***Вадим Дунаев**. Из книги “HTML, скрипты и стили”, 4-е изд.*

Глава 3

Основы CSS

Каскадные таблицы (листы) стилей (CSS) уже неоднократно упоминались в предыдущих главах. Теперь мы рассмотрим более подробно, как они вставляются в (X)HTML-код и как создаются правила форматирования документа.

3.1. Присоединение таблиц стилей к (X)HTML-документу

CSS содержат правила форматирования (стилевые параметры или свойства элементов), которые можно присоединить к (X)HTML-документу несколькими способами.

- Внедрение — запись правил непосредственно в (X)HTML-коде внутри контейнера `<style>`, например

```
<style type="text/css">
  p {color:blue; font-size:12pt}
</style>
```

  Внедренная указанным образом таблица стилей будет действовать только на тот документ, в котором находится. Контейнер `<style>` рекомендуется размещать внутри `<head>`, хотя возможно и иное его расположение.

- Связывание — запись правил в отдельном текстовом файле (обычно с расширением css) и помещение в (X)HTML-коде ссылки на него. Это можно сделать с помощью тега `<link>`, который обычно размещают в контейнере `<head>`:

```
<head>
<link rel="stylesheet" type="text/css"
      href="URL-адрес файла с правилами CSS" />
</head>
```

  или с помощью директивы `@import` в контейнере `<style>`:

```
<style type="text/css">
  @import url("URL-адрес файла с правилами CSS");
</style>
```

Внимание!

Обратите внимание на атрибуты тегов `<link>` и `<style>`, а также на точку с запятой в конце записи директивы `@import`. Теги `<link>` и `<style>` имеют еще необязательный атрибут `media` для указания типа устройства вывода, принимающий значения `"all"` (все устройства), `"screen"` (экран монитора, принимается по умолчанию), `"print"` (печатающие устройства), `"projection"` (проектор), `"braille"` (устройства, основанные на системе Брайля для слепых), `"speech"` (речевые синтезаторы и программы звукового воспроизведения текста). В конце директивы `@import` можно указать через запятую и без кавычек список типов устройств:

all, screen, print, projection, braille, aural (речевые синтезаторы и программы звукового воспроизведения текста), handheld (карманные компьютеры и т. п.), tv (телевизор). Вместо данных способов указания устройств вывода можно применить директиву @media (*см. разд. 3.2*).

Директиву @import можно использовать также и в таблице стилей, сохраненной во внешнем файле.

Таблица стилей из внешнего файла действует на те документы, в которых содержится ссылка на нее. Разумеется, она передается браузеру вместе с документом, который на нее ссылается. Однако браузеры могут кэшировать загруженные данные и при повторном запросе таблицы стилей, последняя не будет передаваться с сервера снова, если она еще находится в кэше. Хранение таблицы стилей в отдельном файле выгодно, если она применяется к нескольким документам одного сайта. В этом случае вы корректируете стилевые параметры в одном файле, а их действие распространяется сразу на несколько (X)HTML-документов.

- Встраивание — указание стиля непосредственно в теге с помощью атрибута style: <тег style="*список параметров*">, например

  ```
  <h1 style="color:red; font-size:36pt">Большой красный заголовок</h1>
  ```

  Данный способ удобен, когда хотят назначить индивидуальные стилевые параметры для некоторых элементов только в одном документе, не корректируя основную таблицу стилей в контейнере <style> или во внешнем файле. Правила форматирования, заданные таким образом, обладают наивысшим приоритетом (*подробнее см. разд. 3.3*).

К одному и тому же документу можно присоединить несколько таблиц стилей, причем различными способами. Их взаимодействие будет определяться приоритетами, которые мы рассмотрим в *разд. 3.3*.

3.2. Правила форматирования

Таблица стилей, внедренная или связанная (расположенная во внешнем файле), содержит одно или несколько правил форматирования, каждое из которых записывают в соответствии с синтаксисом:

список_селекторов {*имя_параметра*:*значение*; *имя_параметра*:*значение*;… }

Здесь *список_селекторов* — один или несколько идентификаторов (селекторов), разделенных запятыми. Селекторы предназначены для указания элементов документа, к которым данное правило применяется. Стилевые параметры (свойства) заключают в фигурные скобки и разделяют точкой с запятой; имя параметра и его значение разделяют двоеточием.

Внимание!

В настоящее время официальной спецификации CSS не существует, поэтому некоторые параметры из CSS 3 могут интерпретироваться различными браузерами по-разному. Кроме того, для некоторых параметров перед их именем следует указывать так называемый вендорный префикс, для каждого браузера свой. Так, для Mozilla Firefox (движок Gecko) добавляется префикс -moz-, для Google Chrome и Apple Safari (движок WebKit) — -webkit-, для Opera — -o-, для Microsoft Internet Explorer (движок Trident) — -ms-. Например, параметр transform, задающий трансформацию элемента, для Opera записывается как -o-transform, а для Chrome — как -webkit-transform. Для обеспечения межбраузерной инвариантности используемый параметр сначала записывают без каких бы то ни было префиксов, а затем дублируют его с префиксами для тех браузеров, для которых это необходимо.

3.2.1. Селекторы

Селекторы нужны для именования групп стилевых параметров, чтобы их можно было назначить одному или нескольким элементам документа. Если селектор задает просто имя для группы параметров, то элементы документа могут ссылаться на них посредством имени. С другой стороны, селектор может еще и указывать, к каким элементам документа относятся соответствующие ему параметры. Сначала рассмотрим простейшие (элементарные) селекторы, из которых затем образуются более сложные (составные).

Элементарные селекторы:

- ***** — параметры с данным селектором (звездочка) применяются ко всем элементам, например

  `* {font-size:20px};`

  Данное правило предписывает отображать шрифт всех элементов документа размером 20 px;
- ***тег*** — имя HTML-тега без уголковых скобок, например, `div`; параметры применяются ко всем элементам, заданным тегом <*тег*>; например

  `h1, h2 {font-family: Arial; font-size:14pt}`

  Здесь всем заголовкам первого и второго уровней (т. е. элементам, заданным посредством тегов `<h1>` и `<h2>`) назначается шрифт Arial размером 14 pt;
- ***#идентификатор*** — произвольное слово, подобное имени переменной, перед которым ставится `#`, например `#myelement`; для применения параметров с данным селектором к элементу последний должен иметь атрибут `id="`*идентификатор*`"` (листинг 3.1). Поскольку значение атрибута `id` должно быть уникальным, то стилевые параметры применяются к единственному элементу.

Листинг 3.1. Пример использования значения атрибута `id` как селектора

```
<style type="text/css">
     #mytext {color:red}
</style>
<p id="mytext">Совершенно секретно</p>
<p>Подробности правил форматирования можно найти в справочнике по CSS</p>
```

Здесь красный цвет текста назначается только первому абзацу, заданному тегом `<p id="mytext">`, а второй абзац будет отформатирован согласно правилам по умолчанию;

- ***.класс*** — произвольное слово, подобное имени переменной, перед которым ставится точка, например `.mystyle2`; для применения параметров CSS с данным селектором к элементам последние должны иметь атрибут `class="`*класс*`"`; класс применяется к тем элементам, у которых значением атрибута `class` является имя этого класса (листинг 3.2).

Листинг 3.2. Пример использования атрибута `class`

```
<style type="text/css">
    .italic {font-style:italic}
    .underline {text-decoration:underline}
</style>
<h1>Обычный заголовок</h1>
```

```
<h1 class="italic">Заголовок курсивом</h1>
<h1 class="underline">Заголовок подчеркнутый</h1>
```

В листинге 3.2 определены два класса (`italic` и `underline`), применимые только к элементам, у которых атрибут `class` имеет соответствующее значение (`"italic"` или `"underline"`). Три заголовка первого уровня в данном (X)HTML-коде будут иметь различный вид. Первое вхождение в документ элемента `<h1>` будет отображено в окне браузера в соответствии с параметрами, установленными по умолчанию, поскольку оно не ссылается на CSS. Представления остальных двух вхождений `<h1>` будут подкорректированы в соответствии с наборами параметров, на которые они ссылаются посредством атрибута `class`: содержимое второго вхождения `<h1>` будет представлено курсивом (`font-style:italic`), а третьего — подчеркнутым шрифтом (`text-decoration:underline`).

Из элементарных образуются составные селекторы, позволяющие уточнить область действия соответствующих стилевых параметров:

- ***селектор.класс*** — элементы, соответствующие *селектор*, у которых атрибут `class="`*класс*`"`. Например, область действия селектора `div.small` — все элементы вида `<div class="small">`;
- ***селектор#идентификатор*** — элементы, соответствующие *селектор*, у которых атрибут `id="`*идентификатор*`"`. Например, область действия селектора `div#mybox` — элемент вида `<div id="mybox">`.

В листинге 3.3 класс `h1.mystyle` определен только для элементов вида `<h1 class="mystyle">`, а класс `.mystyle` — для всех элементов, имеющих атрибут `class="mystyle"`. Поэтому заголовок `<h1>` будет подчеркнутым и красным, первый абзац `<p>` — красным, а второй абзац в соответствии с параметрами по умолчанию будет обычным (черным).

Листинг 3.3. Пример составного селектора

```
<style type="text/css">
     h1.mystyle {text-decoration:underline;}
     .mystyle {color:red}
</style>
<h1 class="mystyle">Заголовок подчеркнутый</h1>
<p class="mystyle">Красный текст</p>
<p>Обычный</p>
```

Кроме перечисленных, составными селекторами являются псевдоселекторы и псевдоэлементы (*см. разд. 3.2.3*).

3.2.2. Контекстные селекторы

Иногда требуется установить стиль для элементов, которые находятся в определенном контексте с другими элементами. В данном случае селектор образуется из элементарных и составных селекторов, между которыми устанавливается пробел, > или + в зависимости от вида учитываемого контекста. Есть и другие правила образования контекстных селекторов. В табл. 3.1 указаны основные виды контекстных селекторов и область действия правил с их участием.

Вот несколько примеров контекстных селекторов:

```
div #mytext { параметры }
div .myclass { параметры }
.myclass1 .myclass2 { параметры }
#myelem >.myclass { параметры }
div * { параметры }
```

Таблица 3.1. *Контекстные селекторы*

Селектор	**Область действия**
селектор1 селектор2	Элементы, соответствующие *селектор2*, которые находятся внутри элементов, соответствующих *селектор1*. Например, область действия селектора div span — все элементы внутри элементов
селектор1>селектор2	Элементы, соответствующие *селектор2*, для которых родительский элемент соответствует *селектор1*. Например, область действия селектора div>span — все элементы , для которых родительским является
селектор1+селектор2	Элементы, соответствующие *селектор2*, которым непосредственно предшествует элемент, соответствующий *селектор1*. Например, область действия селектора div+span — все элементы , непосредственно перед которыми находится
селектор[*attr*]	Элементы, соответствующие *селектор*, у которых установлен атрибут *attr* (независимо от его значения). Например, область действия селектора img[alt] — все элементы графических изображений с установленным атрибутом alt
селектор[*attr*='*значение*']	Элементы, соответствующие *селектор*, у которых атрибут *attr* имеет значение *значение*. Например, область действия селектора img[alt='Мой портрет'] — все элементы графических изображений с атрибутом alt ='Мой портрет'
селектор[*attr*~='*значение*']	Элементы, соответствующие *селектор*, у которых атрибут *attr* имеет в качестве значения список слов, разделенных пробелами, и одно из этих слов есть *значение*
селектор[*attr*|='*значение*']	Элементы, соответствующие *селектор*, у которых атрибут *attr* имеет в качестве значения список слов, разделенных дефисами, и первое слово есть *значение*

Сложный селектор может содержать более двух простых, их примеры будут рассмотрены в *главе 8*.

3.2.3. Псевдоселекторы и псевдоэлементы

В табл. 3.2 приведены псевдоселекторы и псевдоэлементы, которые применяются как составные части селекторов: перед псевдоклассами и псевдоэлементами указывается элементарный или составной селектор из числа рассмотренных ранее.

Таблица 3.2. *Псевдоселекторы и псевдоэлементы*

Наименование	**Обозначение**
`:first-child`	Первый дочерний элемент
`:first-line`	Первая строка элемента
`:first-letter`	Первая буква в элементе
`:hover`	Элемент, на котором находится указатель мыши
`:active`	Активный элемент
`:focus`	Элемент, на котором установлен фокус
`:link`	Неиспользованные ссылки
`:visited`	Использованные ссылки
`:lang(язык)`	Элемент с указанным языком `язык`
`:before`	Предшествующий элемент
`:after`	Следующий элемент

В следующем примере псевдоклассы служат для задания ссылкам с различным статусом различных оттенков красного цвета:

```
a:link {color:#aa0000}
a:visited {color:#800000}
a:hover {color:#ff0000}
```

На рис. 3.1 в качестве примера показан документ, содержащий три абзаца (`<p>`), которые отформатированы по-разному, поскольку они ссылаются на различные правила CSS. Другие примеры приведены в *разд. 7.6, 8.3*.

В коде CSS можно (и полезно) добавлять комментарии, которыми считается все, что заключено между символами `/*` и `*/`.

Важно иметь в виду, что браузеры игнорируют правила CSS с ошибками. Наиболее типичные ошибки — отсутствие размерности для числовых значений и пропуск точки с запятой в качестве разделителя между определениями параметров.

Для различных элементов документа можно определить разные правила форматирования в зависимости от устройства вывода (типа носителя). Например, представление документа на листе бумаги нередко требуется сделать иным, чем на экране компьютера. С этой целью можно применить директиву

```
@media список_типов_носителей {
      правила_форматирования
}
```

Здесь после ключевого слова `@media` указывается без кавычек один или несколько типов носителей (через запятую): `all` (все типы), `screen` (экран монитора, принимается по

умолчанию), `print` (печатающее устройство), `projection` (проектор), `braille` (устройство, основанное на системе Брайля для слепых), `aural` (речевые синтезаторы и программы звукового воспроизведения текста), `handheld` (карманные компьютеры и т. п.), `tv` (телевизор). Далее в фигурных скобках следуют правила форматирования.

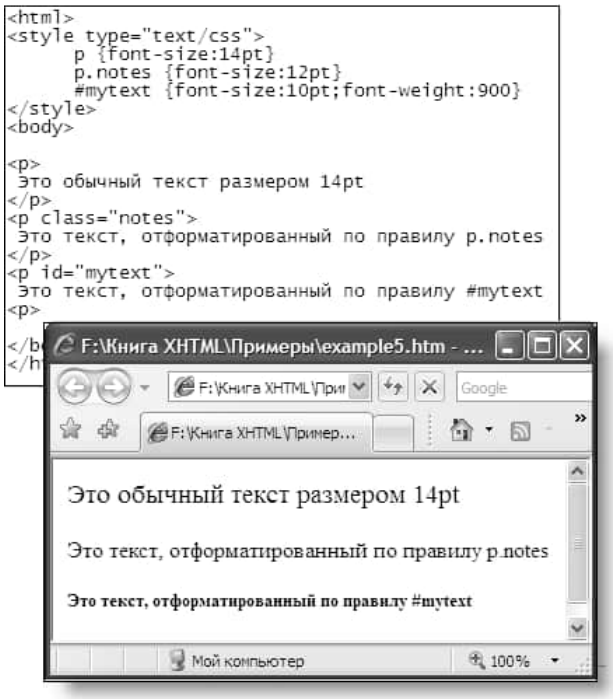

Рис. 3.1. Примеры ссылок на правила CSS

Пример

```
@media screen {
  body {font:12px Arial; color:blue}
}
@media print, handheld {
  body {font: 8pt Times; color:black}
  img {display:none}
}
```

3.3. Приоритеты определений параметров стилей

Не исключено, что в таблице стилей для одного элемента (X)HTML-документа может оказаться несколько различных определений одного и того же параметра. Вероятность такого события возрастает, если к одному документу применено одновременно несколько различных таблиц. Возникающие при этом противоречия браузер разрешает по некоторым правилам, учитывая приоритеты (веса) определений параметров во всех правилах CSS, относящихся к данному элементу документа.

Может статься, что для какого-то элемента автор явно не задал в CSS правило для одного или нескольких параметров. В этом случае элемент все равно приобретет некоторые значения для таких параметров. Кроме таблиц стилей, создаваемых автором документа, браузер учитывает CSS пользователя, заданные в настройках браузера, а также значения параметров, установленные по умолчанию. По умолчанию приоритет таблиц стилей пользователя меньше, чем автора документа. Если между ними возникает конфликт, то применяются таблицы автора. Самым низким приоритетом обладают стилевые параметры, принимаемые браузером по умолчанию. Так что ни один элемент вашего документа не останется без параметров форматирования. Вместе с тем если вы не контролируете форматирование, то результат может оказаться неожиданным.

Итак, если для одного и того же элемента подходят одновременно несколько правил форматирования из одной или нескольких различных таблиц, то они применяются каскадно по следующему алгоритму:

1. Наивысшим приоритетом обладает определение параметра, заданное атрибутом `style` тега рассматриваемого элемента.

   Пример

```
<style type="text/css">
p {font-size:16pt}
</style>
<p style="font-size:20pt">Привет!</p>
```

   Здесь для абзаца (содержимого элемента `<p>`) задано два определения размера шрифта (`font-size`), но второе из них имеет больший приоритет, а потому текст абзаца будет отображен шрифтом размера 20 pt.

   Если определения, заданного атрибутом `style`, нет, то рассматриваются все определения в правилах с селекторами, соответствующие рассматриваемому элементу (см. п. 2). Если таковых нет, то используется значение, наследуемое от родительского элемента. Если оно не определено (не все параметры наследуются), то берется значение по умолчанию. О наследовании *см. разд. 3.6*.

2. Если элементу соответствует несколько определений параметра, то они упорядочиваются по специфичности селектора, вычисляемой следующим образом. В селекторе подсчитываются три числа:

   - k1 — количество значений атрибута `id`;
   - k2 — количество классов;
   - k3 — количество имен тегов.

   Результат записывают как k1k2k3, и полученное таким образом число представляет уровень специфичности (вес) селектора. Выбираются определения параметров с наибольшей специфичностью. Примеры приведены в табл. 3.3.

3. Если элементу все еще соответствует несколько определений параметра, то они упорядочиваются по своему происхождению (от низшего к высшему): принятые браузером по умолчанию, пользовательские, авторские.

4. Если элементу все еще соответствует несколько определений параметра, то они сортируются по порядку их упоминания в таблице стилей. При этом определения в импортированной таблице (вызываемой директивой `@import`) считаются предшествующими определениям таблицы, в которую они были импортированы.

Наибольший приоритет имеет последнее определение.

Таблица 3.3. *Примеры специфичности селекторов*

Селектор	**Значения k1, k2, k3**	**Специфичность**
`h1`	k1 = 0, k2 = 0, k3 = 1	1
`div p`	k1 = 0, k2 = 0, k3 = 2	2
`.myclass`	k1 = 0, k2 = 1, k3 = 0	10
`h1.myclass`	k1 = 0, k2 = 1, k3 = 1	11
`#myelement`	k1 = 1, k2 = 0, k3 = 0	100
`#myelement.myclass`	k1 = 1, k2 = 1, k3 = 0	110

Чаще всего употребляют определения стилевых параметров, заданные не атрибутом `style`, а правилами с селекторами, причем простыми, такими как имя тега, класс и значение атрибута `id`. Приоритеты (специфичность или вес) определений параметров в правилах с такими селекторами нетрудно запомнить (табл. 3.4).

Таблица 3.4. *Приоритеты простых селекторов*

Вид селектора	**Вес**
имя_тега	1
имя_класса	10
имя_тега.имя_класса	11
#значение_id	100
#значение_id.имя_класса	110

На рис. 3.2 показано применение четырех правил к одному элементу, заданному тегом `<p>`. Результат определяется приоритетами правил CSS. Правило `p.notes` сильнее, чем `p`, но слабее правила `#mytext`. Самое сильное правило задано значением атрибута `style`. Следовательно, размер шрифта элемента `<p>` определится именно этим правилом. Кроме того, шрифт приобретет вес (жирность), указанный параметром `font-weight` в третьем правиле, поскольку он не задан в других правилах.

