Декларативная
</TD>
<TD>
Задаётся спецификация решения задачи, то есть описывается, что представляет собой проблема и ожидаемый результат. В этом случае программа представляет собой формальную теорию, а её выполнение является одновременно автоматическим доказательством этой теории
</TD>
</TR>
<TR>
<TD>
Структурная
</TD>
<TD>
В основе парадигмы лежит представление программы в виде иерархической структуры блоков. В данном случае любая программа, состоит из трёх базовых управляющих конструкций: последовательность, ветвление, цикл; кроме того, используются подпрограммы
</TD>
</TR>
<TR>
<TD>
Логическая
</TD>
<TD>
парадигма программирования, основанная на автоматическом доказательстве теорем. Логическое программирование основано на теории и аппарате математической логики
</TD>
</TR>

<TR>
<TD>
Функциональная
</TD>
<TD>
парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании последних (в отличие от функций как подпрограмм в процедурном программировании)
</TD>
</TR>
<TR>
<TD>
Объектно-ориентированная
</TD>
<TD>
парадигма программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования
</TD>
</TR>
</TABLE>

</BODY>
</HTML>

На рис. 13 и рис. 14 показан внешний вид данной web-страницы при просмотре ее в браузере. Таблица стилей на данном этапе разработки страницы еще не была создана и не подключалась. Поэтому вся страница выполнена в черно-белой гамме и какие-либо дополнительные элементы оформления на странице отсутствуют. Таблица на странице имеется, но она не содержит никаких рамок, которые бы позволили определить ее границы. Текст таблицы отображается тем же самым шрифтом, что и основной текст web-документа. Текст заголовка таблицы также никак не выделен.

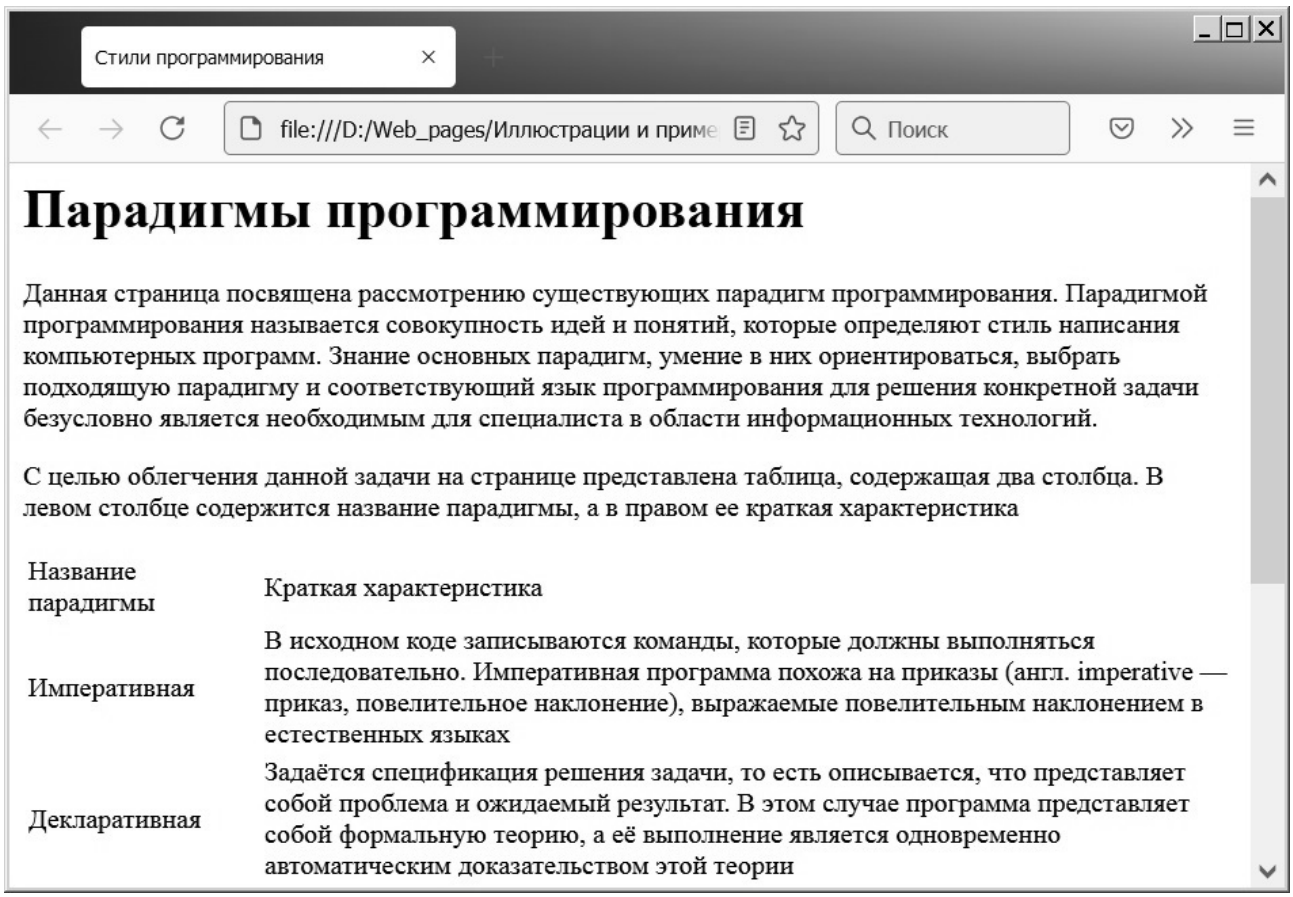

Рис. 13. Верхняя часть страницы, посвященной парадигмам программирования (только HTML-код)