Приоритетом правил можно управлять с помощью директивы `!important` (англ. *important* — важный, значительный). Эту директиву записывают сразу после стилевого параметра, приоритет которого следует увеличить, например,

```
h2 {font-size:24pt !important; color:red}
```

Параметр таблицы стилей пользователя, помеченный как `!important`, имеет более высокий приоритет, чем тот же параметр автора, заданный без этой директивы. Авторский параметр с пометкой `!important` обладает более высоким приоритетом, чем пользовательский, даже если он имеет такую же пометку.

```
<html>
<style type="text/css">
      p {font-size:14pt}
      p.notes {font-size:12pt}
      #mytext {font-size:10pt;font-weight:900}
</style>
<body>

<p id="mytext" class="notes" style="font-size:18pt">
 Это некоторый текст
</p>

</bo
</ht
```

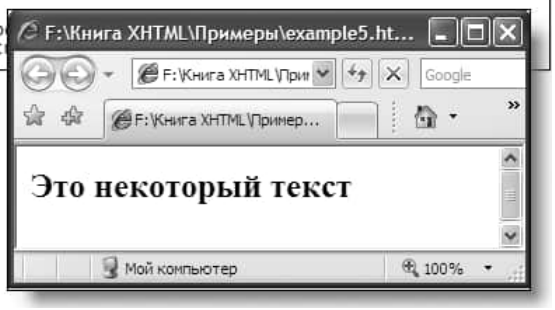

Рис. 3.2. Пример применения нескольких правил CSS к одному элементу

3.4. Размерность и цвета

Многие стилевые параметры принимают в качестве значений числа. Так задают размеры шрифта, видимых элементов, отступов от границ экрана или объемлющих контейнеров. При этом для всех чисел, кроме нуля, необходимо указывать размерность, т. е. в каких единицах измеряется та или другая числовая величина.

В CSS имеются следующие линейные единицы измерения:

- абсолютные — `cm` (сантиметры), `mm` (миллиметры), `in` (дюймы), `pt` (пункты) и `pc` (пики);
- относительные — `em`, `ex` и `%`;
- пикселы — `px`.

Величины абсолютных единиц не зависят от устройств отображения (дисплеев, принтеров и т. п.). Например, отрезок длиной 1 cm будет иметь эту же длину и на дисплее, и на листе бумаги. Абсолютные единицы связаны между собой следующими равенствами: 1 in = 2.54 cm, 1 pt = 1/72 in, 1 pc = 12 pt.

Все иначе для относительных единиц. 1 em равен размеру так называемого родительского шрифта, а 1 ex — высоте строчной буквы "x". Например, в следующем правиле устанавливается отступ заголовка (`<h1>`) от краев контейнера в четыре раза больший, чем размер его шрифта (значение `font-size`, установленное браузером по умолчанию или в других правилах CSS):

```
h1 {margin:4em}
```

Проценты (`%`) показывают отношение к величине некоторого другого элемента. Для каждого свойства, допускающего задание значений в процентах, подразумевается еще и величина, относительно которой определяется процентное соотношение. Например, такой величиной может выступать значение ширины или высоты контейнера, содержащего данный элемент, или величина текущего шрифта.

Пикселы (`px`) часто относят к абсолютным единицам измерения, хотя они имеют относительный характер. Именно поэтому здесь они выделены в отдельную категорию. Дело в том, что видимый размер отрезка прямой, заданный в пикселах, зависит от устройства

отображения (в отличие от абсолютных единиц), но не зависит от размеров других элементов (в отличие от относительных единиц).

Устройства отображают графическую информацию дискретно в виде набора цветных точек, размеры и общее количество которых ограничены. Эти точки (dots) нередко называют пикселами (pixels), хотя последние — элементы графического растрового изображения, а не физической структуры дискретного отображения. Элемент изображения (pixel) не имеет абсолютных размеров, а элемент системы отображения (dot) — имеет, что и ограничивает ее разрешающую способность. Если, например, монитор настроен на отображение по горизонтали 1280 точек (экранных пикселов), то горизонтальный отрезок с установленной длиной `256px` займет на экране 20% ширины. А если монитор настроен на отображение 256 точек, то тот же отрезок займет всю ширину экрана. Поскольку существуют мониторы с одинаковым количеством точек, но имеющие различные абсолютные линейные размеры и наоборот, то становится очевидным — длина отрезка прямой в пикселах является относительной величиной. Тем не менее, в данной книге, говоря об абсолютных единицах измерения, мы будем иметь в виду и пикселы.

Угловые величины представляются в следующих единицах измерения:

- градусы — `deg`;
- радианы — `rad` (1rad ≈ 57,3deg)

Цвет задают в системе RGB последовательностью из трех чисел, определяющих яркость красной (Red), зеленой (Green) и синей (Blue) составляющих, записываемой в шестнадцатеричном или десятичном форматах.

В шестнадцатеричном формате каждая составляющая цвета представляется двухразрядным шестнадцатеричным числом от `00` до `ff` (всего 256 различных значений), указывающим ее яркость. Самый темный цвет — `00`, а самый яркий — `ff`. Например, черному соответствует шестнадцатеричное число `000000`, а белому — `ffffff`. Вот пример задания чистого красного цвета для текста в CSS:

```
color:#ff0000.
```

Если в каждой паре шестнадцатеричных чисел цифры одинаковые, то допускается сокращенная трехразрядная запись. Например, вместо `#ffbb88` можно написать `#fb8`.

Чтобы задать цвет десятичными числами, применяется функция `rgb(`*r*`, `*g*`, `*b*`)`, где *r*, *g*, *b* — целые десятичные числа от 0 до 255, соответствующие яркости красной, зеленой и синей составляющих смеси цветов. Возможно также указание значений яркости в процентах. Например, красный цвет для текста в CSS задается так:

```
color: rgb(255,0,0)или color: rgb(100%,0,0).
```

Напомню, что различные оттенки серого цвета (от черного до белого) задаются равными значениями для трех составляющих, например `color:#808080` или `color: rgb(100,100,100)`.

Кроме числовых значений цвета допустимо указывать их имена, являющиеся ключевыми словами, например `color: red`. В табл. 3.5 приведены имена и шестнадцатеричные значения для некоторых цветов, наиболее популярных в Web.

Все графические изображения, создаваемые для Web, должны использовать либо цветовую модель RGB, либо подмножество индексированных цветов, либо представляться в градациях серого цвета. Это общее правило. Указание имен из табл. 3.5 гарантирует принадлежность ваших

цветов Web-палитре, содержащей 216 цветов, которые отображаются одинаково всеми цветными мониторами.

Таблица 3.5. *Имена и числовые значения основных цветов*

Имя в CSS	**Числовое значение**	**Название**
aqua	00ffff	Аква (бирюзовый)
aquamarine	7fffd4	Аквамарин
beige	f5f5dc	Бежевый
black	000000	Черный
blue	0000ff	Голубой
brown	a52a2a	Коричневый
cyan	00ffff	Циан (бирюзовый)
gold	ffd700	Золотой
gray	808080	Серый
green	008000	Зеленый
magenta	ff00ff	Светло-пурпурный
navy	000080	Синий
olive	808000	Оливковый
pink	ffc0cb	Розовый
purple	800080	Пурпурный
red	ff0000	Красный
sienna	a0522d	Охра
silver	c0c0c0	Серебристый
violet	ee82ee	Фиолетовый
white	ffffff	Белый
yellow	ffff00	Желтый

3.5. Блоки: поля, отступы, границы и размеры

Элементы (X)HTML-документа с точки зрения CSS представляются в виде прямоугольных блоков, каждый из которых имеет информативную область (content — содержимое), отступы (padding), границы (border) и поля (margin) как показано на рис. 3.3.

Периметр каждой из четырех областей называется ее краем или краевой линией. Край имеет четыре сегмента: верхний (top), правый (right), нижний (bottom) и левый (left).

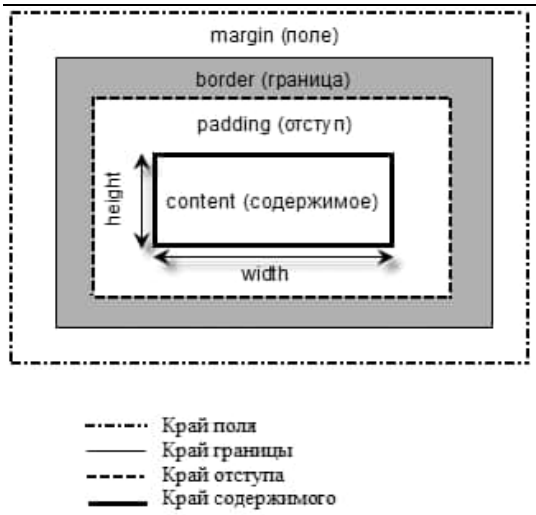

Рис. 3.3. Блоковая модель элемента документа

Фон различных областей блока определяется следующим образом:

- **Информативная область (content)** — свойство `background` элемента, порождающего блок (*подробнее см. разд. 5.1*);
- **Отступы (padding)** — свойство `background` элемента, порождающего блок;
- **Граница (border)** — специальные свойства границ элемента, порождающего блок.
- **Поля (margin)** — всегда прозрачны (transparent).

Поля, отступы и границы могут иметь различную ширину (в том числе и нулевую, а поля — даже отрицательную) по каждой из четырех сторон прямоугольника. Например, ширина верхнего поля задается свойством `margin-top`, а левого — `margin-left`; ширина правого и нижнего полей задается свойствами `margin-right и margin-bottom` соответственно. Чтобы задать одинаковую ширину в 10 пикселов для всех четырех полей, можно воспользоваться свойством `margin:10px`; выражение `margin:10px 20px 10px 20px` задает указанные значения соответственно для верхнего, правого, нижнего и левого полей. Запись такого вида еще называют сокращенной. Неплохо запомнить эту последовательность, начинающуюся от верха (top) и далее по часовой стрелке. Числовые значения задают обязательно с указанием единиц измерения, абсолютных и относительных. Аналогичные свойства имеются для отступов и границ: `padding` и `border`. Поля (`margin`) могут принимать еще значение `auto`, при котором браузер вычислит их ширину автоматически. В частности, данное значение может быть использовано для расположения элемента в центре окна браузера.

Для всех видимых элементов ширина полей, отступов и границ имеет по умолчанию нулевое значение. Исключение представляет элемент `<body>`, для которого свойство `margin` имеет ненулевые значения, причем различные в разных браузерах. Так, в Internet Explorer 7.0 и более ранних версиях значение `margin` по умолчанию равно `15px` для верха и низа и `10px` — для боковых полей; в остальных браузерах (в том числе и в Internet Explorer 8+) все поля имеют ширину `8px`. Для обеспечения межбраузерной инвариантности внешнего вида документа следует явным образом задавать значение `margin` для `<body>` в вашем документе.

В листинге 3.4 приведен HTML-документ с правилами CSS для элементов `<body>` и `<img>`. На соответствующем ему рис. 3.4 видны поля, отступы и границы элемента `<img>`. Обратите

внимание, что элемент <body> имеет ненулевое значение свойства margin, задаваемое по умолчанию. Для других элементов свойства margin, padding и border-width имеют нулевое значение и, следовательно, поля, отступы и границы не видны.

Листинг 3.4. Поля, отступы и границы

```
<!DOCTYPE html>
<html>
<head>
 <title>Отступы, границы и поля</title>
</head>
<style type="text/css">
body {
     background:#f0f0f0;   /* цвет фона */
     border-width:1px;     /* ширина границы */
     border-style:solid;   /* граница — сплошная линия */
     border-color:#000000; /* цвет границы — черный */
}
img {
     width:137px;          /* ширина картинки */
     height:193px;         /* высота картинки */
     padding:25px;         /* отступ */
     margin:15px;          /* поле */
     background:#eeeeee;   /* цвет фона */
     border-width:15px;    /* ширина границы */
     border-style:solid;   /* граница — сплошная линия */
     border-color:#c0c0c0  /* цвет границы — серый */
}
</style>
<body>
<img src="picture.gif" />
</body>
</html>
```

Границам можно назначить еще цвет и стиль. Цвет задается значением, присваиваемым параметру border-color, например

border-color : red или border-color : #ff0000

Можно также задавать цвет отдельных составляющих границы: border-top-color, border-right-color и т. д. Задание стиля (типа) границы в явном виде необходимо, чтобы она была отображена, т. к. по умолчанию граница не видна и не занимает место. Если стиль границы не задан в явном виде, другие ее параметры не будут действовать.

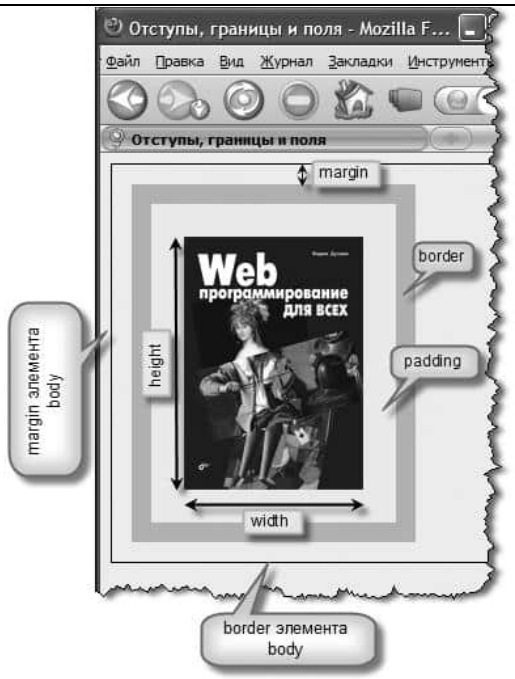

Рис. 3.4. Поля, отступы и границы картинки, заданной тегом `<img>`

Стиль четырех сторон границы задается следующими параметрами: `border-top-style`, `border-right-style`, `border-left-style`, `border-bottom-style`.

Значения стиля границы задаются ключевыми словами:

- `none` — граница не отображается (по умолчанию), даже если указан размер границы (`border-width`);
- `solid` — сплошная линия;
- `double` — двойная линия (сумма толщины двух линий и промежутка между ними определяется параметром `border-width`);
- dotted — точечная линия;
- dashed — штриховая линия;
- `grove` — вдавленный желобок;
- `ridge` — выпуклый валик;
- `inset` — весь блок элемента вдавлен;
- `outset` — весь блок элемента выпуклый.

Кроме того, задать стиль сразу всех или только некоторых сторон границы можно параметром `border-style` (без уточнений типа `top`, `right` и т. п.), подобно тому, как применяются описанные параметры `margin` и `padding`. Сокращенная запись для задания параметров всех четырех фрагментов границы имеет вид

`border:` *`стиль ширина цвет`*

Пример

```
border: solid 1px #ffbb00
```

В листинге 3.5 приведен код, иллюстрирующий применение нескольких различных стилей к элементу `<p>` (абзац). Для всех абзацев задана ширина границы `5px`. Первый абзац не стилизован (`border-style:none`) и поэтому он не имеет видимых границ. Для остальных абзацев указаны конкретные значения стилей. Различия отображения границ разными браузерами обусловлены неодинаковыми значениями цвета границ по умолчанию. С целью обеспечения межбраузерной инвариантности вы можете принудительно задать цвет для различных сторон границы с помощью свойства `border-color` или ряда свойств типа `border-top-color`, `border-right-color` и т. д.

Листинг 3.5. Стили границ

```
<!DOCTYPE html>
<html>
  <head><title>Стили границ</title>
    <style>
      body {background-color:#d0d0d0}
      p    {
        border-width:5px;
        border-style:none
      }
      #p1  {border-style:solid}
      #p2  {border-style:double}
      #p3  {border-style:groove}
      #p4  {border-style:ridge}
      #p5  {border-style:inset}
      #p6  {border-style:outset}
    </style>
  </head>
  <body>
    <p>none</p>
    <p id=p1>solid</p>
    <p id=p2>double</p>
    <p id=p3>groove</p>
    <p id=p4>ridge</p>
    <p id=p5>inset</p>
    <p id=p6>outset</p>
  </body>
</html>
```

В CSS 3 имеются стилевые параметры `border-radius` и `border-image` для оформления границ, которые существенно расширяют возможности дизайнеров. Подробнее они будут рассмотрены в *разд. 5.3, 5.4*

Итак, любой элемент занимает на экране некоторую прямоугольную область. Размеры области содержимого прямоугольного блока определяются свойствами `width` (ширина) и `height` (высота). По умолчанию эти свойства имеют значение `auto`, при котором размеры элемента выбираются автоматически так, чтобы по возможности отобразить его целиком. Так, по

умолчанию элемент вместе с полями, границами и отступами занимает всю ширину клиентской области браузера, а по высоте — столько, сколько необходимо для отображения целиком содержащихся в нем элементов плюс поля, границы и отступы. Вместе с тем, ширину и высоту можно задать явным образом числами с указанием единиц измерения, например width:120px; height:50mm, width:80%. Однако в этом случае возможны различные варианты отображения элементов. Более подробно параметры width и height будут рассмотрены в *разд. 4.9*.

Если указать свойства width и height для графического элемента (<img>), то картинка будет сжата или растянута в зависимости от ее исходных размеров. В случае контейнерных тегов (например,) видимое взаимное расположение собственно контейнера и его содержимого может быть различным. Например, картинка (<img>), расположенная в контейнере , в окне браузера может оказаться вне прямоугольника своего контейнера. Иначе говоря, видимое расположение элементов может не соответствовать логической структуре документа. Рассмотрим несколько примеров.

В листинге 3.6 приведен HTML-документ, в котором три контейнера содержат тексты, а на рис. 3.5 показан его внешний вид. Браузер создает блоки с указанными в CSS размерами, но при этом содержимое может выходить за их границы. В данном случае строка Текст2Текст2Текст2 без внутренних пробелов не поместилась во второй (вложенный) контейнер . Текстовая строка в третьем состоит из слов, разделенных пробелами, и поэтому браузер с помощью переносов разместил ее в блоке.

Листинг 3.6. Контейнеры с текстовым содержимым

```
<!DOCTYPE html>
"http://www.w3.org/TR/HTML4.01/loose.dtd">
<html>
<head><title>div с текстом</title>
<style type="text/css">
div {
     width:100px;
     height:120px;
     background:#cccccc;
     padding:10px;
     margin:20px;
     border-width:1px;
     border-style:solid; /* сплошная линия */
     border-color:#000000
}
#indiv { /* вложенный div */
     width:50px;
     height:60px;
     background:#eeeeee;
     padding:10px;
     margin:20px;
     border-width:1px;
     border-style:solid;
     border-color:#000000
```

```
}
</style>
</head>
<body>
<div>
  Текст1
  <div id="indiv">
    Текст2Текст2Текст2
  </div>
</div>
<div>
  Текст3 Текст3 Текст3 Текст3
</div>
</body>
</html>
```

В листинге 3.7 приведен HTML-документ, в котором контейнер `<div>` содержит графическое изображение, а его внешний вид в браузере иллюстрирует рис. 3.6.

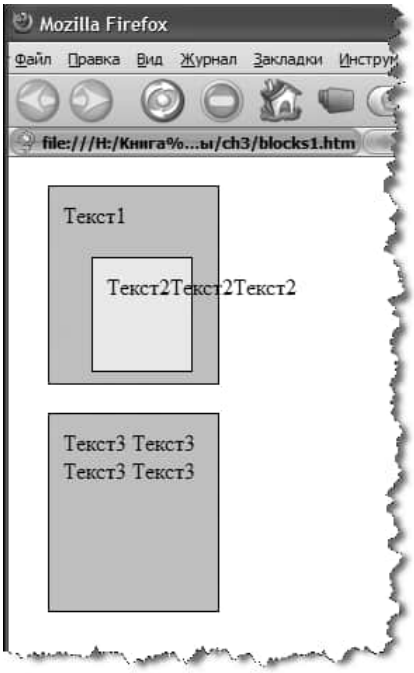

Рис. 3.5. Вид HTML-документа листинга 3.6 в браузере

Рис. 3.6. Вид HTML-документа листинга 3.7 в браузере

Листинг 3.7. Контейнер `<div>` с графическим изображением

```
<!DOCTYPE html>
<html>
```

```
<head><title>div с графикой</title>
<style type="text/css">
div {
     width:100px;
     height:120px;
     background:#cccccc;
     padding:10px;
     margin:20px;
     border-width:1px;
     border-style:solid;
     border-color:#000000
}
img {
     background:white;
     padding:10px;
     margin:20px;
     border-width:5px;
     border-style:solid;
     border-color:#000000
}
</style>
</head>
<body>
<div>
  <img src="picture.jpg"/>
</div>
</body>
</html>
```

Мы рассмотрели несколько случаев, когда содержимое контейнера полностью в нем не помещается. Однако отображением содержимого контейнера можно управлять. Как это делать, будет рассказано в *главе 4*.

3.6. Наследование параметров

Некоторые стилевые параметры, установленные для одних элементов, наследуются их дочерними элементами. Например, цвет текста (`color`), установленный для элемента `<body>`, будет иметь место и для всех вложенных в него других элементов (например, `<h1>`, `<p>`, `<div>` и т. д.), если только для них данный параметр не задан явно. Наследование параметров позволяет сэкономить место и время при создании таблиц стилей.

В листинге 3.8 представлен код документа с тремя элементами `<div>`, причем второй вложен в первый. Весь текст внутри первого и второго `<div>`, включая заголовок `<h1>`, будет красным, т. к. параметр `color:red` здесь явно задан для первого `<div>` и наследуется вложенным в него вторым `<div>`. Текстовое содержимое третьего `<div>` будет зеленым, поскольку параметр `color:green` задан для родительского элемента `<body>`.

Листинг 3.8. Наследование параметров

```
<!DOCTYPE html>
<html>
<head><title>Наследование</title>
<style type="text/css">
body {
  color:green
}
#div1 {
  border: solid 1px; /* граница */
  color:red          /* цвет текста */
}
</style>
</head>
<body>
<div id="div1">Блок 1
  <div id="div2">Блок 2
    <h1>Это некоторый заголовок</h1>
  </div>
</div>
<div id="div3">Блок 3</div>
</body>
</html>
```

По умолчанию наследуются все параметры шрифта (`font`) и списков (`list`), цвет (`color`), многие параметры текста (`text`) и некоторые другие. Однако большинство параметров CSS по умолчанию не наследуется. Если они не заданы явно ни в авторской, ни в пользовательской таблицах стилей, то получают некоторые значения по умолчанию, принятые браузером. Например, параметры границы (`border`) не наследуются по умолчанию. В листинге 3.8 для первого элемента `<div>` задана граница (`border: solid 1px`), но ее не будет у вложенного (дочернего) второго элемента `<div>`.

Вместе с тем, можно принудительно задать наследование параметра, т. е. установить такое же значение, что и у родительского элемента. Для этого достаточно в определении параметра присвоить ему значение `inherit`. Например, если в таблицу стилей листинга 3.8 добавить правило `#div2 {border:inherit}`, то второй элемент `<div>` отобразится с видимой границей.

Глава 4

Позиционирование с помощью CSS

Любой видимый элемент (X)HTML-документа с помощью CSS можно расположить (позиционировать) на Web-странице практически как угодно, вопреки его естественному положению, диктуемому нормальным потоком (*см. разд. 2.6*). Таким образом, CSS позволяет сверстать Web-страницу подобно странице какого-нибудь глянцевого журнала или газеты.

На рис. 4.1 показаны код и внешний вид HTML-документа, отображаемого в нормальном потоке. Здесь CSS служит только для того, чтобы окрасить фон и обозначить границы контейнеров `<div>` в нормальном потоке, не изменяя их естественное позиционирование. Как и положено, три контейнера `<div>` блокового типа следуют один ниже другого в естественном порядке, т. е. в порядке их упоминания в (X)HTML-коде.

Посредством CSS можно задать параметры `top` и `left` — вертикальную и горизонтальную координату верхнего левого угла элемента соответственно. Можно также задать параметры `bottom` и `right` — соответственно вертикальную и горизонтальную координату для правого нижнего угла элемента. Но этого еще недостаточно. Необходимо прежде указать с помощью параметра `position` способ позиционирования. Данный параметр определяет, как следует понимать принудительное позиционирование элемента и располагать остальные элементы в нормальном потоке. Кроме того, от значения параметра `position` зависит начало отсчета координат элементов `top`, `left`, `bottom` и `right`.

Итак, чтобы позиционировать элемент в соответствии с заданными координатами, необходимо указать способ позиционирования и собственно требуемые координаты верхнего левого угла элемента.

Возможные значения параметра `position`:

- `static` (статический);
- `relative` (относительный);
- `absolute` (абсолютный);
- `fixed` (фиксированный).

Далее мы рассмотрим способы позиционирования более подробно.

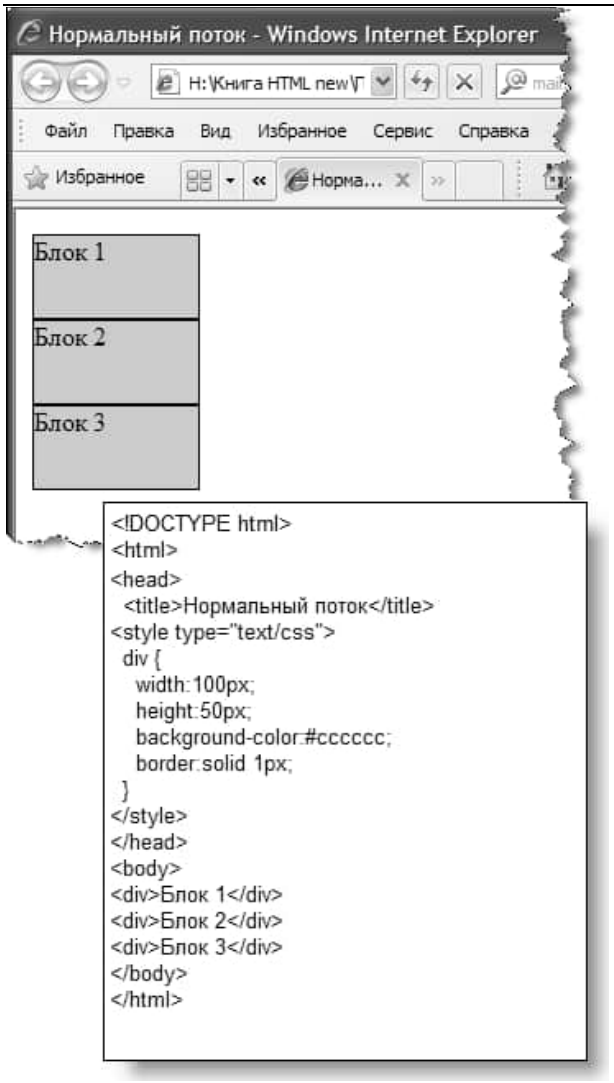

Рис. 4.1. Позиционирование трех контейнеров `<div>` в нормальном потоке

4.1. *position:static*

Статическое позиционирование (`position:static`) аналогично расположению элемента в нормальном потоке, при этом координаты `top`, `left`, `bottom` и `right` не играют никакой роли. Поэтому в CSS данный способ практически никогда не задают с помощью параметра `position:static`.

4.2. *position:relative*

При относительном позиционировании (`position:relative`) элемент размещается по координатам, задаваемым параметрами `top` и `left` относительно своего положения в

нормальном потоке. Иначе говоря, `top` и `left` определяют смещение элемента относительно его естественного положения. Последующие элементы в нормальном потоке позиционируются так, будто данный элемент продолжает оставаться в нормальном потоке, т. е. занимать свое естественное положение. На рис. 4.2 приведен пример относительного позиционирования второго элемента `<div>`. Обратите внимание на пустое место между первым и третьим элементами `<div>`, освобожденное вторым элементом `<div>`, который смещен посредством параметров `top` и `left`. Третий `<div>`, будучи в нормальном потоке, позиционируется так, будто второй `<div>` остается в том же нормальном потоке.

Предположим, ваш документ отображается в нормальном потоке и требуется немного скорректировать положение лишь некоторых элементов, оставив другие на своих местах. Тогда лучше всего это сделать путем относительного позиционирования.

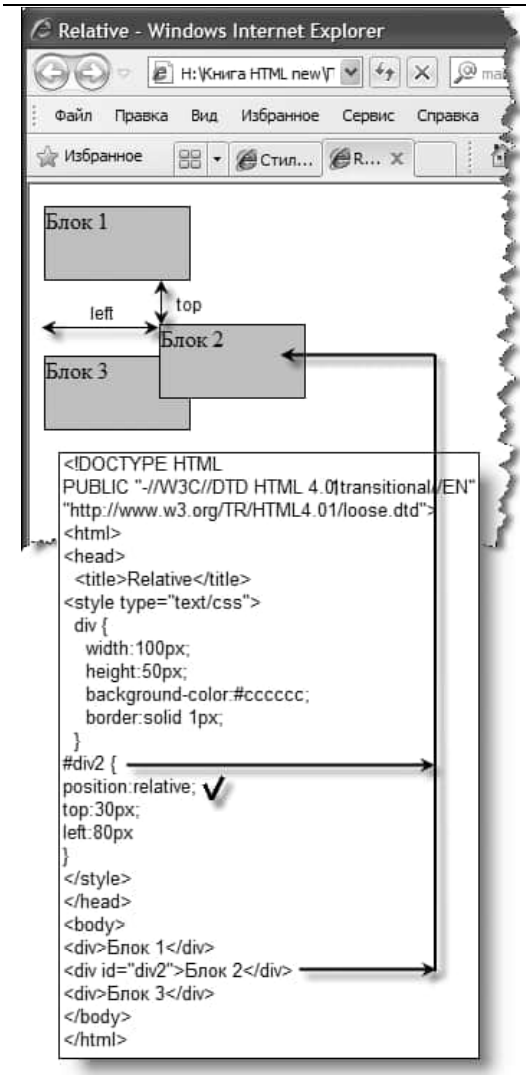

Рис. 4.2. Относительное позиционирование второго контейнера `<div>`

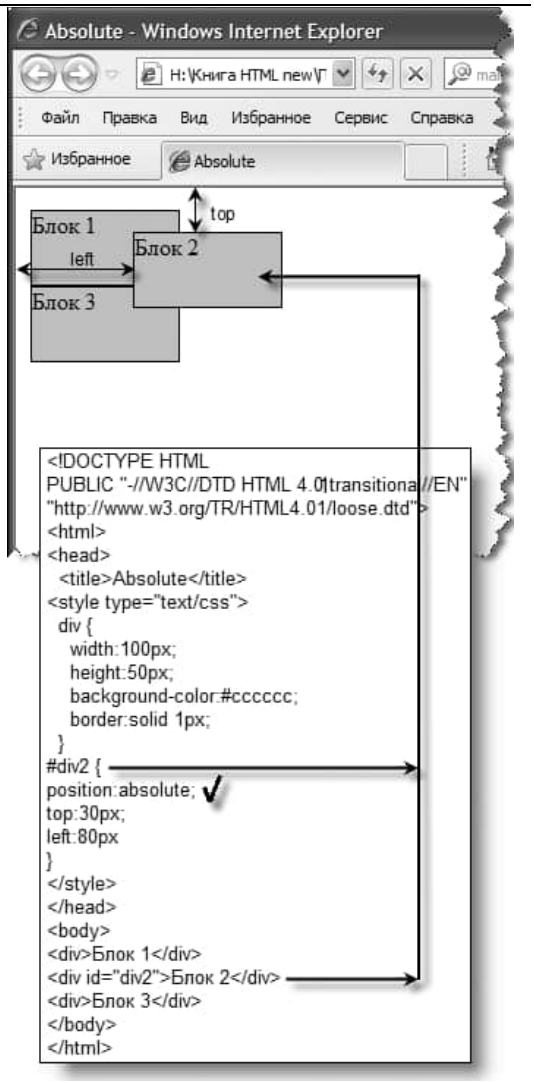

Рис. 4.3. Абсолютное позиционирование второго контейнера `<div>`

4.3. *position:absolute*

При абсолютном позиционировании (`position:absolute`) элемент располагается по координатам, задаваемым параметрами `top`, `left`, `bottom` и `right` относительно содержащего его контейнера. Достаточно указать какую-нибудь одну пару координат. Координаты `top` и `left` задают положение верхнего левого угла элемента относительно верхнего левого угла содержащего контейнера. Координаты `bottom` и `right` определяют положение нижнего правого угла элемента и отсчитываются от нижнего правого угла контейнера.

Если задать все четыре координаты для контейнерного тега, то у прямоугольной области

соответствующего элемента будут позиционированы одновременно два угла — верхний левый и нижний правый. Некоторые элементы (например, <img>) позиционируются посредством какой-нибудь одной пары координат. Если указаны две пары, то предпочтение отдается первой, т. е. top и left.

При абсолютном позиционировании последующие элементы в нормальном потоке располагаются так, будто данный элемент изъят из нормального потока. На рис. 4.3 приведен пример абсолютного позиционирования второго элемента . В данном случае свойства top и left задают координаты элемента относительно верхнего левого угла документа (окна браузера). Обратите внимание на то, что абсолютно позиционируемый элемент освобождает место, которое он занимал бы в нормальном потоке. Освободившееся место занимает последующий элемент из нормального потока.

Теперь рассмотрим подробнее, относительно чего абсолютно позиционируется элемент. В предыдущем примере (рис. 4.3) второй абсолютно позиционировался относительно окна браузера. Если элемент находится непосредственно в контейнере <body>, то он позиционируется относительно верхнего левого угла документа (клиентской области окна браузера). Напомню, что если вы не упомянули в своем (X)HTML-коде тег <body>, то соответствующий ему контейнер все равно автоматически появится в структуре документа и, более того, он будет иметь некоторые параметры CSS, принятые по умолчанию данным браузером, в том числе и параметры позиционирования. В рассмотренном примере все теги находятся в одном и том же контейнере <body>, а потому любой из них абсолютно позиционируется относительно левого верхнего угла документа.

А что будет, если элемент находится в другом контейнере, отличном от <body>? Результат зависит от того, как позиционирован контейнер, содержащий данный элемент.

На рис. 4.4 показаны четыре элемента , расположенные в нормальном потоке, поскольку в CSS не задано их позиционирование. При этом четвертый вложен во второй. Размеры данных элементов таковы, что вложенность соответствующих им блоков хорошо видна на рисунке.

На рис. 4.5 показаны два варианта того же HTML-документа, но с позиционированием второго и четвертого элементов . В обоих примерах четвертый позиционирован абсолютно (position:absolute), а содержащий его второй — по-разному.

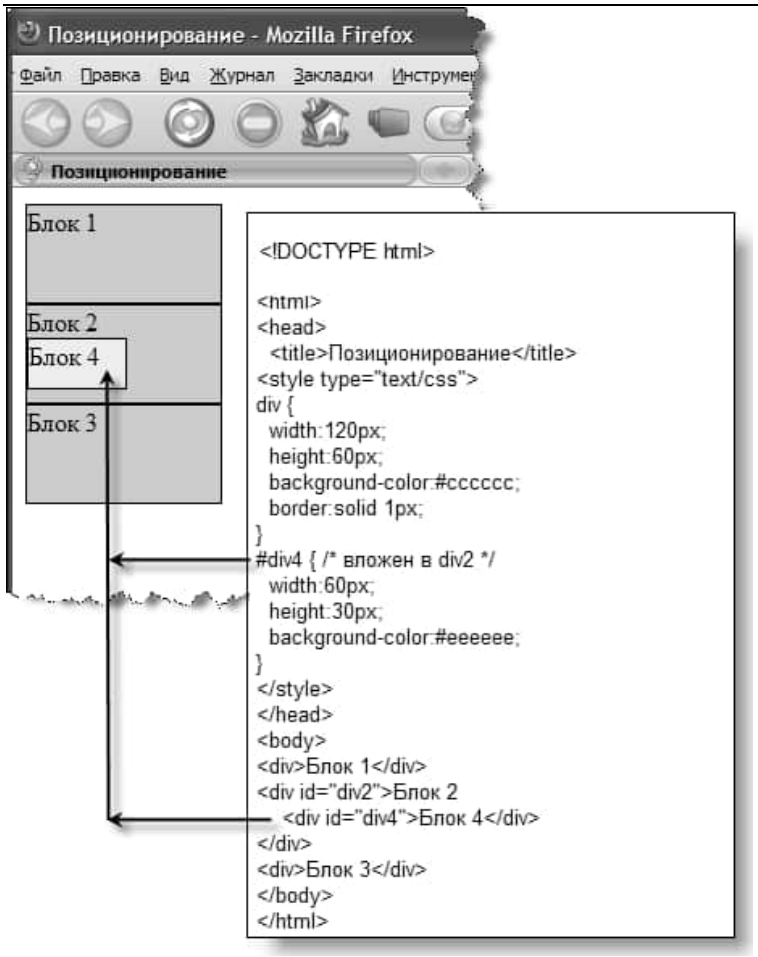

Рис. 4.4. Элементы `<div>` в нормальном потоке, при этом четвертый `<div>` вложен во второй

Так, в первом варианте (сверху на рис. 4.5) второй `<div>` имеет относительное позиционирование (`position:relative`). Поскольку при этом параметры `top` и `left` не заданы, то данный элемент остается на месте (смещение относительно его положения в нормальном потоке равно нулю). Однако четвертый `<div>` позиционируется относительно верхнего левого угла содержащего его второго `<div>`, т. е. своего контейнера, а не окна браузера.

Во втором варианте (снизу на рис. 4.5) второй `<div>` имеет абсолютное позиционирование (`position:absolute`). Поскольку его ближайшим контейнером служит `<body>`, то он расположится относительно окна браузера, выйдя из нормального потока. Третий `<div>` займет освободившееся место. При этом четвертый `<div>` позиционируется относительно второго так же, как и в первом варианте.

Пусть родительский элемент не позиционирован относительно или абсолютно, а его дочерний элемент позиционирован. Тогда он будет размещен относительно ближайшего прародительского элемента. Если таковым является `<body>`, то элемент позиционируется относительно окна браузера.

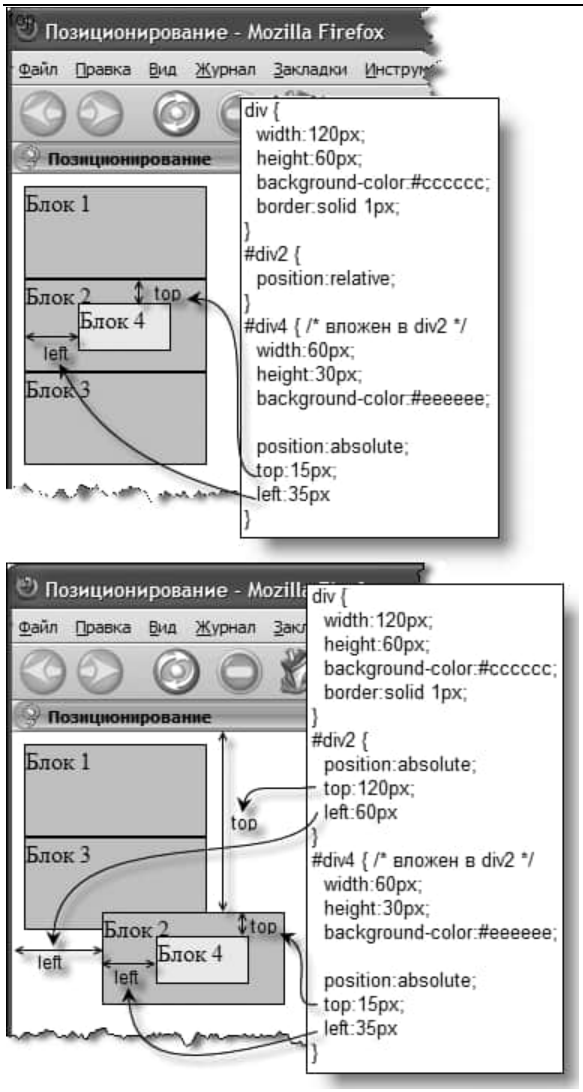

Рис. 4.5. Позиционирование элементов `<div>`, при этом четвертый `<div>` вложен во второй

А что произойдет, если второй `<div>` позиционировать абсолютно, не задавая координаты явно? Он отобразится на том же месте, что и в нормальном потоке или при относительном позиционировании (`position:relative`), но при этом будет изъят из нормального потока, освободив свое место другому элементу нормального потока. В результате может возникнуть перекрытие первого вторым. В примере, показанном на рис. 4.6, второй `<div>` оказался отображенным над третьим и частично закрыл его.