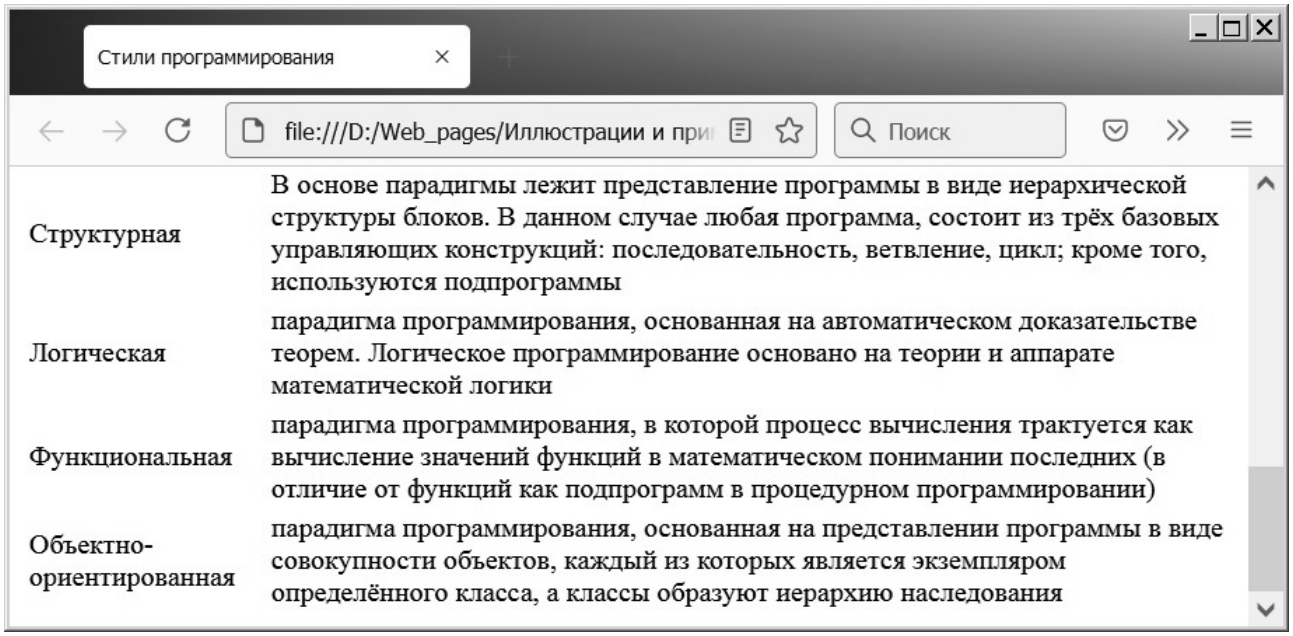

Рис. 14. Нижняя часть страницы, посвященной парадигмам программирования (только HTML-код)

Следующим шагом в оформлении web-документа должно стать создание таблицы стилей, в которой будет использоваться ряд из перечисленных в начале данного раздела правил для оформления таблиц.

Данная таблица стилей содержит правила для оформления заголовка страницы, ее фонового цвета и основного текста, включая гарнитуру шрифта, цвет шрифта, его размер, величину красной строки и полей абзаца. Поскольку аналогичные правила были описаны в предыдущих главах пособия, мы не будем на них подробно останавливаться.

Более детально разберем те правила, которые относятся непосредственно к таблице и ее отдельным элементам. Правило **TABLE** относится ко всей таблице в целом. В нем задается стиль рамки таблицы – сплошной и толщина этой рамки – 4 пикселя. Затем указывается цвет рамки – синий. Далее указывается, что вокруг каждой отдельной ячейки рамка создаваться не будет, но ячейки будут отделены друг от друга разделительными линиями. Таблица должна иметь отступы от края страницы шириной в 20 пунктов. Шрифт для текста, содержащегося в таблице, должен быть цвета красного дерева, размер его должен быть равен 15 пунктам. Начертание шрифта должно быть полужирным, и должна использоваться моноширинная гарнитура Courier New.

Отдельные правила предусмотрены для ячеек, находящихся в заголовке таблицы. Здесь задан укрупненный размер шрифта (18 пунктов), гарнитура шрифта Arial и в качестве фонового цвета выбран розовый. Также предусмотрено, что размеры полей внутри ячейки (расстояние между текстом и границей ячейки) должны составлять 10 пунктов. Задается сплошная граница для ячеек, ширина этой границы составляет 4 пикселя. Определяется и цвет рамки: синий, как и для внешней границы.

Следующее правило относится ко всем остальным ячейкам (кроме ячеек заголовка). Стиль границы, ее ширина и цвет, размеры полей внутри ячейки задаются так же, как для ячеек заголовка. Единственное отличие состоит в том, что не задается укрупненный размер шрифта.

Также в данной таблице стилей определяются еще два дополнительных класса для ячеек. Один из них определяет параметры ячеек для левого столбца таблицы, а другой – для правого столбца. Первый класс носит название **TAB1** и задает золотистый фоновый цвет для ячеек, размер шрифта в 16 пунктов и гарнитуру шрифта Arial. Второй класс называется **TAB2** и определяет светло-голубой фоновый цвет. Ниже приводится текст каскадной таблицы стилей:

```
BODY {background-color:#CCFFCC}
H1 {text-align: center;color:red; font-size:26pt; font-family:Arial}
P {font-family: "Times New Roman"; font-size: 14pt; color:navy; text-align: justify; text-indent: 20pt; margin-left:20pt; margin-right:20pt}
TABLE {border-style: solid; border-width: 4px; border-color: blue; border-collapse:collapse; margin-left: 20pt; margin-right: 20pt; color:maroon; font-family:"Courier New"; font-size:15; font-weight:700}
```

TH {border-style: solid; border-width: 4px; border-color:blue; padding:10pt; font-size:18; font-family:"Arial"; background-color:pink}

TD {border-style: solid; border-width: 4px; border-color:blue; padding:10pt}

TD.TAB1 {font-size:16; font-family:"Arial"; background-color:gold}

TD.TAB2 {background-color:lightblue}

Далее необходимо подключить каскадную таблицу стилей к HTML-коду web-страницы и внести необходимые изменения в код с учетом использования двух классов, описанных в этой таблице. Ниже приводится измененный HTML-код для данной страницы:

```
<HTML>
<HEAD>
<TITLE>
Стили программирования
</TITLE>
<LINK HREF="Style4.css" REL="Stylesheet" TYPE="Text/CSS">
</HEAD>
<BODY>
<H1>
Парадигмы программирования
</H1>
<P>
Данная страница посвящена рассмотрению существующих парадигм программирования. Парадигмой программирования называется совокупность идей и понятий, которые определяют стиль написания компьютерных программ. Знание основных парадигм, умение в них ориентироваться, выбрать подходящую парадигму и соответствующий язык программирования для решения конкретной задачи безусловно является необходимым для специалиста в области информационных технологий.
</P>
<P>
С целью облегчения данной задачи на странице представлена таблица, содержащая два столбца. В левом столбце содержится название парадигмы, а в правом ее краткая характеристика
</P>
<TABLE>
<TR>
<TH>
Название парадигмы
</TH>
<TH>
```