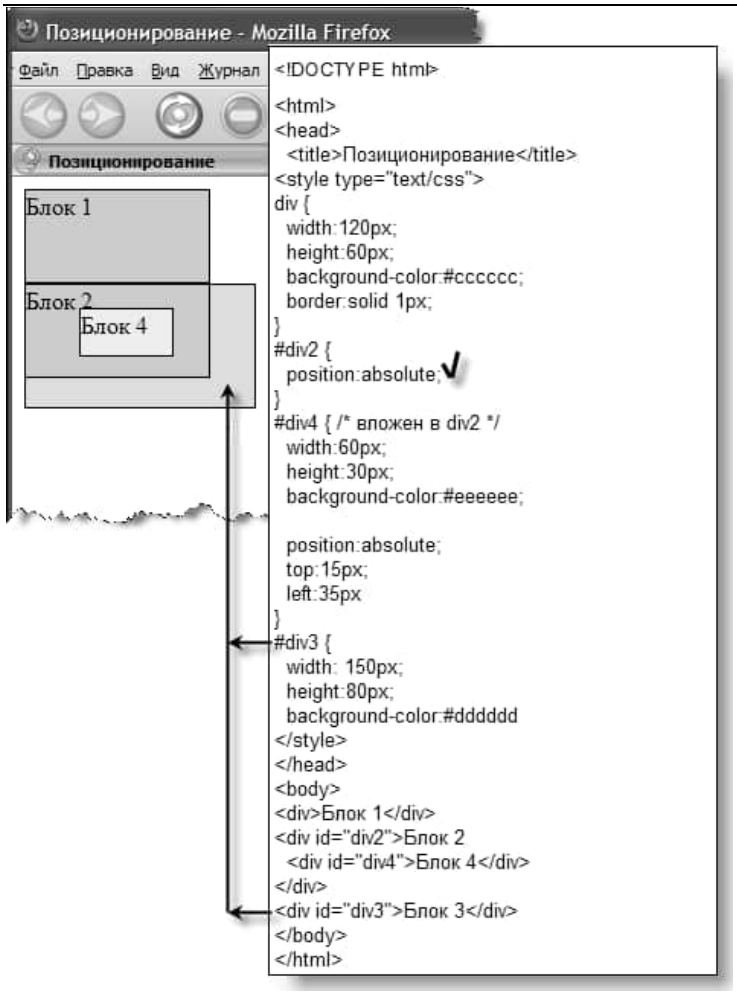

Рис. 4.6. Позиционирование элементов `<div>`, при этом третий `<div>` находится под вторым

4.4. *position:fixed*

Фиксированное позиционирование (`position:fixed`) позволяет зафиксировать элемент так, чтобы он оставался видимым при прокрутке содержимого Web-страницы и занимал неизменное положение относительно окна браузера. При этом собственно позиционирование аналогично абсолютному, просто появляется дополнительный эффект. На рис. 4.7 второй элемент `<div>` имеет параметр `position:fixed`, слева внизу показана страница до ее вертикальной прокрутки, а справа — после. В обоих случаях положения второго `<div>` относительно окна браузера одинаковы.

В исходном состоянии второй `<div>` имеет те же координаты, что и при расположении в нормальном потоке, поскольку других ему в данном примере не назначено. Тем не менее, он изъят из нормального потока (как и в случае абсолютного позиционирования), а потому перекрывает следующий за ним третий `<div>`.

```
<!DOCTYPE html>

<html>
<head>
<title>Fixed</title>
<style type="text/css">
  div {
    width:100px;
    height:50px;
    background-color:#cccccc;
    border:solid 1px;
  }
  #div2 {
    position:fixed;
    width:80px;
    height:30px;
    background-color:#dddddd;
  }
</style>
</head>
<body>
<div>Блок 1</div>
<div id="div2">Блок 2</div>
<div>Блок 3</div>
<div>Блок 4</div>
<div>Блок 5</div>
</body>
</html>
```

Рис. 4.7. Фиксированное позиционирование

4.5. Отсчет координат

В предыдущих разделах уже упоминалось, относительно чего отсчитываются координаты `top` и `left` позиционируемого элемента. Здесь мы сделаем несколько уточняющих замечаний для случая абсолютного позиционирования (`position:absolute`) элементов.

Рассмотрим два элемента, один из которых непосредственно вложен в другой или, иначе говоря, один элемент дочерний по отношению к родительскому. Родительский элемент

может быть в свою очередь дочерним по отношению к другому элементу и т. д. Родительский элемент наивысшего уровня — <body>. Так что любой элемент имеет, по крайней мере, одного родителя — элемент <body>. Итак, непосредственным родителем для элемента может быть <body> или какой-нибудь другой элемент. Однако эти две ситуации с точки зрения отсчета координат существенно различаются.

Если непосредственным родительским элементом является <body>, то его дочерний элемент абсолютно позиционируется относительно клиентской области окна браузера, т. е. значения параметров top и left отсчитываются соответственно от верхней и левой границ окна до верхнего и левого краев поля (margin) элемента.

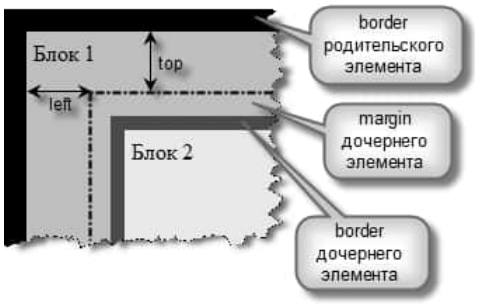

Рис. 4.8. Отсчет координат при абсолютном позиционировании, когда непосредственный родительский элемент отличен от <body>

Если непосредственный родительский элемент отличен от <body>, то его дочерний элемент абсолютно позиционируется относительно родительского: значения параметров top и left отсчитываются соответственно от внутренних краев границы (border) родительского элемента до верхнего и левого краев поля (margin) дочернего элемента (рис. 4.8).

4.6. Слои

Порядком отображения (над/под) элементов можно управлять с помощью параметра z-index, который принимает целочисленные значения. Он определяет третье измерение для Web-страницы, т. е. уровень слоя, в котором находится элемент. Чем больше значение этого параметра, тем выше (ближе к переднему плану) будет находиться элемент, и наоборот (рис. 4.9). По умолчанию все элементы находятся в одном слое: значение параметра z-idex для них равно нулю.

Как и другие параметры, задающие координаты, z-index действует, только если указан способ позиционирования (position) со значениями relative, absolute или fixed.

На рис. 4.10 в качестве примера показано абсолютное позиционирование двух элементов . Координаты top и left выбраны так, что элементы перекрываются. Поскольку второй упомянут в HTML-коде последним, то он располагается над первым , как показано в верхней части рисунка. Однако этот порядок можно изменить, если установить достаточно большое значение параметра z-index для первого (нижняя часть рисунка). В рассматриваемом случае достаточно было задать z-index:1. Другой пример применения параметра z-index рассмотрен в *разд. 4.10.2*.

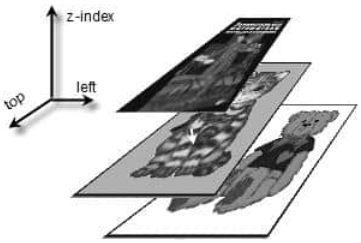

Рис. 4.9. Чем больше значение `z-index`, тем ближе к переднему плану будет находиться элемент

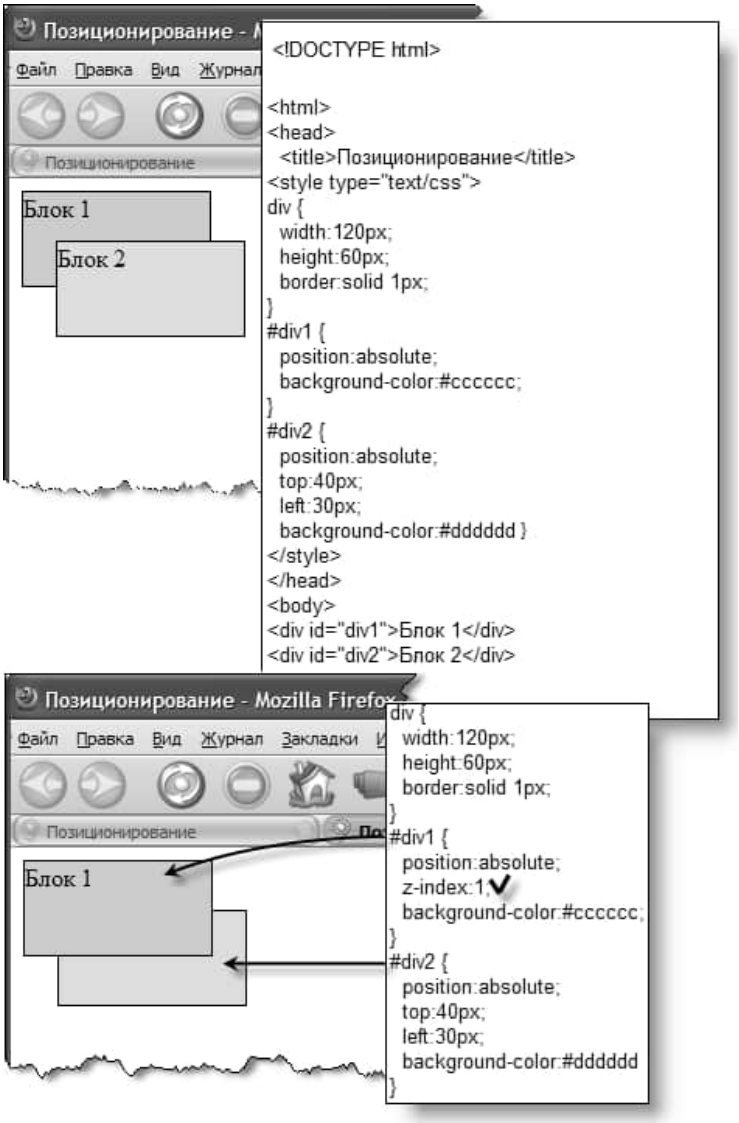

Рис. 4.10. Управление расположением элементов с помощью параметра `z-index`

4.7. Обтекание

Способы позиционирования, рассмотренные в предыдущих разделах, позволяют решить все задачи расположения элементов на Web-странице. Вместе с тем имеются и другие возможности.

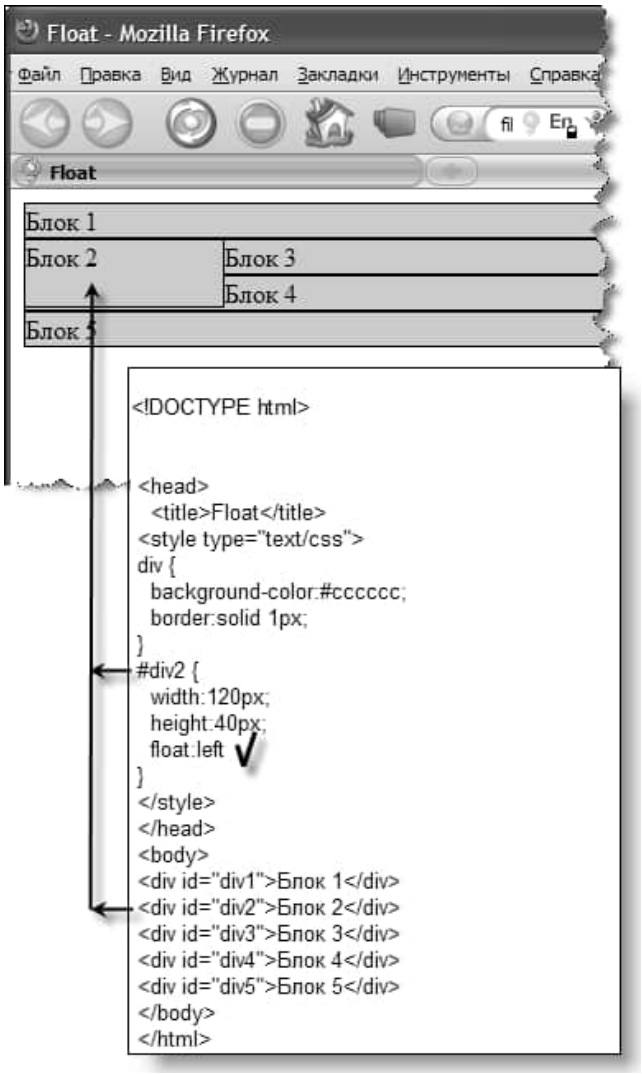

Рис. 4.11. Позиционирование с помощью параметра `float`

Положение элементов в нормальном потоке можно изменить с помощью параметра `float`, который принимает значения `left` и `right`. Данный параметр смещает элемент по горизонтали, прижимая его к левому или правому краю контейнера. При этом последующие элементы в нормальном потоке располагаются справа или слева от сдвинутого элемента, если, конечно, позволяет освободившееся место. Таким образом, они как бы обтекают элемент с параметром `float`. На рис. 4.11 второй из пяти элементов `<div>` смещен влево (`float:left`),

а остальные его обтекают справа. Третий и четвертый элементы сдвинулись вверх и заняли освободившееся место справа от второго, а для пятого `<div>` места справа не нашлось, и поэтому он остался ниже четвертого. При отсутствии параметра `float:left` для второго `<div>` все элементы `<div>` в данном примере расположились бы друг за другом по вертикали.

Если в рассматриваемом примере для второго `<div>` не указывать размеры (`width` и `height`), они будут заданы автоматически минимальными так, чтобы поместилось их содержимое. На рис. 4.12 показан соответствующий пример, в котором применяется обтекание как слева, так и справа.

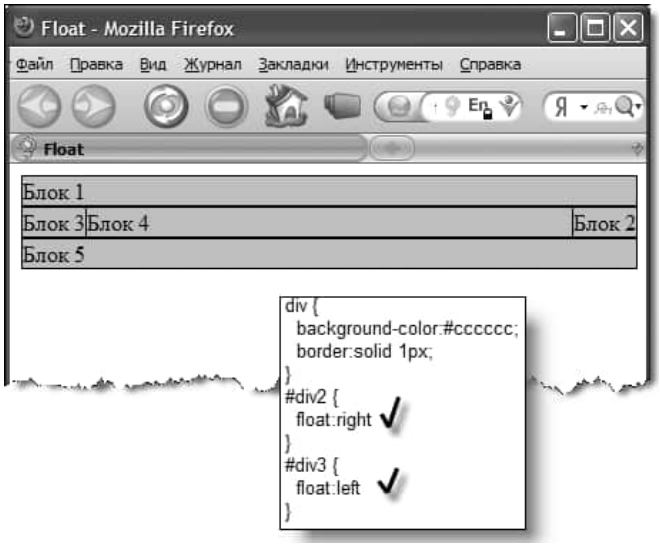

Рис. 4.12. Позиционирование с помощью параметров `float:left` и `float:right`

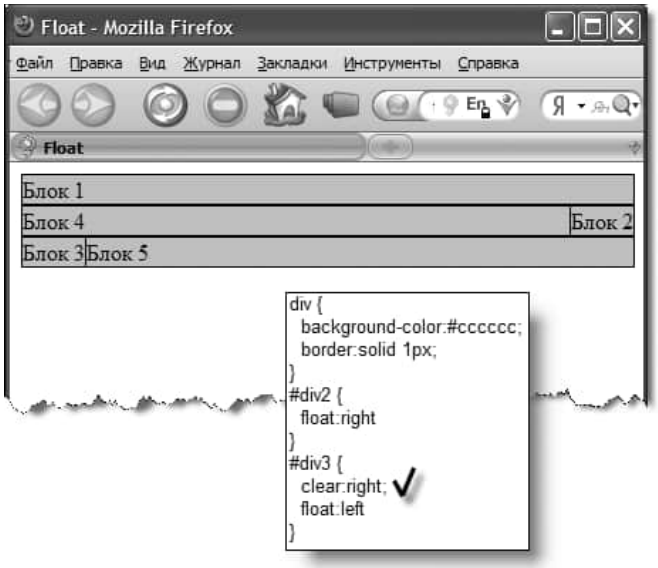

Рис. 4.13. Применение параметра `clear`

Отменить ранее установленный режим обтекания элемента позволяет параметр `clear`, принимающий значения `left`, `right` и `both` (для отмены обтекания как слева, так и справа). На рис. 4.13 показан пример, в котором для третьего `<div>` отменяется режим обтекания, установленный для предыдущего элемента (второго `<div>`), и назначается режим обтекания справа. Сравните данный рисунок с рис. 4.12.

С помощью параметра `float` удобно управлять относительным расположением картинок и текста. На рис. 4.14 приведен пример расположения текста и графических изображений. В контейнере `<p>` находится некоторый текст и две картинки. Первое графическое изображение прижимается к левому краю окна (`float:left`) и обтекается текстом справа, а второе прижимается к правому краю окна (`float:right`) и обтекается текстом слева. Без указания параметра `float` картинки располагались бы в нормальном потоке в зависимости от текущих размеров окна браузера, как показано на рис. 4.15. Очевидно, что такое расположение текста с картинками не годится для публикации иллюстрированной статьи.

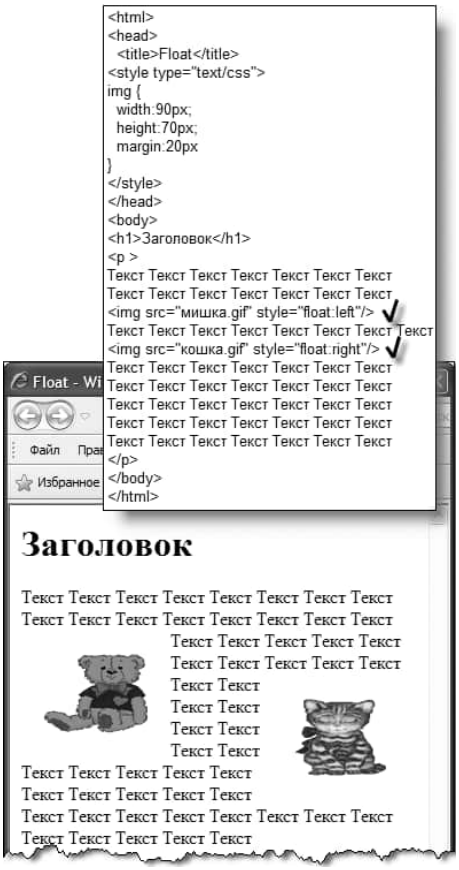

Рис. 4.14. Позиционирование текста и картинок с помощью параметра `float`

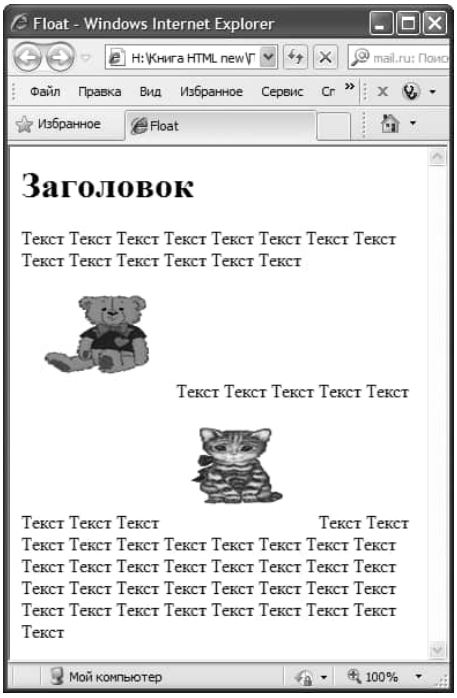

Рис. 4.15. Позиционирование текста и картинок в нормальном потоке, т. е. при отсутствии параметра `float`

4.8. Видимость

4.8.1. *overflow*

Может случиться, что содержимое контейнера полностью в нем не помещается, т. е. размеры контейнера меньше размеров его содержимого. Например, вы создали контейнер `<div>` с заданными размерами `width=500px` и `height=100px` и поместили в него фотографию размером 1280×1024 px, полученную посредством цифровой фотокамеры, даже не задав нужных размеров для ее публикации в Web. Эту ситуацию еще называют переполнением контейнера. Как отобразится содержимое переполненного контейнера? По умолчанию оно будет показано в полном объеме, как будто границ контейнера не существует (см. рис. 3.6). Тем не менее, в таких случаях отображением содержимого контейнера можно управлять с помощью параметра `overflow`, который принимает следующие значения:

- `visible` — если содержимое выйдет за пределы контейнера, оно все равно будет показано полностью (значение по умолчанию);
- `hidden` — выступающие за границы контейнера части содержимого будут обрезаны, т. е. окажутся невидимыми (это значение принято по умолчанию в Intenet Explorer, но не в других браузерах);
- `auto` — добавляет полосы прокрутки, если содержимое выходит за границы элемента-контейнера;
- `scroll` — добавляет полосы прокрутки в любом случае.