Краткая характеристика
</TH>
</TR>
<TR>
<TD CLASS="TAB1">
Императивная
</TD>
<TD CLASS="TAB2">
В исходном коде записываются команды, которые должны выполняться последовательно. Императивная программа похожа на приказы (англ. imperative — приказ, повелительное наклонение), выражаемые повелительным наклонением в естественных языках
</TD>
</TR>
<TR>
<TD CLASS=TAB1>
Декларативная
</TD>
<TD CLASS=TAB2>
Задаётся спецификация решения задачи, то есть описывается, что представляет собой проблема и ожидаемый результат. В этом случае программа представляет собой формальную теорию, а её выполнение является одновременно автоматическим доказательством этой теории
</TD>
</TR>
<TR>
<TD CLASS=TAB1>
Структурная
</TD>
<TD CLASS=TAB2>
В основе парадигмы лежит представление программы в виде иерархической структуры блоков. В данном случае любая программа, состоит из трёх базовых управляющих конструкций: последовательность, ветвление, цикл; кроме того, используются подпрограммы
</TD>
</TR>
<TR>
<TD CLASS=TAB1>
Логическая
</TD>
<TD CLASS=TAB2>
парадигма программирования, основанная на автоматическом доказательстве теорем. Логическое программирование основано на теории и аппарате математической логики

</TD>
</TR>
<TR>
<TD CLASS=TAB1>
Функциональная
</TD>
<TD CLASS=TAB2>
парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании последних (в отличие от функций как подпрограмм в процедурном программировании)
</TD>
</TR>
<TR>
<TD CLASS=TAB1>
Объектно-ориентированная
</TD>
<TD CLASS=TAB2>
парадигма программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования
</TD>
</TR>
</TABLE>
</BODY>
</HTML>

На рис. 15 и рис. 16 показан внешний вид web-страницы, посвященной парадигмам программирования после подключения к ней каскадной таблицы стилей при просмотре ее в браузере Firefox.

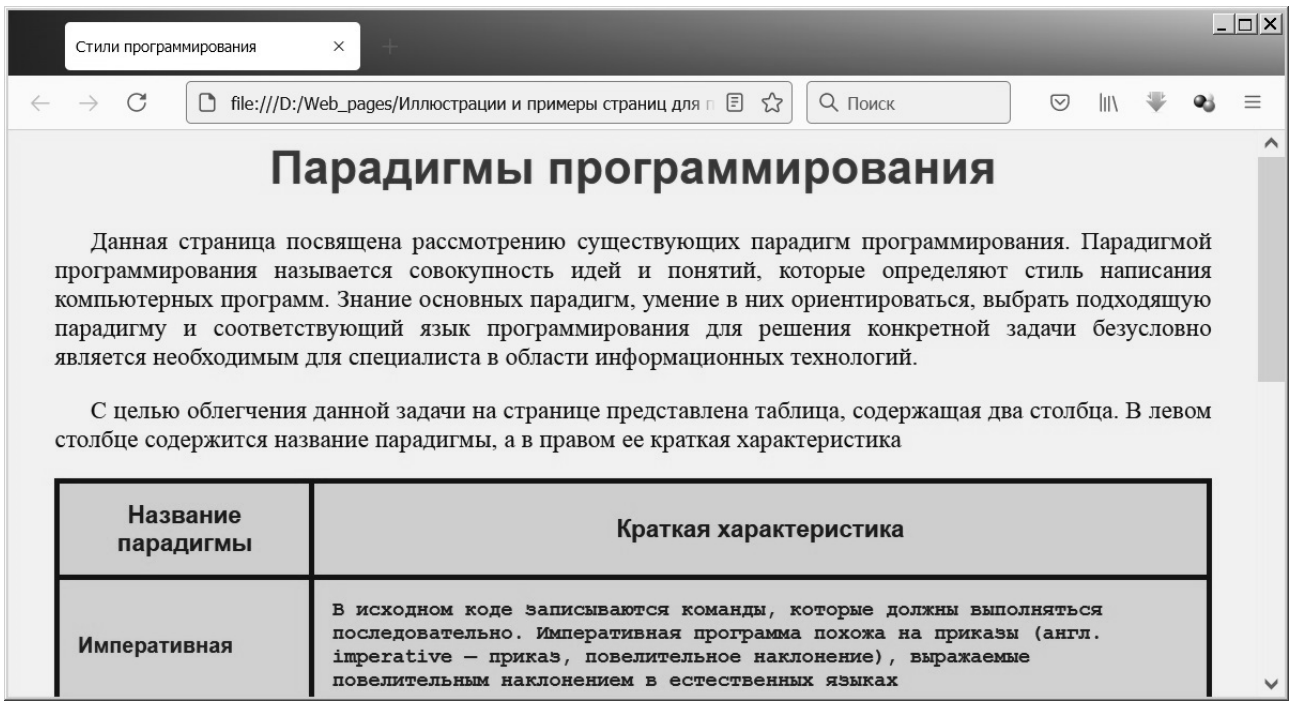

Рис. 15. Верхняя часть страницы, посвященной парадигмам программирования (таблица стилей подключена)

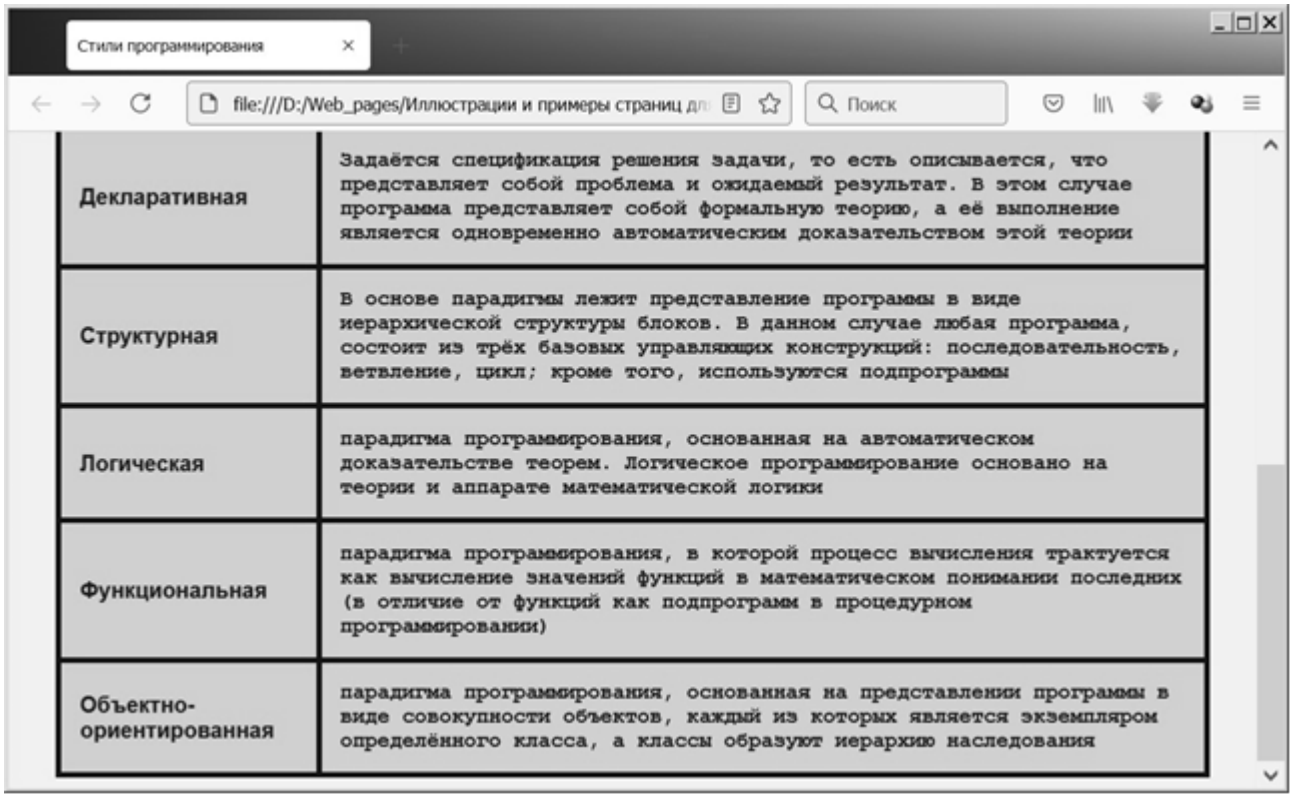

Рис. 16. Нижняя часть страницы, посвященной парадигмам программирования (таблица стилей подключена)

Даже при просмотре данных иллюстраций в черно-белом варианте видно отличие от рис. 13 и рис. 14. Видно, что шрифт основного текста отличается от шрифта заголовка и шрифта, которым набран текст таблицы. У таблицы в целом и отдельных ее ячеек теперь имеется рамка. Различные части таблицы выделены разными фоновыми цветами, каждая ячейка имеет внутренние поля. Как абзацы основного текста, так и таблица не соприкасается с границами страницы, поскольку имеют внешние поля.

Рассмотренный выше пример показывает, что использование средств языка каскадных таблиц стилей применительно к таблицам, имеющимся в web-документах, позволяет качественно улучшить их вид, сделать их удобным и полезным средством для представления больших массивов данных.

Контрольные вопросы

1. Какое свойство каскадных таблиц стилей позволяет определить способ выравнивания текста в ячейках таблицы?
2. Какое свойство каскадных таблиц стилей позволяет выравнивать содержимое ячеек в таблице по вертикали?
3. Какое свойство каскадных таблиц стилей определяет внешний вид границы в таблице?
4. Какое свойство каскадных таблиц стилей определяет цвет внешний рамки таблицы и границ между ячейками?
5. Какое свойство каскадных таблиц стилей определяет ширину внешней рамки таблицы и границ между ячейками?
6. Какое свойство каскадных таблиц стилей определяет стиль границ между ячейками?
7. Какое свойство каскадных таблиц стилей определяет расстояние между текстом и границами ячейки?
8. Какое свойство каскадных таблиц стилей определяет, могут ли изменяться размеры таблицы в целом и отдельных ее ячеек?
9. Какое свойство каскадных таблиц стилей определяет, будут ли отображаться на экране пустые ячейки таблицы?
10. Какие свойства, определяющие параметры шрифта, могут использоваться в правилах для таблиц и отдельных ячеек?

ГЛАВА 5. РАБОТА С БЛОКАМИ

Одним из распространенных в настоящее время подходов к разработке внешнего вида web-страниц является создание ее на основе блочных контейнеров, называемых также для краткости блоками. Блок создается с помощью парного тега языка HTML <**DIV**>…</**DIV**>, внутри которого могут располагаться самые различные элементы. К вложенным элементам могут относиться абзацы, заголовки, таблицы, гиперссылки, изображения и другие элементы. Эти элементы также записываются с помощью тегов языка HTML. Затем к такой странице подключается каскадная таблица стилей. Эта таблица определяет способы оформления как страницы в целом, так и присутствующих в ее составе отдельных блоков с помощью набора соответствующих правил.

Этот подход отчасти напоминает построение web-страницы на основе таблицы, но зачастую такая таблица получается достаточно сложной, поскольку ее ячейки должны также содержать вложенные таблицы. Использование же блочного принципа построения позволяет сделать код более простым и понятным, а функциональность страницы при этом нисколько не страдает.

Для того чтобы успешно использовать блоки при оформлении web-документов, нужно уметь ориентироваться в их свойствах. Ниже будут перечислены наиболее часто применяемые свойства, а к свойствам, специфическим для блоков относятся следующие.

Свойство **float** задает так называемые плавающие блоки, которые будут обтекаться текстом. Это свойство имеет следующие значения:

left – блок будет расположен слева, а текст будет обтекать его справа;

right – блок будет находиться справа, а текст будет обтекать его слева;

none – обтекание блока текстом отсутствует; это значение свойство float имеет по умолчанию.

С помощью этого свойства разработчик может определить подходящим для него образом положение блочного элемента на странице. Если не использовать данное свойство, то блочные элементы будут выстраиваться один под другим по вертикали. Но если мы для двух и более блоков зададим одинаковое значение свойство **float**, то они будут располагаться по-иному. Например, зададим трем блокам для свойства **float** значение **left.** Тогда первый блок будет прижиматься к левому краю страницы, второй блок будет располагаться справа от первого блока, а третий блок справа от второго. Таким образом, разработчик web-документа получает возможность размещать блочные элементы по горизонтали.

Свойство **clear** задает расположение текущего блочного элемента ниже предыдущих блоков. Это свойство может быть полезным в том случае, если предыдущие блоки были выровнены по горизонтали, а текущий надо разместить под ним по вертикали. Это свойство имеет следующие значения:

left – текущий блочный элемент должен располагаться ниже всех блочных элементов, у которых свойство float имеет значение left;

right – текущий блочный элемент должен располагаться ниже всех блочных элементов, у которых свойство float имеет значение right;

both – текущий блочный элемент должен располагаться ниже всех блочных элементов, у которых свойство float имеет значение left или right.

Свойство **width** задает ширину блочного элемента. По умолчанию это свойство имеет значение auto, и тогда ширина блока регулируется браузером автоматически. Но можно задавать ширину элемента и явно. Это можно делать в абсолютных единицах (пикселях, пунктах, миллиметрах) или в относительных единицах (процентах от общей ширины страницы). Последний способ является предпочтительным, поскольку он позволяет гибко регулировать ширину блока.

Свойство **height** определяет высоту блочного элемента. Все возможные значения этого свойства аналогичны значениям свойства width.