4.8.2. *clip*

Параметр `clip` позволяет показать некоторую прямоугольную часть содержимого элемента (например, графического изображения), который абсолютно позиционирован (`position:absolute`). Данный параметр принимает два значения:

- `auto` — элемент полностью виден (значение по умолчанию);
- `rect(`*`top right bottom left`*`)` — в скобках, разделяя пробелом, указывают верхнюю, правую, нижнюю и левую координаты, задающие прямоугольную область содержимого элемента, которую требуется показать.

На рис. 4.16 показано применение параметра `clip:rect(top right bottom left)` к графическому изображению. Обратите внимание, от чего отсчитываются координаты.

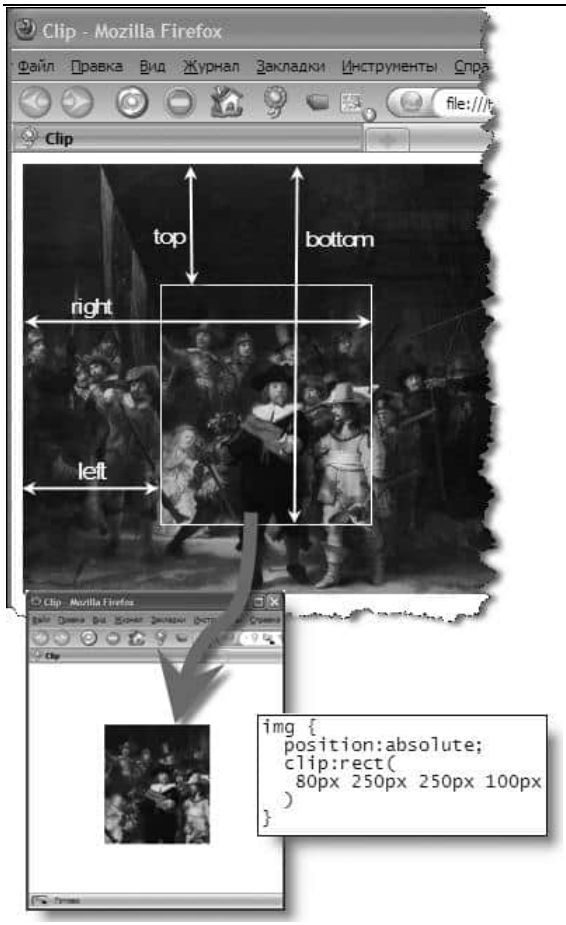

Рис. 4.16. Применение параметра `clip`

4.8.3. *visibility*

Иногда в документе требуется использовать элемент, но не показывать его. Например, можно загрузить в браузер таблицу с данными, которые будут как-то задействованы сценарием, но пользователь не должен их видеть. Элементы, по умолчанию видимые в окне браузера, можно сделать невидимыми с помощью параметра `visibility`, который принимает следующие значения:

- `visible` — элемент виден (задано по умолчанию);
- `hidden` — элемент не виден (скрыт), однако занимает на странице место, т. е. он прозрачен, а остальные элементы нормального потока отображаются так, будто бы он виден;
- `inherit` — элемент отображается, если виден его родительский элемент;
- `auto` — добавляет полосы прокрутки, если содержимое выходит за пределы контейнера;
- `scroll` — добавляет полосы прокрутки в любом случае.

4.8.4. *display*

Управлять видимостью, а также изменять расположение элемента можно с помощью параметра `display`, принимающего следующие значения:

- `none` — элемент не виден, при этом он изымается из нормального потока (в отличие от параметра `visibility:hidden`), т. е. не занимает место на странице. Значит, последующие элементы документа в нормальном потоке отобразятся так, будто невидимого элемента нет в документе;
- `block` — элемент создает структурный блок, который визуализируется в нормальном потоке как блоковый элемент;
- `inline` — элемент создает внутристрочный блок.

> ***Примечание***
> Здесь перечислены основные, но далеко не все возможные значения параметра `display`.

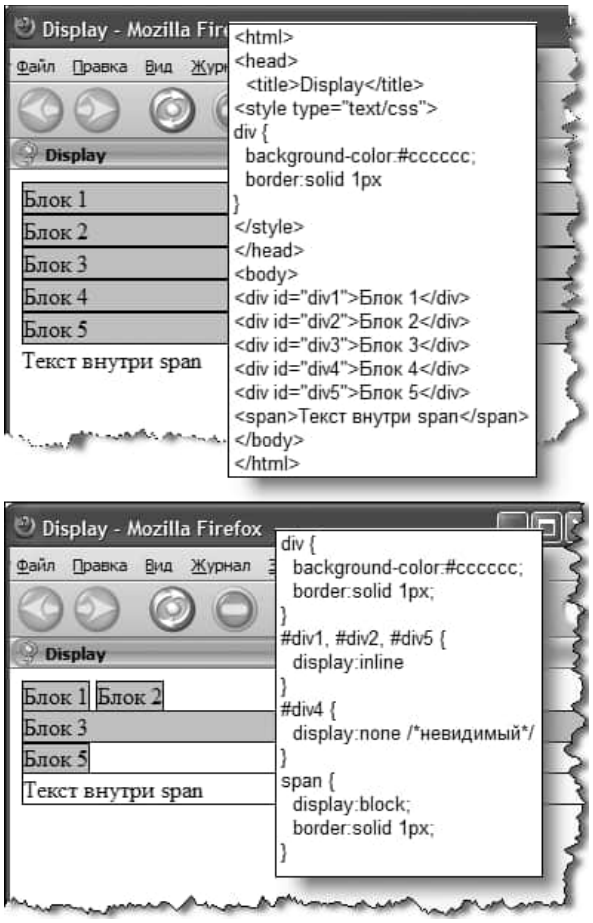

Рис. 4.17. Влияние параметра `display` на отображение блоковых и внутристрочных элементов

В верхней части рис. 4.17 показаны пять блоковых элементов `<div>` и один внутристрочный элемент `<span>`, расположенные в нормальном потоке при отсутствии параметра `display`, а в

нижней части эти элементы расположены иначе благодаря использованию данного параметра. Так, три блоковых элемента (div1, div2 и div5) превращаются во внутристрочные блоки, div4 становится невидимым, выпадая из потока, а внутристрочный элемент превращается в блоковый.

Таким образом, с помощью параметра display можно удалить из нормального потока элемент, а также превратить блоковый элемент во внутристрочный и наоборот. Данный параметр широко применяется при создании раскрывающихся меню. Его значением управляют посредством сценариев.

4.9. Размеры

Размеры элементов задаются параметрами width (ширина) и height (высота) в линейных единицах или процентах (*см. разд. 3.4*). Данные параметры не включают в себя соответствующие величины полей, границ и отступов (см. рис. 3.3). Размеры в процентах задают относительно аналогичных величин родительского элемента.

Примеры:

```
div {width:300px; height:50%}
p {width:15cm; height:20mm}
```

> ***Примечание***
> Указанное определение размеров элементов соответствует блочной модели от W3C, которую поддерживают все современные браузеры. Internet Explorer ранних версий (4, 5 и 5.5) и Opera 7 реализовали модель, в которой размеры включали в себя отступы и границы. В разрабатываемой спецификации CSS 3 введен параметр box-sizing со значениями content-box для указания на использование стандартной модели W3C и border-box — для модели Internet Explorer 5.

Если параметры width и height не указывать, то размеры элемента на Web-странице будут зависеть от его типа. Например, ширина блоковых элементов и <p> определяется так, чтобы они вместе с полями, границами и отступами заняли всю ширину клиентской области браузера, а высота определяется так, чтобы содержимое данных элементов поместилось целиком. Графическое изображение из файла, вставляемое в документ посредством внутристрочного тега <img>, будет иметь оригинальные размеры в пикселах. Элемент <object>, представляющий содержимое внешнего файла (например, видео- или аудиоклипа), имеет некоторые размеры в зависимости от используемого плагина.

Если размеры заданы в абсолютных единицах, то блоковые элементы (например, и <p>) теряют свое свойство изменять их в зависимости от текущих ширины окна браузера и объема содержимого. При указании относительных единиц (например, процентов) данное свойство сохраняется, если только родительский элемент имеет относительные размеры.

На рис. 4.18 в качестве примера показан HTML-документ, в котором второй элемент вложен в первый. При этом размеры первого заданы в пикселах, а второго — в процентах относительно размеров родительского элемента, т. е. первого .

Кроме параметров width и height, можно задавать экстремальные (т. е. минимальные и максимальные) размеры: min-width, min-height и max-width, max-height. Их действие легче понять в случае, когда соответствующие обычные размеры (width и height) не заданы.

Рассмотрим сначала случай, когда для элемента `<div>` не указан параметр `width`, но задан `min-width`. Тогда ширина данного элемента должна быть не меньше значения параметра `min-width`. Какой именно она будет, зависит от текущей ширины клиентской области браузера. Больше она может стать, только если клиентская область браузера достаточно широка, чтобы вместить элемент. Если задан параметр `max-width`, то ширина элемента не может быть больше значения данного параметра. Меньше она становится только в том случае, если ширина клиентской области слишком мала, чтобы вместить элемент. Параметры `min-height` и `max-height` действуют аналогично.

Примечание

Различные браузеры позволяют изменять размеры окна браузера в разной степени, и может оказаться, что при достаточно малых установленных значениях `min-width` действительная минимальная ширина элемента будет несколько больше. Моменты появления полос прокрутки также могут различаться. Иначе говоря, в различных браузерах эффекты изменения размеров элемента при вариации размеров окна могут несколько отличаться. Так, например, в рассматриваемом отношении Internet Explorer 8, Opera 10.5, Crome 4.1 и Safari 4 ведут себя одинаково, а Firefox 3.6 иначе.

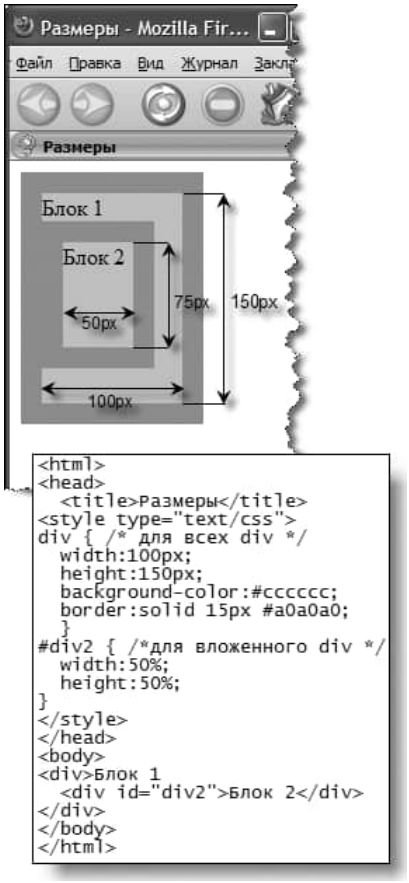

Рис. 4.18. Установка размеров для блоковых элементов

Совместное применение `min-width` и `max-width` (`min-height` и `max-height`) позволяет регулировать ширину (высоту) элемента в заданных пределах при изменении размеров окна браузера. Это полезно при определении, например, области, занимаемой текстом. Такой текст можно заключить в тег `<div>` или/и `<p>` и сделать так, чтобы занимаемая им область могла расширяться или сужаться в зависимости от размеров окна браузера, но в заданных пределах.

Более сложная ситуация возникает, когда заданы как обычные, так и экстремальные размеры, причем одного и того же типа (ширина или длина). Например, параметры `width` и `min-width` относятся к одному типу — ширине, а `width` и `min-height` — к разным. Эффект совместного применения данных параметров при изменении размеров окна браузера зависит от соотношения значений параметров и типа единиц их измерения (абсолютные и относительные). Не останавливаясь на изучении данного случая, отмечу лишь, что его нетрудно исследовать экспериментальным путем.

4.10. Практические примеры

Примеры, рассмотренные в предыдущих разделах, служили в основном для изучения параметров CSS. Теперь перейдем к решению некоторых простых, но практически важных

задач.

4.10.1. Центрирование элемента

При разработке Web-страниц нередко требуется расположить элемент фиксированных размеров в центре клиентской области окна браузера или же в центре какого-нибудь элемента-контейнера. При изменении размеров окна браузера элемент должен сохранять свои заданные размеры, но центрироваться по горизонтали и/или вертикали. Для этой цели в HTML предусмотрен контейнерный тег `<center>`. Все элементы в нем центрируются по ширине окна браузера или содержащего контейнера. Однако сейчас его не рекомендуется использовать и для центрирования предлагается применять надлежащие параметры CSS. Рассмотрим два варианта решения этой задачи на примере центрирования элемента `<div>`.

Первый вариант основан на задании значения `auto` параметра `margin`, определяющего ширину прозрачных полей. На рис. 4.19 показан пример, когда элемент `<div>` с фиксированными размерами центрируется по горизонтали относительно окна браузера. Обратите внимание, эффект центрирования происходит благодаря параметрам `margin-left:auto` и `margin-right:auto` при условии, что параметры `width` и `height` для центрируемого элемента заданы. Однако применение параметров `margin-top:auto` и `margin-bottom:auto` не приведет к центрированию по вертикали.

Второй вариант позволяет центрировать элемент как по горизонтали, так и по вертикали. Как известно, положение элемента при абсолютном позиционировании (`position:absolute`) определяется координатами `left` и `top`, отсчитываемыми от края окна браузера или внутреннего края границы контейнера до внешнего края прозрачного поля (`margin`) элемента (*см. разд. 4.5*). Если ширина поля отлична от нуля, то видимое содержимое элемента располагается по координатам `left + margin-left` и `top + margin-top`. Однако ширина поля может быть задана и отрицательным числом. Если задать `left:50%`, а величину `margin-left` указать равной половине ширины элемента со знаком минус, то последний будет центрироваться по горизонтали. Аналогично, если задать `top:50%`, а величину `margin-top` указать равной минус половине высоты, то элемент будет центрироваться по вертикали. Однако при определенном соотношении размеров окна браузера и элемента края последнего могут оказаться за границами окна. На рис. 4.20 показан пример центрирования элемента `<div>` относительно окна одновременно и по горизонтали, и по вертикали.

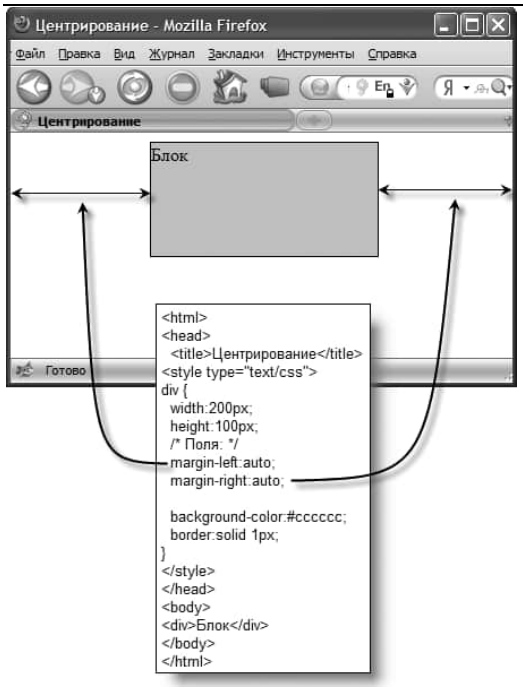

Рис. 4.19. Центрирование элемента по горизонтали относительно окна браузера

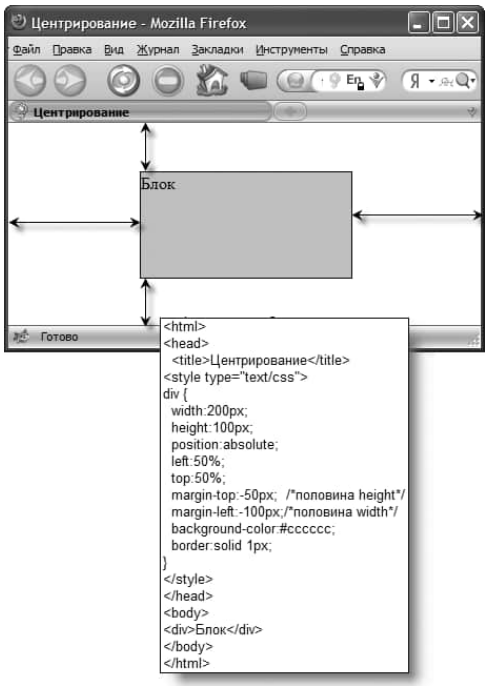

Рис. 4.20. Центрирование элемента относительно окна браузера одновременно по горизонтали и по вертикали

4.10.2. Управление положением элемента с помощью мыши

Динамически изменять положение элемента и другие его параметры CSS при наведении указателя мыши можно посредством псевдокласса `:hover` (*см. разд. 3.2*).

На рис. 4.21 показан пример, в котором изображения игральных карт абсолютно позиционированы так, чтобы они частично перекрывали друг друга, располагаясь в "стопке". Это исходное положение карт. При наведении указателя мыши на какую-либо карту последняя принимает параметры CSS, указанные в правиле с селектором `img:hover`. В данном случае это правило задает достаточно большое значение параметра `z-index`, определяющее номер слоя (см. *разд. 4.6*). В результате изображение будет выведено на передний план, поверх остальных. Кроме изменения слоя в рассмотренном примере меняется ширина границы карты, на которой находится указатель мыши. При желании можно в правиле `img:hover` установить новые значения для ширины и высоты самой карты. С уходом указателя мыши данная карта принимает свои исходные параметры.

Теперь рассмотрим случай, когда требуется переместить картинку, на которую наведен указатель мыши. Казалось бы, для этого достаточно использовать правило `img:hover {top:значение; left:значение}`. Однако такое решение, как ни странно, не приведет к цели. Чтобы перемещение все-таки было возможно, необходимо назначить псевдокласс `:hover` индивидуально для каждой картинки, т. е. `#img1:hover{…}`, `#img1:hover{…}` и т. д. Однако при этом возникает нестабильная ситуация: при наведении указателя мыши картинка перемещается на новое место, а указатель остается на прежнем месте и при малейшем движении указателя картинка возвращается в исходное состояние. Поэтому для перемещения элементов лучше использовать соответствующий скрипт, а не псевдокласс `:hover`.

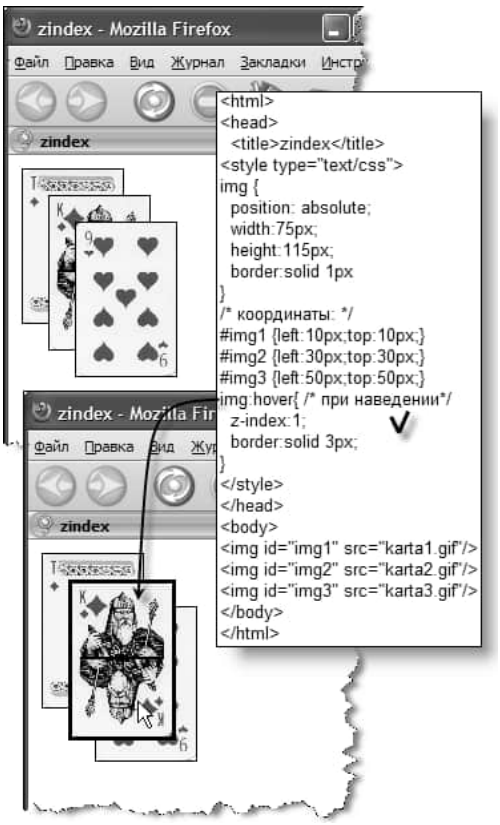

Рис. 4.21. Выведение картинки на передний план при наведении на нее указателя мыши

4.10.3. Раскрывающаяся панель

Раскрывающаяся панель — это прямоугольник, наведение указателя мыши на который приводит к увеличению его размеров и, как следствие, к видимости некоторого содержимого, например текста, ссылок и т. п. Уход указателя за границы данного прямоугольника приводит к сворачиванию последнего до исходных размеров. Очевидно, раскрывающееся меню является вариантом раскрывающейся панели или совокупности таких панелей.

Создать такую панель можно на основе элемента `<div>`, используя правило CSS `div:hover`, устанавливающее ее параметры при наведении указателя мыши. При этом необходимо решить вопрос о том, как раскрытие панели повлияет на расположение других элементов документа. Возможно, что они должны потесниться, уступая место увеличившемуся элементу, а возможно все они останутся в исходном положении, но тогда раскрытая панель должна расположиться на переднем плане (в верхнем слое). Первый вариант характерен для раскрывающихся списков, а второй — для всплывающих подсказок и раскрывающихся меню. Остановимся на втором варианте, т. е. на панели, которая при раскрытии не сдвигает остальные элементы, а занимает верхний слой.

На рис. 4.22 приведен пример раскрывающейся панели с некоторым текстом. Для расположения панели на переднем плане применено абсолютное позиционирование с достаточно большим значением параметра `z-index`. В исходном состоянии размеры панели невелики (`width:150px`; `height:20px`) и выходящее за ее пределы содержимое не видно (`overflow:hidden`). При наведении на панель указателя мыши действует правило, устанавливающее новое значение ее высоты (`height:100px`). Вообще говоря, значение `height` подбирают экспериментально так, чтобы все содержимое панели в ее раскрытом состоянии было видно. Впрочем, при необходимости можно добавить полосы прокрутки, для чего в правиле с селектором `div.pane:hover` достаточно указать параметр `overflow:scroll` или `overflow:auto`.

Развитие данной темы применительно к созданию раскрывающихся меню можно найти в *разд. 6.1*.

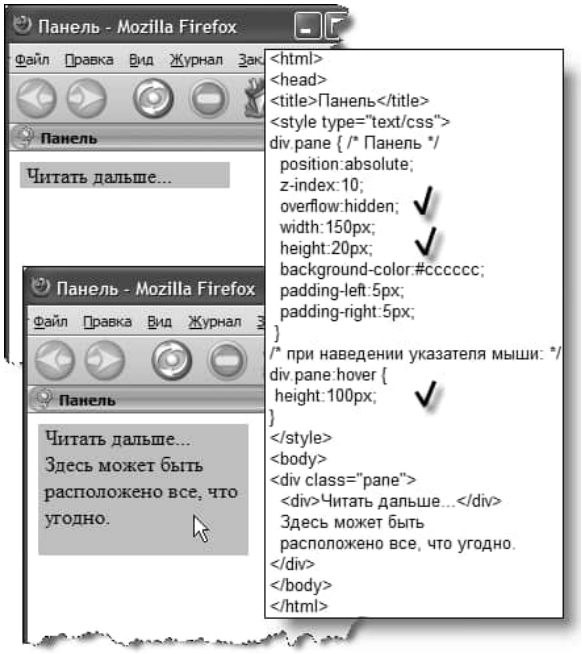

Рис. 4.22. Раскрывающаяся панель

Глава 5

Фон элементов и границ

Фон страницы в целом, а также ее элементов задают, указывая цвет или графическое изображение из внешнего файла. Для этого существует целый ряд параметров CSS 2 и 3. Группа параметров `background` позволяет задать фон элементов, а `border` — границы. Некоторые из параметров группы `border` (`border-color`, `border-style` и др.) были описаны в *разд. 3.5*. Здесь мы рассмотрим еще два параметра (`border-image` и `border-radius`), применение которых позволяет существенно расширить возможности по оформлению внешнего вида элементов.

5.1. *background*

Параметры для задания фона элементов:

- `background-color` — цвет фона; принимает значения:
  - *`значение цвета`* (*см. разд. 3.4*);
  - `transparent` (прозрачный).

  Примеры:

  ```
  body {background-color: #808080}
  p {background-color: yellow}
  div {background-color: rgb(255,128,0)}
  ```

- `background-image` — графическое изображение в качестве фона; принимает значения:
  - `url("`*`URL файла изображения`*`");`
  - `none` (изображение отсутствует).

  Если заданы цвет фона и изображение, то последнее отображается сверху цветового фона, если оно доступно. Фоновое графическое изображение применяется в своих исходных (оригинальных) размерах и не масштабируется. Если оно не помещается в контейнер, для которого служит фоном, то соответствующим образом усекается. В этом случае по умолчанию отображается верхняя левая часть картинки, но можно указать и иную ее область с помощью параметра `background-position` (*см. далее*).

  Пример:

  ```
  body {background-image: url("/images/mypicture.jpg")}
  ```

- `background-repeat` — определяет, будет ли фоновое изображение дублироваться, чтобы покрыть всю страницу или элемент; принимает значения:
  - `repeat` — дублируется по горизонтали и вертикали (принято по умолчанию);
  - `repeat-x` — дублируется по горизонтали;
  - `repeat-y` — дублируется по вертикали;
  - `no-repeat` — не дублируется (используется единственная копия).

  Пример:

```
body {
  background-image: url("/images/mypicture.jpg");
  background-repeat:repeat-y
}
```

- background-attachment — определяет способ фиксации фонового изображения; принимает значения:
  - fixed — фоновое изображение фиксируется относительно окна просмотра;
  - scroll — фоновое изображение фиксируется относительно документа и перемещается вместе с ним при прокрутке;
- background-position — определяет начальное положение фонового изображения с помощью двух значений (горизонтальной и вертикальной координат), разделенных пробелом. Значения задаются в абсолютных величинах длины (например, в пикселах), процентах или посредством ключевых слов left, top, right, bottom, center. Значения по умолчанию: left, top. Если указаны координаты *x*, *y*, то точка изображения, смещенная относительно его левого верхнего угла на *x* вправо по горизонтали и на *y* вниз по вертикали, помещается в точку, смещенную вдоль области отступов, на *x* вправо по горизонтали и на *y* вниз по вертикали. В частности, пара координат 0% 0% означает выравнивание левого верхнего угла изображения относительно левого верхнего угла краевой линии отступов, а 100% 100% — выравнивание правого нижнего угла изображения относительно правого нижнего угла краевой линии отступов. Если указана лишь одна координата, то считается, что обе координаты равны.

  Пример:

  ```
  body { background-image: url("/images/mypicture.jpg");
         background-position: 50% 25%}
  ```

- background — позволяет задать значения сразу нескольких свойств, перечисленных ранее (так называемая сокращенная запись). Значения всех или только некоторых частных свойств (background-color, background-image и др.) указывают через пробел в произвольном порядке.

  Примеры:

  ```
  body {background: blue url("/images/mypicture.jpg") repeat-y}
  body {background: url("/images/mypicture.jpg") no-repeat}
  ```

Параметр background имеет еще специальные значения, позволяющие залить градиентными цветами блочные элементы (например, , <body>). В CSS 3 предусмотрены линейный и радиальный градиенты, задаваемые значениями linear-gradient(*параметры*) и radial-gradient(*параметры*). Более подробно они будут рассмотрены в *разд. 5.6*.

В листинге 5.1 приведен код для демонстрации применения параметров фона для <body> и двух , а на рис. 5.1 — вид соответствующей страницы в окне браузера. Здесь фон страницы (<body>) полностью "замощен" копиями графического изображения некоторой затейливой линии. Два блока внутри <body> окрашены синим цветом и, кроме того, заполнены изображением автомобиля, причем по-разному. В первом блоке фоновая картинка дублируется по вертикали и, поскольку параметр background-position для нее явно не задан, ее первая копия выравнивается по верхнему левому углу контейнера. Во втором блоке фоновая картинка позиционируется не в верхнем левом углу и, кроме того, ее первая копия видна лишь частично. Это объясняется выбором значений параметра background-position:25% 70%.

Листинг 5.1. Применение параметров фона

```
<!DOCTYPE html>
<html>
<head>
<title>Фон</title>
<style type="text/css">
     body {
          margin:0;
          padding-left:10px;
          padding-right:10px;
          color:white;
       /* картинка в качестве фона всего документа: */
          background: url("line.jpg") repeat
     }
     div { /* общие параметры для всех div */
   position:absolute;
          height:150px;
          width:250px;
          padding:10px;
          text-align:right; /* выравнивание по правому краю */
      /* синий фон с картинкой: */
          background: blue url("auto.gif")
     }
     #div1 {
          top:30px;
          left:30px;
          /* дублирование фона по вертикали: */
          background-repeat:repeat-y
     }
     #div2 {
          position:absolute;
          top:220px;
          left:30px;
         /* дублирование фона по горизонтали и
            задание начальной позиции: */
          background-repeat:repeat-x;
          background-position:25% 70%
     }
</style>
</head>
<body>
<div id="div1">
     Это элемент div1
</div>
<div id="div2">
     Это элемент div2
</div>
```

```
</body>
</html>
```

Нередко в качестве фона страницы выбирают так называемую градиентную заливку: цвет постепенно переходит от более темных к светлым оттенкам, или наоборот. Для достижения данного эффекта в графическом редакторе (например, Adobe Photoshop или GIMP) создают длинную и узкую (шириной от одного до нескольких пикселов) полоску с градиентной заливкой и сохраняют ее в файле. Полосу необходимо сделать как можно у́же, чтобы объем файла с ее изображением был минимальным. Затем эту полоску дублируют по вертикали или горизонтали. Вместе с тем, для создания градиентной заливки можно использовать параметр `background` со значениями `linear-gradient()` и `radial-gradient()`, определяющими линейный или радиальный градиент (*разд. 5.6*). Бордюры создают аналогично дублированием небольшого изображения в одном из направлений.

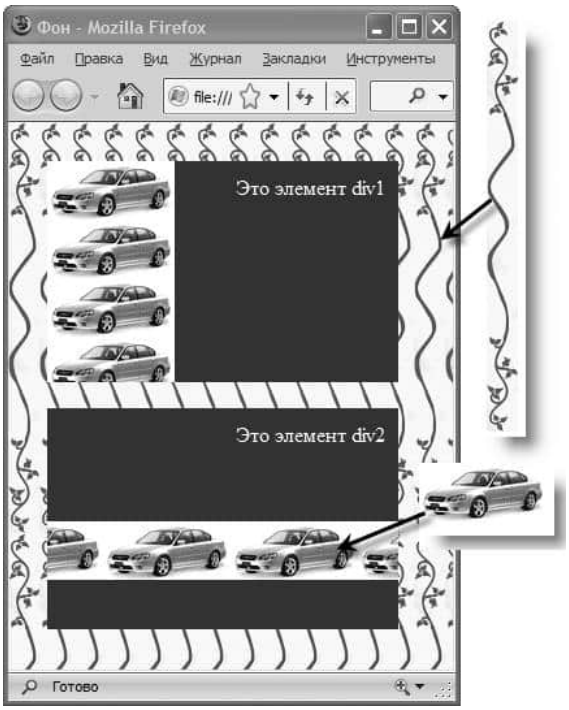

Рис. 5.1. Вид кода листинга 5.1

Графическое изображение в качестве фонового можно поместить в единственном экземпляре (`background-repeat:no-repeat`) в любом месте страницы (`background-position`). Нередко графический дизайн всей страницы выполняют в каком-либо графическом редакторе, а затем полученное изображение используют в качестве фонового. Однако при этом не следует забывать, что большому графическому изображению, покрывающему в качестве фона всю страницу, соответствует немалый объем файла. Как известно, объем файла и время его загрузки в браузер находятся в прямой зависимости.

5.2. *opacity*

В CSS 3 для задания прозрачности предлагается применять параметр `opacity` со значениями от нуля (прозрачный) до единицы (непрозрачный) и значение `rgba(r,g,b,a)` для параметра `background` или `background-color`. В отличие от обычного значения вида `rgb(r, g, b)` значение вида `rgba(r,g,b,a)` позволяет кроме уровней *r*, *g*, *b* яркости красной, зеленой и синей составляющих цвета задать еще и уровень прозрачности *a* (так называемый альфа-канал) со значениями от нуля (прозрачный) до единицы (непрозрачный). Параметр `opacity` назначает уровень прозрачности всего элемента, к которому применяется (например, графического изображения, элемента `<div>` и др.), а значение вида `rgba(r, g, b, a)` устанавливает лишь прозрачность цвета фона.

В листинге 5.2 в качестве примера приведен код документа с тремя графическими изображениями, которые частично перекрываются. Для всех картинок установлен уровень прозрачности 0,5. На рис. 5.2 показан вид в браузере документа с тремя изображениями игральных карт, когда прозрачность равна 1 и когда она равна 0,5.

Листинг 5.2. Установка уровня прозрачности

```
<!DOCTYPE html>
<html>
<head>
<title>Прозрачность</title>
<style type="text/css">
  img {
   position: absolute;
   width:150px;
   height:225px;
   border:solid 1px;
   opacity:0.5 /* прозрачность */
  }
/* координаты */
  #img1 {left:10px;top:10px;}
  #img2 {left:30px;top:30px;}
  #img3 {left:50px;top:50px;}
</style>
</head>
<body>
<img id="img1" src="karta1.gif"/>
<img id="img2" src="karta2.gif"/>
<img id="img3" src="karta3.gif"/>
</body>
</html>
```

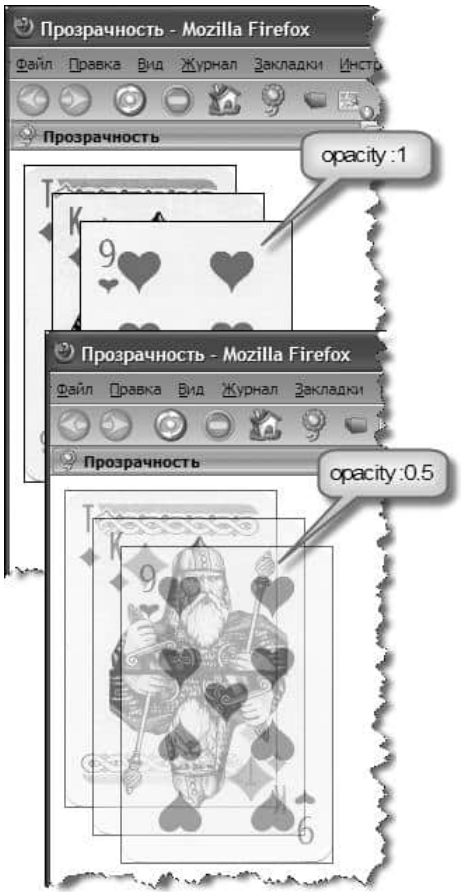

Рис. 5.2. Отображение в браузере документа листинга 5.2

На рис. 5.3 показан пример, в котором элемент `<div>` с параметром прозрачности 0,6 черного цвета расположен поверх графического изображения. Если цвет фона для `<div>` не указать, то эффект полупрозрачности не получится.

Если контейнер `<div>` содержит другие элементы (например, картинки, текст и т. д.) и для него задан уровень прозрачности, то последний будет действовать на все содержимое данного контейнера. С другой стороны, полупрозрачные элементы внутри контейнера останутся полупрозрачными, если контейнер сделать непрозрачным.

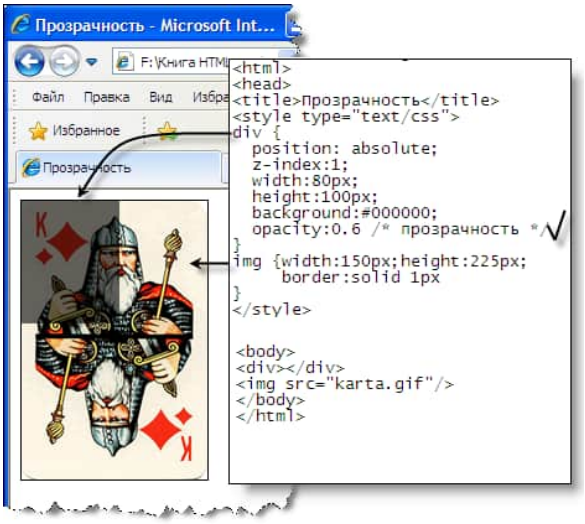

Рис. 5.3. Полупрозрачный элемент `<div>` расположен поверх картинки

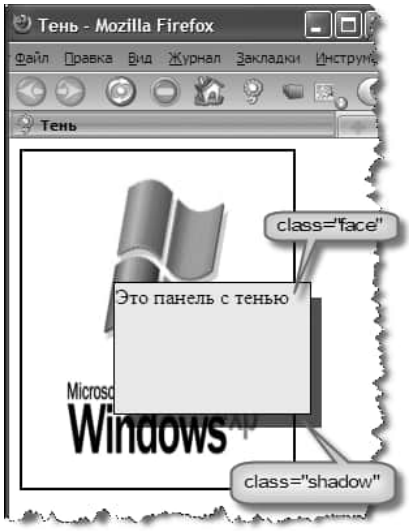

Рис. 5.4. Отображение в браузере документа листинга 5.3

Одно из типичных применений свойства прозрачности — создание полупрозрачной тени, отбрасываемой элементом, например `<div>` или `<img>`. В листинге 5.3 приведен пример, в котором панель отображается с тенью на фоне картинки (рис. 5.4). Такую панель создают посредством трех элементов `<div>`. Первый из них — контейнер, включающий в себя остальные два, из которых один представляет лицевую сторону (face) панели, а другой — полупрозрачный прямоугольник, изображающий тень (shadow) от панели. Лицевая сторона и ее тень — равновеликие прямоугольники, но "теневой" прямоугольник сдвинут относительно лицевого вправо и вниз (будто бы источник света находится в верхнем левом углу) благодаря положительному значению координаты `left`, отрицательному значению координаты `top` и параметру `position:relative`. В данном примере ширина тени равна 10 px, хотя в типичных дизайнерских решениях ее задают в пределах 2—5 px. Общий контейнер (`<div class="pane">`) позиционирован абсолютно (`position:absolute`) и его расположение на экране задают параметры `left` и `top`. При этом элементы его содержимого (`<div class="face">` и `<div class="shadow">`) позиционируются на экране, сохраняя заданное положение относительно друг друга. Обратите внимание, что лицевая сторона панели имеет большее значение параметра `z-index`, чем теневая. Общему контейнеру (`<div class="pane">`) назначены нулевые размеры.

Листинг 5.3. Панель с тенью

```
<!DOCTYPE html>
<html>
<head>
<title>Тень</title>
<style type="text/css">
pane{ /* панель */
   position:absolute;
   top:110px;
   left:80px;
   width:0;
   height:0
}
face { /* лицевая сторона */
  position: relative;
  z-index:1;
  width:150px;
  height:100px;
  background:#eeeeee;
  border:solid 1px;
}
.shadow { /* тень */
  position: relative;
  z-index:0;
  left:10px;/* смещение вправо */
  top:-90px; /* смещение вверх */
  width:150px;
  height:100px;
  background:#000000;
  opacity:0.5 /* прозрачность */
}
</style>
</head>
<body>
<img src="winlogo.jpg" style="border:solid 2px"/>
<div class="pane">
  <div class="face">Это панель с тенью</div>
  <div class="shadow"></div>
</div>
</body>
</html>
```

Развитие данной темы применительно к созданию меню рассматривается в *разд. 6.1*. Для реализации эффекта тени в CSS 3 имеется специальный параметр `box-shadow` (*см. разд. 5.5*).