Свойства **max-width** и **max-height** задают наибольшую возможную ширину и высоту блочного элемента соответственно. Эти величины также можно задавать как в абсолютном, так и в относительном виде. Свойства **min-width** и **min-height** задают наименьшую возможную ширину и высоту блочного элемента соответственно.

Свойство **overflow** определяет, что будет в том случае, если текст или иное содержимое, относящееся к блочному элементу, полностью не помещается внутри данного элемента. Возможные значения свойства, следующие:

visible – высота блочного элемента автоматически увеличивается настолько, что вместит все его содержимое; это значение присваивается свойству по умолчанию;

hidden – не помещающееся в блочный элемент содержимое будет скрыто, а блок сохранит свои исходные размеры;

scroll – в блочном элементе появляются полосы прокрутки, позволяющие посмотреть все его содержимое; при этом исходные размеры блока также остаются неизменными.

Свойство **margin** нам уже знакомо по предыдущим главам данного пособия, но оно может применяться не только ко всему web-документу в целом, но и к отдельным блочным элементам. Для блочного элемента это свойство создает дополнительные внешние поля. Следует отметить, что поля не обязательно должны быть одинаковыми для всего элемента в целом. Свойства **margin-left**, **margin-right**, **margin-top** и **margin-bottom** позволяют определять размеры соответственно левого, правого, верхнего и нижнего полей блочного элемента. Размеры всех полей могут задаваться в виде абсолютных и относительных величин.

При работе с полями необходимо иметь в виду один существенный момент. Внешние поля не являются частью блочного элемента и поэтому при наличии полей фактическая ширина блочного элемента равна его ширине,

заданной свойством **width** плюс ширине полей элемента. Поэтому возможна такая ситуация: разработчик web-страницы создал несколько плавающих блоков, суммарная ширина которых составляет 100 % от ширины web-страницы. При отсутствии полей все блоки выстраиваются по горизонтали, как и было задумано разработчиком. Но при добавлении к блокам внешних полей итоговая ширина всех плавающих блоков оказывается больше, и часть блоков «съезжает» вниз. Следовательно, если планируется добавлять к плавающим блочным элементам внешние поля, то суммарная ширина блоков, определяемая их свойствами **width**, должна быть несколько меньше 100 %, чтобы оставить резерв по ширине для полей.

Свойство **padding**, знакомое нам по главе пособия, посвященной таблицам, в данной случае определяет величину внутренних полей блочных элементов. Так же, как и для внешнего поля, размеры левого, правого, верхнего и нижнего внутренних полей можно задавать отдельно атрибутами **padding-left**, **padding-right**, **padding-top** и **padding-bottom**.

Свойства, определяющие параметры границы, такие как **border-style**, **border-width** и **border-color** также могут применяться не только для таблиц, но и для других элементов web-страницы, к которым относятся, в частности, блоки и элементы списков.

Отметим, что существует еще одно полезное свойство из этой же категории – **border-radius**. Как известно, блоки web-документа (и их границы) по умолчанию имеют прямоугольную форму. Но можно создать границу в виде прямоугольника с закругленными краями. Для этой цели и используется свойство **border-radius**, значением которого является радиус закругления границы элемента, заданный в пикселях или миллиметрах.

Кроме того, следует отметить, что при создании правил для блочных элементов в каскадных таблицах стилей в них можно включать также свойства, описывающие параметры шрифта, который используется в данном блоке, фоновый цвет элемента, способ выравнивания текста в абзацах, и другие свойства, рассмотренные в предыдущих главах пособия.

Теперь рассмотрим применение блочных элементов и их свойств на практике. В качестве примера выберем создание страницы, которая посвящена популярному и востребованному сегодня языку программирования Java. Данная страница должна состоять из следующих четырех основных блоков:

а) заголовок страницы;

б) навигационная панель;

в) основной контент страницы;

г) сведения о разработчике страницы.

Для создания данной страницы напишем следующий HTML-код, включающий, соответственно, четыре тега <**DIV**>:

```
<HTML>
<HEAD>
<TITLE> Пример 5 </TITLE>
```

</HEAD>
<BODY>
<DIV >
<H1>
Сайт о языке программирования Java
</H1>
</DIV>
<DIV>
<UL>
<LI> <A HREF="Page5_1.html"> История языка </A>
<LI> <A HREF="Page5_2.html"> Основные операторы языка </A>
<LI> <A HREF="Page5_3.html"> Примеры программ на языке Java </A>
<LI> <A HREF="Page5_4.html"> О языке Javascript </A>
<LI> <A HREF="Page5_5.hmtl"> Полезные ссылки </A>
</UL>
</DIV>
<DIV CLASS="MAIN_CONT">
<P> Добро пожаловать на сайт, посвященный языку программирования Java. Сегодня Java является одним из наиболее популярных и востребованных языков программирования. Java изначально разрабатывался как средство для программирования бытовых электронных устройств, но впоследствии он вышел за рамки столь узкой специализации и применяется в самых разнообразных сферах человеческой деятельности.
</P>
<P>
Целью создания данного сайта является помощь всем желающим в овладении базовыми знаниями по языку Java. На сайте собрана разнообразная информация, относящаяся как непосредственно к данному языку, так и к созданному на его основе сценарному языку Javascript. При подготовке сайта автор использовал свой многолетний опыт преподавания программирования и информатики и стремился представить материалы сайта таким образом, чтобы они были максимально понятными и доступными для пользователей.
</P>
<P>
Автор надеется, что сайт окажется полезным как для тех пользователей, кто собирается в дальнейшем специализироваться в области программирования (и особенно web-программирования), так и для всех, интересующихся современными информационными технологиями
</P>
</DIV>
<DIV>

© А.Н. Маслобоев 2021
</DIV>
</BODY>
</HTML>

Внешний вид данной главной страницы сайта при просмотре его в браузере Firefox показан на рис. 17. Поскольку каскадная таблица стилей для данного сайта еще не создана и не подключена, то страница представляет собой просто текстовый документ без каких-либо элементов оформления.

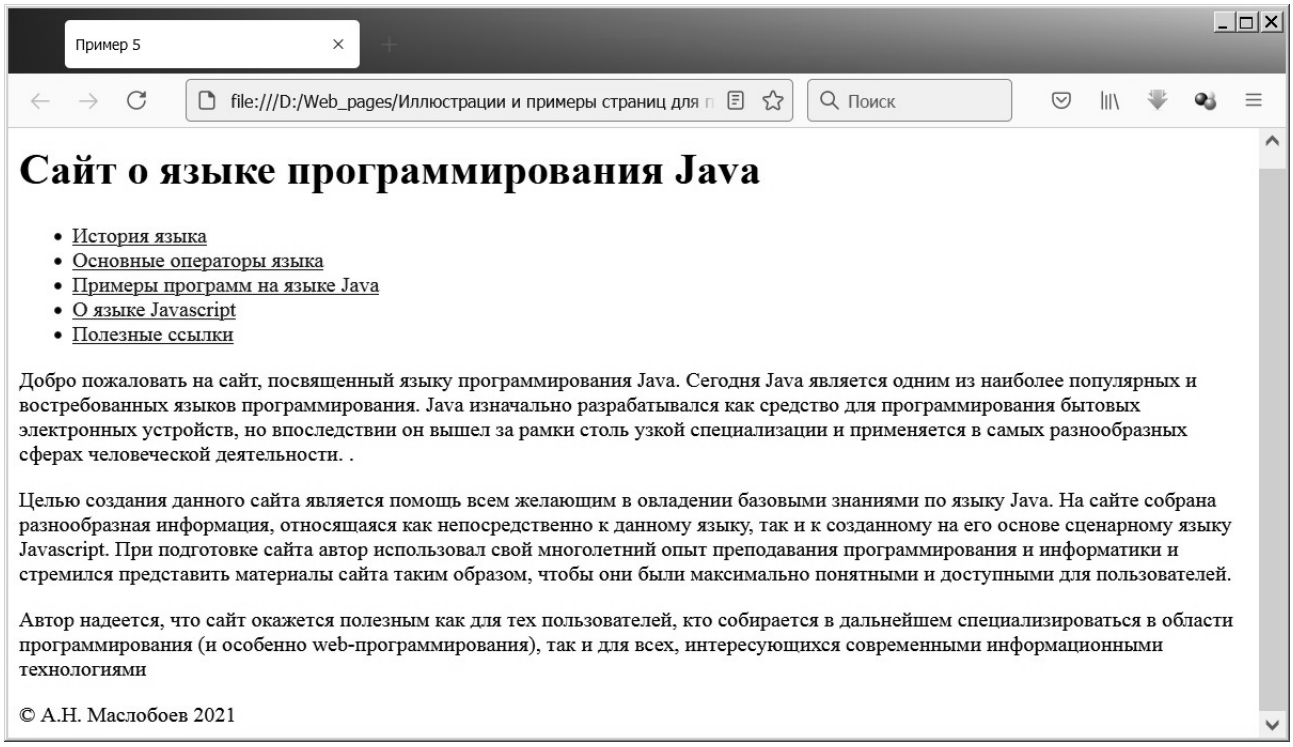

Рис.17. Главная страница сайта, посвященного языку программирования Java (до подключения к нему каскадной таблицы стилей)

Для преобразования данной страницы к виду, соответствующему современным требованиям web-дизайна, необходимо будет выполнить следующие действия. Во-первых, основной заголовок web-документа должен быть выполнен с помощью шрифта, который по гарнитуре и цвету (а не только по размеру) отличается от основного текста документа. Во-вторых, имеющиеся на странице ссылки лучше оформить в виде навигационной панели, которая должна располагаться под заголовком в левой части страницы. Тогда под заголовком справа будет находиться основное содержание страницы. В-третьих, сведения об авторе с авторским копирайтом целесообразно разместить внизу, но выровнять их по правому краю страницы. Все эти требования

необходимо будет учесть при создании каскадной таблицы стилей. Рассмотрим основные компоненты создаваемой таблицы стилей.

В начале таблицы задается правило для тега **BODY**. Определяется светло-зеленый фоновый цвет, синий цвет текста и общие поля для всего web-документа шириной в 20 пунктов. Затем описываются четыре класса, каждый из которых связан с одним из блоков страницы.

Класс **DIV.BEGIN** определяет вид первого блока. Ширина этого блока равна 100 % от ширины страницы, так что он занимает всю ее верхнюю полосу. По высоте эта полоса составляет 12 % от всей страницы. Для верхнего внешнего поля задается отрицательная величина, равная -10 пунктов. Это допускается правилами языка CSS и сделано для того, что заголовок, находящийся в этом блоке был подтянут выше к верхнему краю документа. Также задается выравнивание текста по центру, красный цвет шрифта, гарнитура Arial и его полужирное начертание. Используется еще одно свойство для обработки текста **Text-Transform**. Если этому свойству присвоить значение **Uppercase**, то текст будет автоматически преобразован к верхнему регистру. Все это сделано для выделения заголовка, который и составляет содержание блока.

Класс **DIV.NAVIGATION** используется для оформления второго блока, содержащего навигационную панель. Это блок описан как плавающий с выравниванием по левому краю. Ширина блока составляет 32 % от общей ширины страницы, а высота – 80 % от общей высоты. Для левого внешнего поля задано отрицательное значение, чтобы переместить блок ближе к левому краю страницы. Также задано внутреннее поле размером 8 пунктов, чтобы навигационные кнопки непосредственно не соприкасались с границами блока. Сама эта граница задана сплошной, цвета красного дерева и шириной в три пункта. Размер шрифта для внутриблочного текста задан равным 14 пунктам, гарнитуры Arial и полужирного начертания, текст будет преобразован к верхнему регистру. Подобные манипуляции с шрифтом нужны для того, чтобы надписи на навигационных кнопках (т. е. текст гиперссылок) также отличался от основного текста документа.

Класс **DIV.MAIN_CONT** определяет содержание третьего блока, содержащего основной контент страницы, представляющий собой базовые сведения о языке Java. Этот блок так же, как и второй, является плавающим с выравниванием по левому. Таким образом, второй и третий блок будут расположены по горизонтали под первым блоком. Ширина блока составляет 60 % от общей ширины страницы, а высота – 80 % от общей высоты. Для свойства **Overflow** задано значение **Scroll**, что обеспечивает прокручивание текста внутри блока. При ином значении этого свойства текст блока превратился бы в длинную «простыню», которая бы портила внешний вид страницы. А значение **Scroll** обеспечивает компактный вид блочного элемента и в то же время позволяет всем желающим ознакомиться с полным текстом. В блоке задано внутреннее поле в 8 пунктов, чтобы текст непосредственно не

соприкасался с границами блока. Границы данного блока заданы сплошными и цвета красного дерева.

Класс **DIV.AUTHOR** отвечает за оформление четвертого блока документа со сведениями об авторе. Для описания блока использовано свойство **CLEAR**, которое обеспечивает расположение этого блока под вторым и третьим. Ширина блока – 100 % от ширины страницы, т. е. он занимает всю ее нижнюю полосу, а высота – 8 % от высоты страницы. Для текста задано выравнивание по правому краю (именно так обычно располагаются сведения об авторе документа). Для шрифта задан красный цвет, размер 16 пунктов и полужирное начертание, чтобы он также отличался от основного текста документа.