5.3. *border-image*

В CSS 3 предусмотрен параметр `border-image`, с помощью которого можно заполнить

графическим изображением границу элемента. Здесь мы рассмотрим лишь основные значения данного параметра в следующем формате:

`border-image: url("URL файла изображения") ломтики повторение;`

Первая часть значения параметра `border-image` задает графическое изображение, которое будет заполнять границы элемента.

Вторая часть определяет девять фрагментов (ломтиков, slice) исходного изображения для заполнения четырех углов и четырех сторон границы элемента, а также окаймляемой ею центральной области. Таким образом, заполняется не только граница (ее размеры задаются параметром `border-width`), но и создается графический фон самого элемента.

Третья часть определяет, как графические изображения ломтиков будут заполнять стороны границы и центральную область. Для отмены заполнения границы изображением служит значение `none`.

Ломтики задаются указанием от одного до четырех чисел, определяющих отступы линий разреза от верхнего, правого, нижнего и левого краев исходного графического изображения. Для задания относительных величин отступов указывается символ %. В противном случае (без единиц измерения) подразумеваются пикселы. Например,

`border-image: url("myimage.png") 10% 20 15 25;`

На рис. 5.5 показана схема деления исходного изображения на девять ломтиков с помощью четырех чисел n1, n2, n3 и n4.

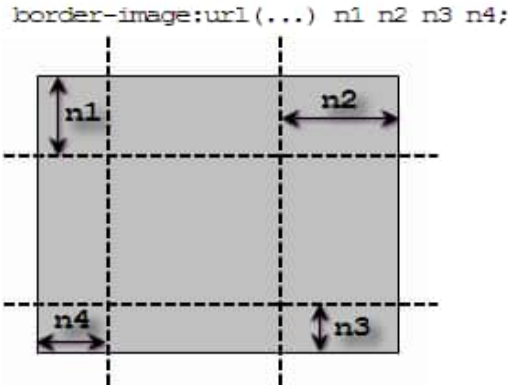

Рис. 5.5. Разрезка графического изображения на ломтики

Если четвертое число не указано, то его значение устанавливается равным второму; если третье число также отсутствует, то его значение равно первому; если и второе число не задано, то его значение равно первому. Отрицательные значения недопустимы, а значения, превышающие соответствующий размер изображения, интерпретируются как 100%.

Ломтики могут перекрываться. Так, если n2 + n4 больше или равно ширине изображения, то ломтики для верхней и нижней сторон границы элемента, а также центральная область окажутся пустыми. Аналогично, пустыми будут ломтики для левой, правой сторон и центральной области, если n1 + n3 окажется больше либо равным высоте изображения.

Ломтики могут по-разному заполнять стороны границы и центральной области в зависимости от третьей части значения параметра `border-image` (*повторение*), которое может принимать в качестве значения одно или два из ключевых слов: `stretch` (по умолчанию), `round` и `repeat`. Если второе слово отсутствует, то подразумевается, что оно совпадает с первым. Если оба слова опущены, то предполагается `stretch`. На рис. 5.6 показано, как действуют данные значения на

заполнение сторон границы и центральной области. Исходное изображение разрезано на девять примерно одинаковых по размерам ломтиков. Если значение *повторение* опущено или задано `stretch`, то ломтики будут масштабированы так, чтобы полностью заполнить соответствующий участок границы и центральной области. При значении `round` ломтики масштабируются так, чтобы в заполняемой области поместилось целое их количество. При значении `repeat` ломтики тоже масштабируются и повторяются, но без учета требования их целочисленности. Более сложные преобразования происходят при указании двух различных ключевых слов.

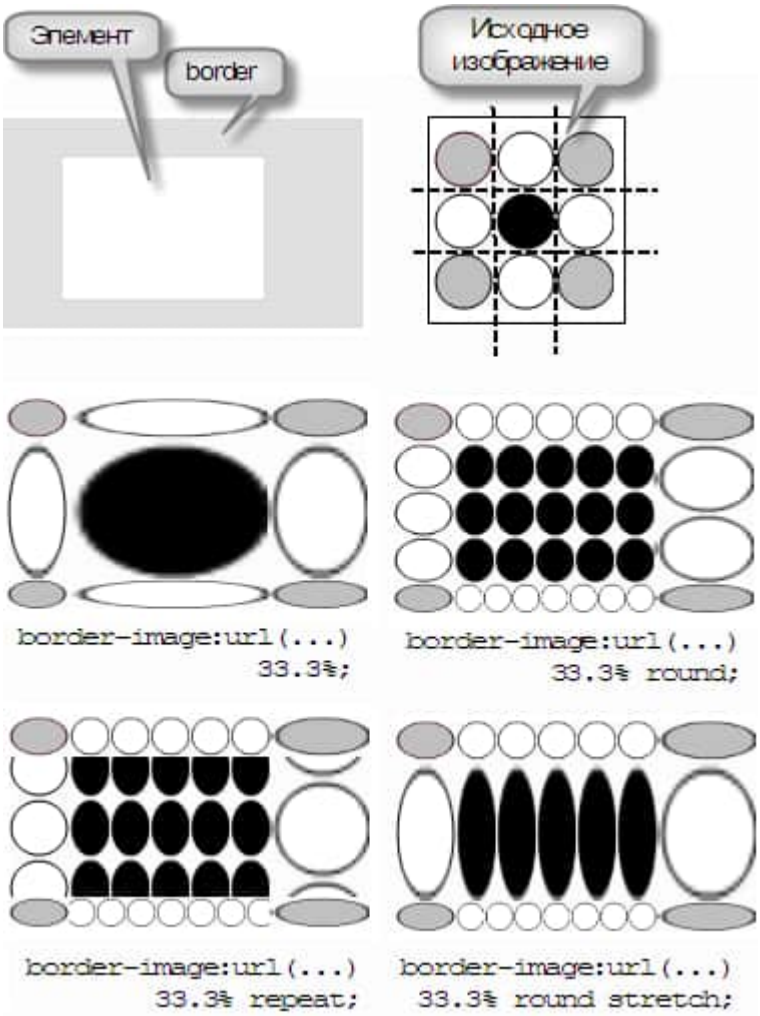

Рис. 5.6. Варианты заполнения границ и центральной области ломтиками изображения

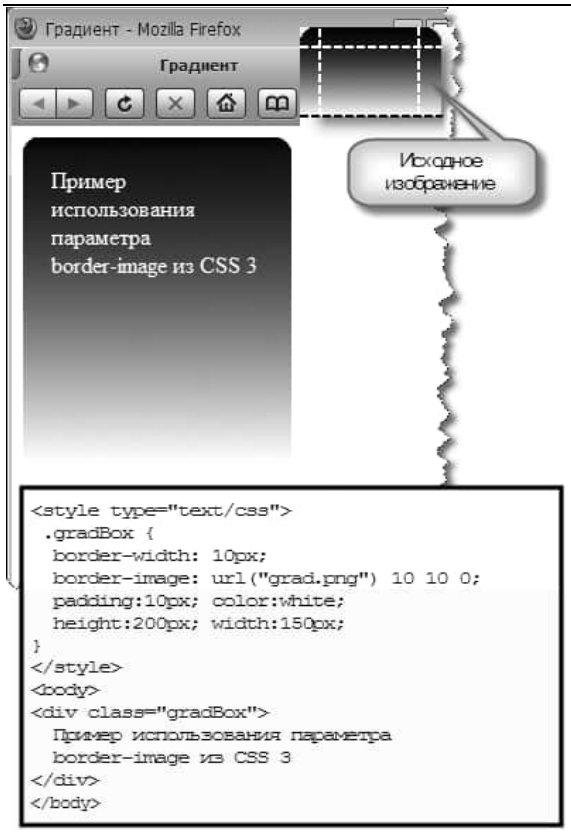

Рис. 5.7. Пример использования параметра `border-image`

На рис. 5.6 показаны элемент (например, `<div>`), исходное графическое изображение, разделенное на девять одинаковых по размеру ломтиков, и четыре варианта заполнения ими данного элемента.

Пример использования параметра `border-image` для создания панели с закругленными верхними углами и градиентным фоном приведен на рис. 5.7. Исходное изображение в данном случае разделяется на шесть ломтиков, поскольку отступ линии разреза от нижнего края равен нулю. Размеры границ элемента `<div>` и угловых ломтиков примерно одинаковы.

5.4. *border-radius*

Для создания границ элементов с закругленными углами в CSS 3 имеется параметр `border-radius`, задающий радиус закругления в единицах длины или процентах. Например, `border-radius: 5px` определяет радиус закругления всех углов равным 5 пикселам. В качестве значения можно задать радиусы верхнего левого, верхнего правого, нижнего правого и нижнего левого углов. Например, `border-radius: 5px 10px 10px 30px`. При этом закругления будут представляться дугой окружности. Возможно также задание закруглений по эллиптической дуге, определяемой двумя радиусами — горизонтальным и вертикальным. Наборы горизонтальных и вертикальных радиусов разделяются прямым слэшем, например

```
border-radius: 5px 10px 10px 30px/10px 20px 15px 5px
```

В частности, параметр `border-radius: 5px/10px` задает для всех углов границы элемента горизонтальный и вертикальный радиусы, равные соответственно 5 и 10 пикселам. Значение в процентах для горизонтального радиуса определяется относительно ширины элемента, а для вертикального — относительно высоты.

На рис. 5.8 показано несколько вариантов применения параметра `border-radius`.

Значения радиусов углов можно также определить и отдельно с помощью стилевых параметров `border-top-right-radius`, `border-bottom-right-radius`, `border-bottom-left-radius` и `border-top-right-radius`.

Определение радиуса закругления, приведенное в следующем примере, подойдет для всех основных браузеров:

```
.raundBox { border: 4px solid red;
            border-radius: 20px;
}
```

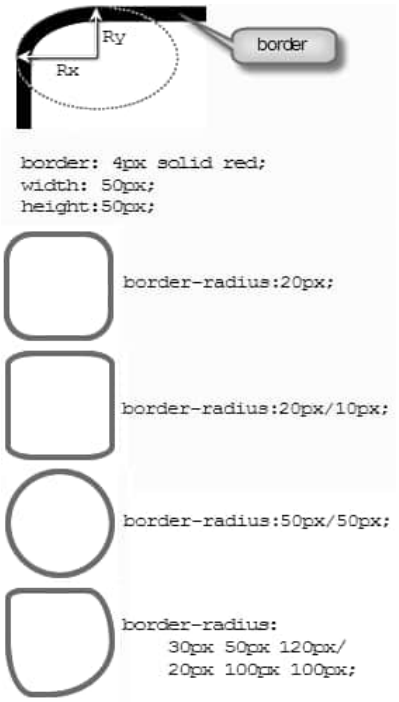

Рис. 5.8. Примеры использования параметра border-radius

5.5. *box-shadow*

В *разд. 5.2* рассматривалось применение параметра `opacity` для создания эффекта тени. Такого же эффекта можно добиться с помощью параметра CSS 3 `box-shadow`. Значением `box-shadow` является `none` для отмены задания тени или строка параметров тени:

- `inset` — если указано, то тень будет направлена внутрь элемента;
- *x* — смещение тени по горизонтали;
- *y* — смещение тени по вертикали;
- *r1*— радиус размытия тени (необязательный параметр);
- *r2* — растяжение тени (необязательный параметр);
- *цвет* — значение цвета тени.

Смещения и радиусы задаются числами с указанием размерности. Например

```
box-shadow: 5px 5px 10px #808080;
```

Определение тени из следующего примера подойдет для всех основных браузеров:

```
.shadow { box-shadow: 5px 5px 10px black;
}
```

Можно задать сразу несколько теней, для чего следует указать через запятую несколько соответствующих строк параметров.

5.6. Градиенты

Градиенты в качестве фона блочных элементов задаются специальными значениями (функциями) параметра `background`. В CSS 3 предусмотрены два типа градиентов — линейный и радиальный:

- `background:linear-gradient(`*параметры*`);`
- `background:radial-gradient(`*параметры*`).`

Те же значения можно применять и для параметра `background-image`.

Из параметров градиента следует указать, по крайней мере, два цвета (начальный и конечный), между которыми требуется выполнить плавный (градиентный) переход. Цвета задают в шестнадцатеричной, десятичной (через `rgb()` или `rgba()`) формах или именами. Кроме того, для линейного градиента можно указать направление, а для радиального градиента — позицию центра и тип (круговой или эллиптический).

> ***Примечание***
> Градиенты можно создать не только с помощью CSS, но и SVG (*см. разд.* 13.8), а также JavaScript (*см. разд.* 19.10.3).

5.6.1. Линейный градиент

Линейный градиент задается следующими через запятую параметрами для `linear-gradient()`:

- направление — задается как `to` *сторона* или *угол*, где *сторона* определяется одним или двумя ключевыми словами — `left`, `right`, `top`, `bottom`; а *угол* — угол в градусах или радианах, отсчитываемый от вертикальной оси по часовой стрелке; данный параметр не обязателен, по умолчанию направление градиента сверху вниз (`to bottom`, или `180deg`);
- цвета — два или более цвета, между которыми происходит плавный переход; рядом с цветом указывается размер области, в которой данный цвет не подвергается градиентному преобразованию (остается "твердым").

На рис. 5.9 показано несколько примеров линейного градиента. Изображения градиентов получены назначением стилевого параметра `background:linear-gradient(`*параметры*`)` элементу `<div>` с соответствующими размерами.

Рис. 5.9. Примеры линейного градиента

5.6.2. Радиальный градиент

Радиальный градиент задается следующими через запятую параметрами для `radial-gradient()`:

- тип и позиция центра — тип задается ключевым словом `ellipse` (эллиптический) или `circle` (круговой), это необязательный параметр (значение по умолчанию `ellipse`); позиция центра задается как `at` *x* `y` или `at` *x*, где *x*, *y* — ключевые слова `left`, `right`, `top`, `bottom`, `center`, или числа с указанием `%` или другой линейной размерности (например, `px`); значением по умолчанию является `at 50% 50%`.
- цвета — два или более цвета, между которыми происходит плавный переход; рядом с цветом указывается размер области, в которой данный цвет не подвергается градиентному преобразованию (остается "твердым").

На рис. 5.10 показано несколько примеров радиального градиента. Изображения градиентов получены назначением стилевого параметра `background:radial-gradient(`*параметры*`)`

элементу с соответствующими размерами. Разумеется, градиент можно применять и к другим элементам, например, к кнопке, задаваемой тегом <botton>.

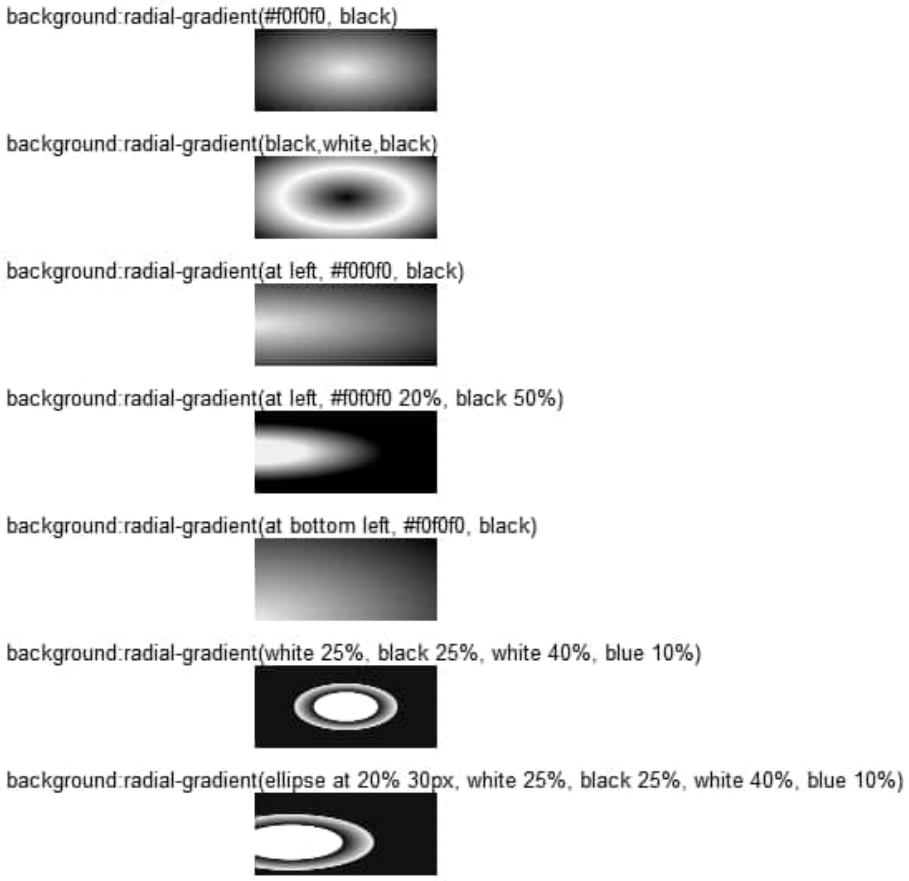

Рис. 5.10. Примеры радиального градиента

На рис. 5.11 показан пример создания круга, залитого радиальным элементом. В основание положен квадратный элемент , для границ которого установлен радиус, равный половине стороны квадрата. Для текста и элемента заданы параметры тени. При наведении указателя мыши параметры градиента изменяются (эффект подсветки).

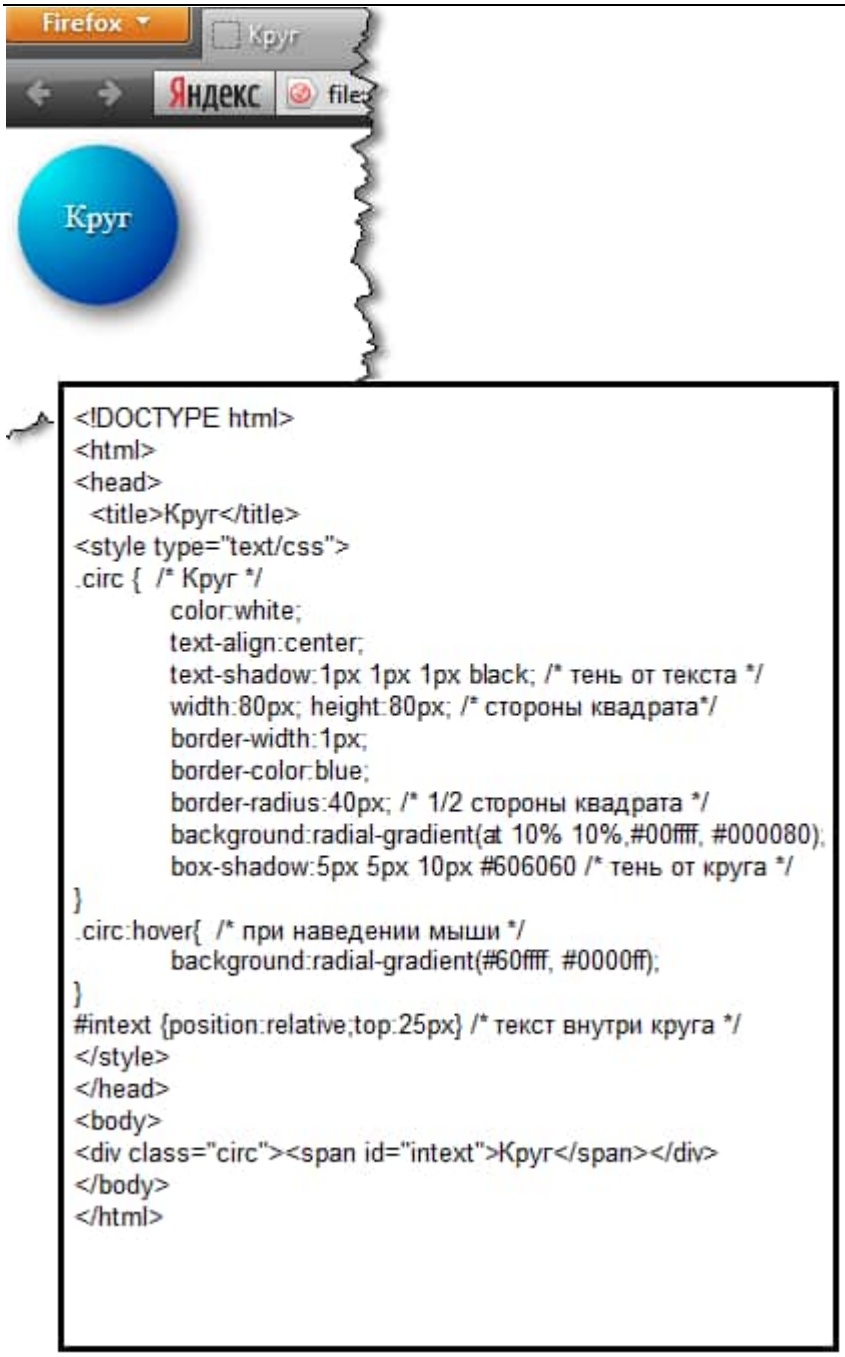

Рис. 5.11. Пример создания круга с тенью и радиальным градиентом в фоне

Глава 12

Трансформация и анимация с помощью CSS

В CSS 3 имеются параметры, посредством которых можно трансформировать (поворачивать, масштабировать, перемещать) видимые элементы страницы (блочные и внутристрочные), а также анимировать (плавно изменять во времени) их свойства.