Далее идут правила для отдельных тегов, используемых в web-документе. Правило для заголовка первого уровня задает для него размер шрифта в 24 пункта. Правило для абзаца создает для него красную строку размером в 20 пунктов, обеспечивает выравнивание текста абзаца по ширине, гарнитуру Calibri и кегль 14 пунктов. Именно таким шрифтом будет набран основной текст документа.

Правило для списка устанавливает для свойства **List-style-type** значение **none**, т. е. отсутствие маркеров в списке. Маркеры в данном случае не нужны, поскольку элементы списка (гиперссылки) будут представлять собой отдельные навигационные кнопки. Также значение левого внутреннего поля в списке устанавливается равным нулю, поскольку отступы здесь не требуются.

Правило для элементов списка трансформирует их внешний вид, превращая их в экранные навигационные кнопки. Для этого каждый элемент списка окружается сплошной толстой рамкой коричневого цвета с закругленными краями. Фоновый цвет для кнопок будет золотистым. Между текстом кнопок и их границами создаются поля шириной 5 пунктов.

Последнее правило в каскадной таблице стилей относится к гиперссылкам (они же надписи на навигационных кнопках). Цвет гиперссылок меняется на зеленый, а подчеркивание ссылок (в котором в данном случае нет необходимости) удаляется, для чего свойству **Text-decoration** присваивается значение **none**. Полный текст таблицы стилей приведен ниже.

```
BODY {BACKGROUND: #CCFFCC; COLOR:BLUE; MARGIN: 20pt}
DIV.BEGIN { HEIGHT: 12%; WIDTH: 100%;
MARGIN-TOP:-10pt;
COLOR: RED; FONT-FAMILY: ARIAL; TEXT-ALIGN: CENTER;
FONT- WEIGHT:BOLD; TEXT-TRANSFORM:UPPERCASE}
DIV.NAVIGATION { FLOAT:LEFT; WIDTH:32%; HEIGHT:80%;
MARGIN-LEFT:-10pt; MARGIN-BOTTOM: 10pt; PADDING: 8pt;
BORDER-STYLE:SOLID; BORDER-WIDTH:3pt; BORDER-
COLOR:MAROON;
FONT-SIZE:14pt; FONT-FAMILY:ARIAL; FONT-WEIGHT:BOLD;
TEXT-TRANSFORM:UPPERCASE}
```

```
DIV.MAIN_CONT {FLOAT:LEFT; HEIGHT:80%; WIDTH:60%; OVERFLOW:SCROLL;
MARGIN-LEFT:10pt; MARGIN-BOTTOM:10pt; PADDING:8pt;
BORDER-STYLE: SOLID; BORDER-COLOR:MAROON}
DIV.AUTHOR { CLEAR: BOTH; HEIGHT 8%; WIDTH:100%;
TEXT-ALIGN:RIGHT; COLOR: RED; FONT-FAMILY:CALIBRI; FONT-SIZE:16pt; FONT-WEIGHT:BOLD }
H1{FONT-SIZE:24pt}
P {TEXT-INDENT:20pt; FONT-FAMILY:CALIBRI; FONT-SIZE:14pt; TEXT-ALIGN:JUSTIFY}
UL {LIST-STYLE-TYPE:NONE; PADDING-LEFT:0pt}
LI {PADDING:5pt; MARGIN:5pt; BORDER-COLOR: BROWN;
BORDER-STYLE:SOLID; BORDER-WIDTH:THICK; BORDER-RADIUS:15px; BACKGROUND:GOLD }
A {COLOR:GREEN; TEXT-DECORATION:NONE}
```

Завершающим шагом в подготовке web-документа о языке Java является внесение необходимых изменений в HTML-код. Для этого таблицу стилей нужно подключить к странице с помощью тега **LINK**, в описание блочных элементов вставить ссылки на соответствующие классы из таблицы стилей. Тогда HTML-код web-страницы будет выглядеть таким образом:

```
<HTML>
<HEAD>
<TITLE> Пример 5 </TITLE>
<LINK HREF="STYLE5.CSS" REL="STYLESHEET" TYPE="TEXT/CSS">
</HEAD>
<BODY>
<DIV CLASS=BEGIN>
<H1>
Сайт о языке программирования Java
</H1>
</DIV>
<DIV CLASS=NAVIGATION>
<UL>
<LI > <A HREF="Page5_1.html"> История языка </A>
<LI > <A HREF="Page5_2.html"> Основные операторы языка </A>
<LI > <A HREF="Page5_3.html"> Примеры программ на Java </A>
<LI > <A HREF="Page5_4.html"> О языке Javascript </A>
<LI > <A HREF="Page5_5.hmtl"> Полезные ссылки </A>
</UL>
</DIV>
<DIV CLASS=MAIN_CONT>
```

<P> Добро пожаловать на сайт, посвященный языку программирования Java. Сегодня Java является одним из наиболее популярных и востребованных языков программирования. Java изначально разрабатывался как средство для программирования бытовых электронных устройств, но впоследствии он вышел за рамки столь узкой специализации и применяется в самых разнообразных сферах человеческой деятельности.

</P>

<P>

Целью создания данного сайта является помощь всем желающим в овладении базовыми знаниями по языку Java. На сайте собрана разнообразная информация, относящаяся как непосредственно к данному языку, так и к созданному на его основе сценарному языку Javascript. При подготовке сайта автор использовал свой многолетний опыт преподавания программирования и информатики и стремился представить материалы сайта таким образом, чтобы они были максимально понятными и доступными для пользователей.

</P>

<P>

Автор надеется, что сайт окажется полезным как для тех пользователей, кто собирается в дальнейшем специализироваться в области программирования (и особенно web-программирования), так и для всех, интересующихся современными информационными технологиями

</P>

</DIV>

<DIV CLASS=AUTHOR>

© А.Н. Маслобоев 2021

</DIV>

</BODY>

</HTML>

На рис. 18 показана web-страница после подключения каскадной таблицы стилей при просмотре ее в браузере Firefox. Достаточно сравнить полученный результат с исходным, представленным на рис. 17, чтобы убедиться в нужности и полезности блочной технологии создания и оформления web-страниц. На порядок улучшить внешний вид страницы позволили достаточно простые, но эффективные средства, которые, безусловно, должны занять достойное место в арсенале web-разработчика.

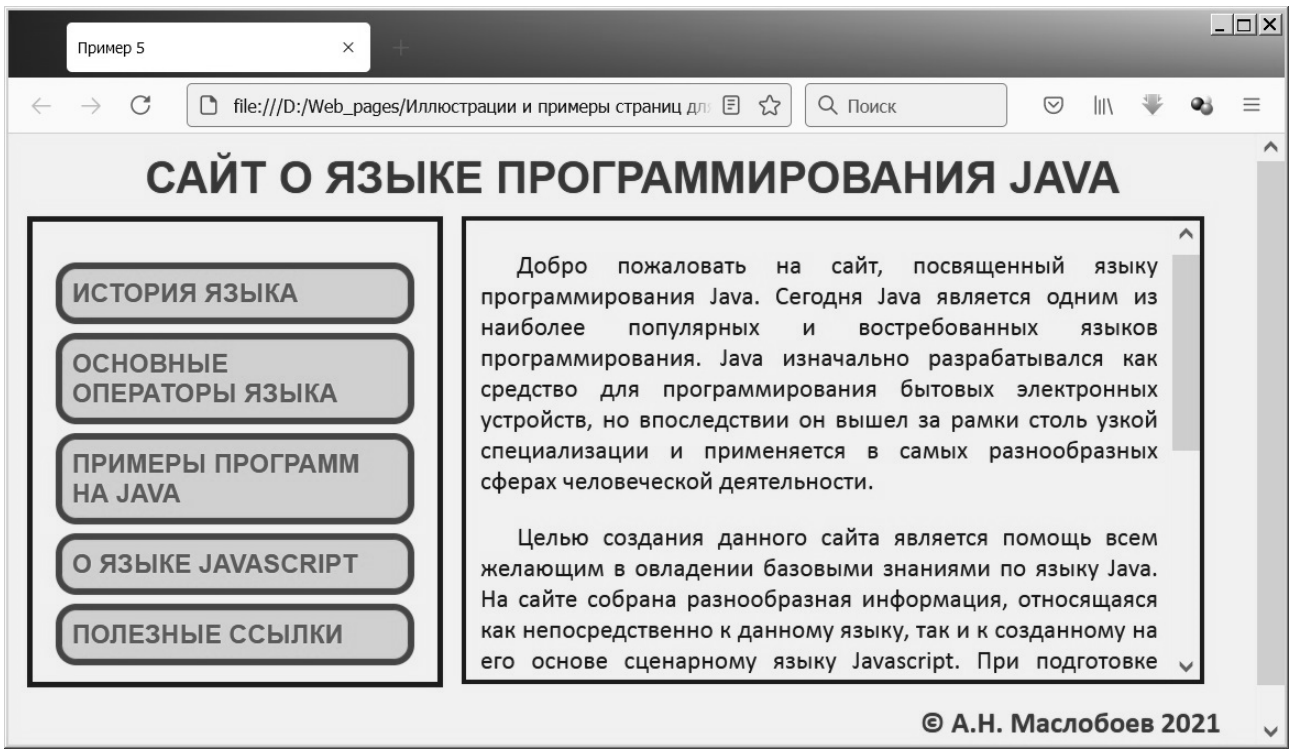

Рис. 18. Главная страница сайта о языке Java
(после подключения каскадной таблицы стилей)

Контрольные вопросы

1. Какой парный тег языка HTML используется для создания нового блочного элемента в web-документе?
2. Какое свойство языка CSS позволяет преобразовывать любой текст в верхний регистр?
3. Какое свойство языка CSS задает различные способы выравнивания плавающих блочных элементов на web-странице?
4. Какое свойство языка CSS позволяет расположить текущий блочный элемент ниже предшествующих ему элементов?
5. Какое свойство языка CSS позволяет указывать ширину блочного элемента в процентах от общей ширины экрана?
6. Какие свойства языка CSS позволяют задавать максимальную и минимальную ширину блочного элемента?
7. Какие свойства языка CSS позволяют задавать максимальную и минимальную высоту блочного элемента?
8. Какое свойство языка CSS определяет поведение блочного элемента в случае его переполнения?
9. Какие свойства языка CSS позволяют задавать отдельно левое, правое, верхнее и нижнее поле для блочного элемента?
10. С помощью какого свойства языка CSS можно создать закругленные границы элемента?

ЗАКЛЮЧЕНИЕ

Задачей данного учебного пособия является ознакомление студентов только с базовыми принципами использования каскадных таблиц стилей. В связи с этим за рамками пособия остались многие возможности применения каскадных таблиц стилей.

К ним относится использование браузерных префиксов, применение CSS-переходов (включая использование временных функций), регулировка прозрачности элементов с помощью цветовой модели RGBA, различные виды CSS-трансформаций, работа с множественными фонами, CSS-анимации и целый ряд других вопросов.

Но полученные при изучении этого пособия знания могут стать необходимой основой для дальнейшего расширения и углубления обучающимися знаний и профессиональных умений, которые потребуются будущим специалистам в области информационных технологий.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дронов, В. А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов [Текст] / В.А. Дронов. — СПб.: БХВ-Петербург, 2011. – 416 с. – ISBN 978-5-9775-0596-3.
2. Ломов, А. Ю. HTML, CSS, скрипты: практика создания сайтов [Текст] / А. Ю. Ломов. – СПб.: БХВ-Петербург, 2006. – 416 с. – ISBN 5-94157-698-6.
3. Маслобоев, А. Н. Базовые возможности языка HTML по разработке и оформлению Web-страниц: учеб.-метод. пособие [Текст] / ВШТЭ СПбГУПТД. – СПб. 2016. – 87 с.
4. Маслобоев, А.Н. Дополнительные возможности языка HTML по разработке и оформлению Web-страниц: учебно-методическое пособие[Текст] / ВШТЭ СПбГУПТД. – СПб., 2018. – 48с.
5. Робсон, Э. Изучаем HTML, XHTML и CSS. 2-е изд. [Текст] / Э.Робсон, Э. Фримен – СПб.: Питер, 2014. – 720 с. –ISBN 978-5-496-00653-8.
6. Сидерхолм, Д. CSS3 для веб-дизайнеров [Текст] / Д. Сидерхолм; пер. с англ. Е. Кудашева. – М.: Манн, Иванов и Фербер, 2013 – 144 с. – ISBN 978-5-91657-595-8.

Учебное издание

Артур Николаевич Маслобоев

Web-страницы
Использование каскадных таблиц стилей для оформления web-страниц

:

Редактор и корректор А. А. Чернышева
Техн. редактор Д. А. Романова Темплан 2021 г., поз. 43

Подписано к печати 30.11.21. Формат 60x84/16. Бумага тип № 1.
Печать офсетная. Печ.л. 4,1. Уч.-изд. л. 4,1.
Тираж 50 экз. Изд. № 43. Цена «С». Заказ №

Ризограф Высшей школы технологии и энергетики СПбГУПТД,
198095, Санкт-Петербург, ул. Ивана Черных, 4.