12.1. Трансформация

Для трансформации видимых элементов страницы служит параметр `transform`, который может принимать следующие значения:

- `none` — отмена трансформации;
- `rotate(`*угол*`)` — поворот в плоскости экрана на заданный *угол*, положительные значения которого отсчитываются от вертикальной оси по часовой стрелке; поворот в плоскости экрана представляет собой поворот вокруг оси *z*, перпендикулярной этой плоскости;
- `scale(`k_x`, `k_y`)` — масштабирование по горизонтали и вертикали с коэффициентами *kx* и *ky* соответственно; значения коэффициентов больше 1 увеличивают, а значения коэффициентов меньше 1 — уменьшают размеры элемента; если указан только один коэффициент, то предполагается масштабирование с данным коэффициентом и по горизонтали, и по вертикали; отрицательные значения коэффициентов дополнительно приводят к отражению относительно горизонтальной и вертикальной осей;
- `scaleX(`k_x`)` — масштабирование по горизонтали с коэффициентом масштабирования k_x;
- `scaleY(`k_y`)` — масштабирование по вертикали с коэффициентом масштабирования k_y;
- `skewX(`*угол*`)` — наклон на `угол` по горизонтали;
- `skewY(`*угол*`)` — наклон на `угол` по вертикали;
- `translate(`*x*`, `*y*`)` — перемещение на расстояния `x` и `y` по горизонтали и вертикали соответственно;
- `translateX(`*x*`)` — перемещение на расстояние `x` по горизонтали;
- `translateY(`*y*`)` — перемещение на расстояние `y` по вертикали;
- `matrix(`a_{11}`,`a_{21}`,`a_{12}`,`a_{22}`,`a_{13}`,`a_{23}`)` — матрица преобразований точек элемента; позволяет выполнить одно или сразу несколько из перечисленных ранее преобразований.

При необходимости выполнить несколько трансформаций достаточно в качестве значения параметра `transform` указать через пробел несколько функций: `transform:`*функция(…) функция(…) … функция(…)*. Например, параметр `transform: rotate(20deg) scale(2)` предписывает поворот на 20 градусов и увеличение горизонтального и вертикального размеров в два раза. Перечисленные преобразования являются плоскими, при которых параллельные прямые остаются параллельными. Эффект перспективы в виде схождения исходно параллельных прямых с помощью данных преобразований получить нельзя.

На рис. 12.1 показаны несколько примеров трансформации элемента `<div>`, содержащего текст и графическое изображение. Соответствующий код приведен в листинге 12.1.

Отражение элемента по горизонтали и по вертикали получается при отрицательных коэффициентах масштабирования в функции `scale()`. Примеры показаны на рис. 12.2.

Смысл матричного преобразования, обеспечиваемого параметром `transform:` `matrix(`a_{11}`,`a_{21}`,`a_{12}`,`a_{22}`,`a_{13}`,`a_{23}`)`, такой же, как и в SVG, и подробно рассматривается в *разд. 13.11.6*.

Трансформация выполняется относительно точки, называемой центром трансформации. По умолчанию это геометрический центр элемента, но его можно установить по своему усмотрению с помощью параметра `transform-origin:`*x y z*, который принимает значения:

- *x* — координата по горизонтальной оси, указанная в значениях длины, процентах или словами `left`, `center`, `right`;
- *y* — координата по вертикальной оси, указанная в значениях длины, процентах или словами `top`, `center`, `bottom`;
- *z* — координата по оси, перпендикулярной плоскости экрана; указывается в значениях длины.

Центр трансформации в плоскости экрана по умолчанию соответствует `transform-origin:50% 50%` или `transform-origin:center center`. Установке центра трансформации в верхний левый угол соответствует `transform-origin:0 0`. В любом случае центр трансформации всегда находится внутри трансформируемого элемента. Разумеется, параметр `transform-origin` применяется совместно с `transform`.

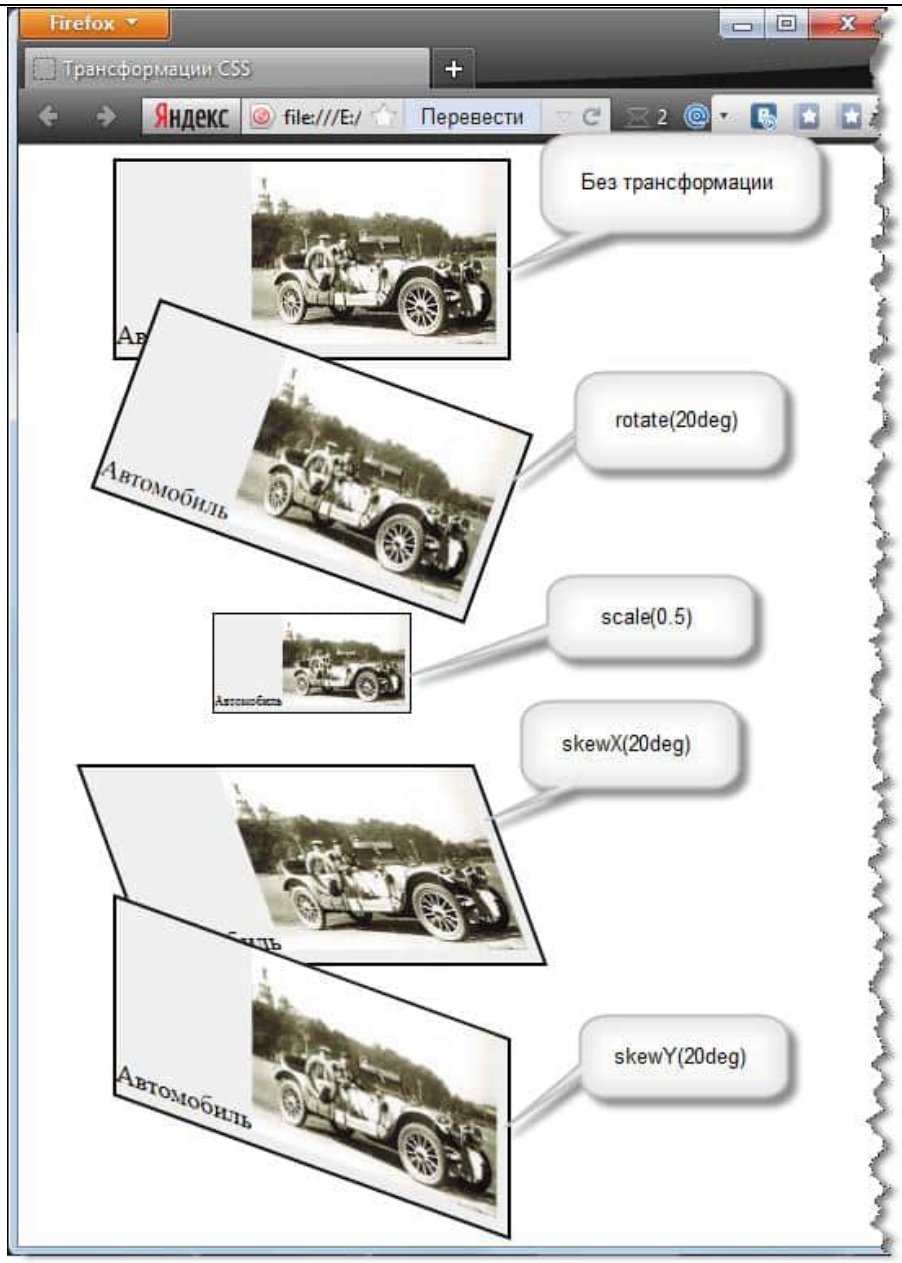

Рис. 12.1. Примеры трансформации

Листинг 12.1. Примеры трансформации

```
<DOCTYPE html>
<html>
<head>
<title>Трансформации CSS</title>
```

```
<style type="text/css">
        div {position:relative;left:50;width:240px;height:120px;
                background:#f0f0f0; border:solid 2px
        }
        .rotate {transform:rotate(20deg)}
        .scale {transform:scale(0.5)}
        .skewx {transform:skewX(20deg)}
        .skewy {transform:skewY(20deg)}
</style>
</head>
<body>
<div>
        Автомобиль<img src="mashina.jpg"/>
</div>
<div class="rotate">
        Автомобиль<img src="mashina.jpg"/>
</div>
<div class="scale">
        Автомобиль<img src="mashina.jpg"/>
</div>
<div class="skewx">
        Автомобиль<img src="mashina.jpg"/>
</div>
<div class="skewy">
        Автомобиль<img src="mashina.jpg"/>
</div>
</body>
</html>
```

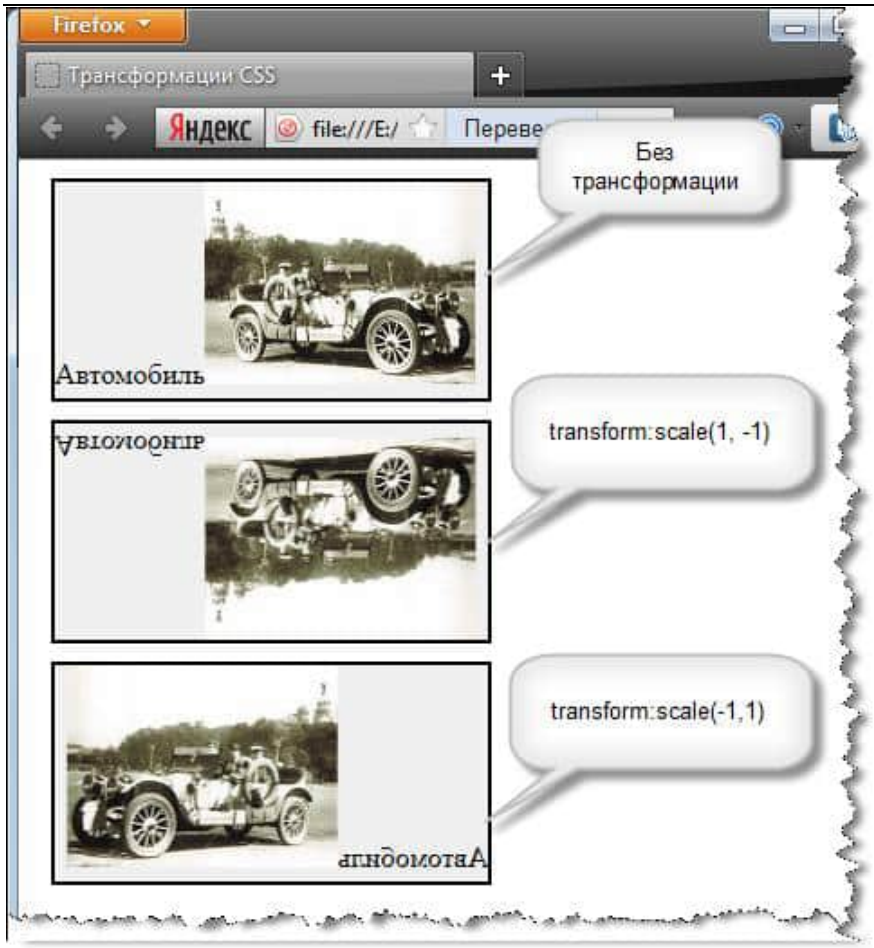

Рис. 12.2. Отражение элемента

> ***Примечание***
> Параметры `transform` и `transform-origin` поддерживаются браузерами Firefox и Internet Explorer 10+, для Chrome 32 и Opera 18 пока требуется добавлять префиксы соответственно `-webkit-` и `-o-`. Функцию `matrix()` пока поддерживает только Firefox при использовании префикса `-moz-`.

Рассмотренные преобразования являются плоскими (2d). В настоящее время ведется подготовка к имитации с помощью CSS трехмерных (3d) преобразований, для которых предназначен параметр `transform-style`, указывающий тип преобразования и имеющий следующие значения:

- `flat` — преобразование в плоскости экрана (по умолчанию), т. е. вокруг оси *z*, перпендикулярной плоскости экрана;
- `preserve-3d` — псевдо-трехмерное преобразование.

Параметр `transform-style` применяется совместно с `transform`. Если он не указывается явно, то имеется в виду его значение `flat`. Если указать `transform-style: preserve-3d`,

то возможны искажения элемента, имитирующие повороты вокруг горизонтальной (*x*) и вертикальной (*y*) осей посредством следующих значений параметра `transform`:

- `rotateX(`*угол*`)` — поворот вокруг горизонтальной оси;
- `rotateY(`*угол*`)` — поворот вокруг вертикальной оси.

На рис. 12.3 показаны примеры псевдо-трехмерного поворота графического изображения. Разумеется, это не полноценная имитация трехмерного поворота (собственно поворота не происходит), но для подготовки к нему данные преобразования пригодятся.

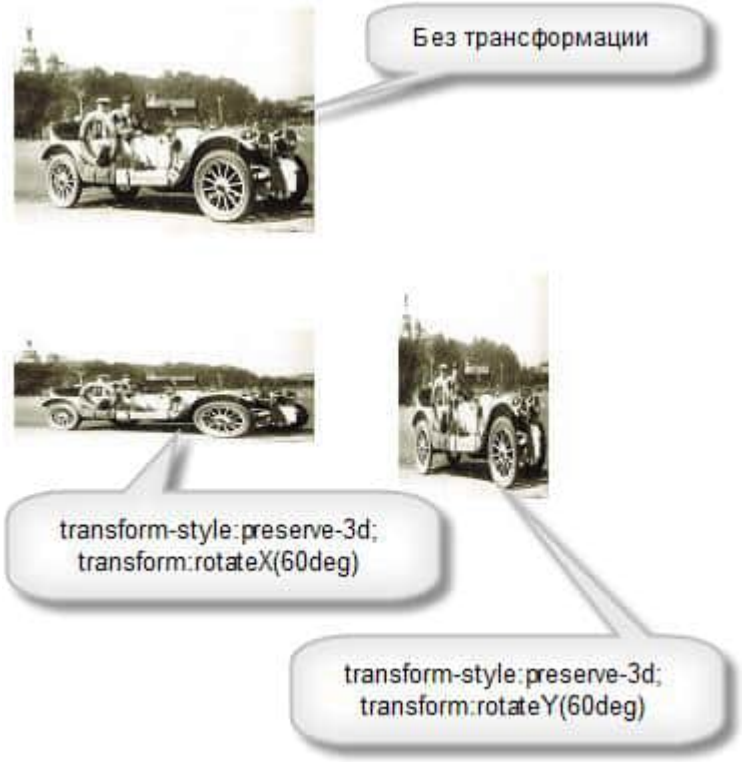

Рис. 12.3. Примеры псевдо-трехмерных поворотов

> ***Примечание***
> Параметры `transform-style`, `transform: rotateX()` и `transform:rotateY()` поддерживаются браузерами Firefox и Internet Explorer 11, для Chrome 32 пока требуется добавлять префикс `-webkit-`.

12.1. Анимация

Одному и тому же видимому элементу страницы можно назначить различные состояния (наборы стилевых параметров) и также установить параметры постепенного (плавного) перехода между ними во времени. В результате возникнет эффект анимации — плавное изменение цвета, формы, положения и других свойств элемента, заданных посредством стилевых параметров. С помощью средств анимации создают плавно раскрывающиеся панели, например, меню и виджеты.

Анимацию можно задать с помощью параметра `transition`, который принимает следующие значения:

- `none` — отмена анимации;
- строка параметров перехода, указанных через пробел: *`свойство длительность временная_функция задержка`*; здесь *`свойство`* — имя стилевого параметра элемента, который требуется изменить (анимировать), например, `transition: left 1s ease-out 0.5s` предписывает изменять стилевой параметр `left` в течение 1 с по закону, заданному функцией `ease-out`, с задержкой выполнения перехода 0,5 с; можно указать через запятую несколько строк параметров перехода, каждая для отдельного анимируемого свойства.

Переход можно определить и посредством отдельных стилевых параметров, перечисленных далее. Значения этих специальных параметров такие же, что указываются в интегральном параметре `transition` через пробел в одной строке.

Специальные параметры перехода:

- `transition-property` — имя стилевого параметра, который требуется анимировать; принимает следующие значения:
  - `none` — никакой стилевой параметр не задан;
  - `all` — все стилевые параметры предполагается анимировать;
  - *`свойства`* — имя или несколько имен анимируемых стилевых параметров, указанных через запятую;
- `transition-duration` — длительность перехода в единицах времени; значение по умолчанию — `0s` (переход осуществляется мгновенно, эффекта анимации нет); можно указать несколько значений через запятую, соответственно для свойств, перечисленных в значении параметра `transition-property`.
- `transition-timing-function` — временная функция, определяющая скорость перехода в различные моменты времени в интервале, заданном посредством параметра `transition-duration`; графики возможных вариантов данной функции показаны на рис. 12.4; параметр `transition-timing-function` принимает следующие значения:
  - `ease` — медленно в начале, ускорение до середины, а затем замедление; аналогично кривой Безье `cubic-bezier(0.25,0.1,0.25,1)`;
  - `ease-in` — медленно в начале с ускорением к концу; аналогично `cubic-bezier(0.42,0,1,1)`;
  - `ease-out` — быстро начинается и замедляется к концу; аналогично `cubic-bezier(0,0,0.58,1)`;
  - `ease-in-out` — начинается и заканчивается медленно; аналогично `cubic-bezier(0.42,0,0.58,1)`;
  - `linear` — постоянная скорость (значение по умолчанию);
  - `step-start` — моментальный переход в конечное состояние (эффекта анимации нет);

- `step-end` — весь заданный период (`transition-duration`) анимируемые стилевые параметры находятся в начальном состоянии, а затем моментально переходят в конечное состояние (эффекта анимации нет);
- `steps(`*`число, вид`*`)` — ступенчатая функция, принимающая два параметра; первый параметр — целое положительное число, задающее количество ступеней, а второй параметр задает вид ступенчатой функции: `start` — функция полунепрерывна снизу, `end` — функция полунепрерывна сверху;
- `cubic-bezier(`*`числовые параметры`*`)` — кривая Безье (*см. разд. 13.3, 19.10.2*).

❑ `transition-delay` — задержка (время ожидания) запуска анимации, указываемая в единицах времени; по умолчанию анимация начинается при инициации без задержки, такой же эффект достигается при значениях `0s` и `0ms`; при отрицательных значениях анимация запускается без задержки, но при этом может измениться ее начальный вид.

Графики временных функций перехода (значения параметра `transition-timing-function`), показанные на рис. 12.4, интерпретируются следующим образом. Горизонтальная ось координат соответствует времени, отнесенному к длительности перехода, а вертикальная — относительному пройденному "пути" перехода (относительной его части). Если переход происходит равномерно (с постоянной скоростью), то график временной функции изображается прямой линией (linear). Если график представлен кривой линией, то переход неравномерный.

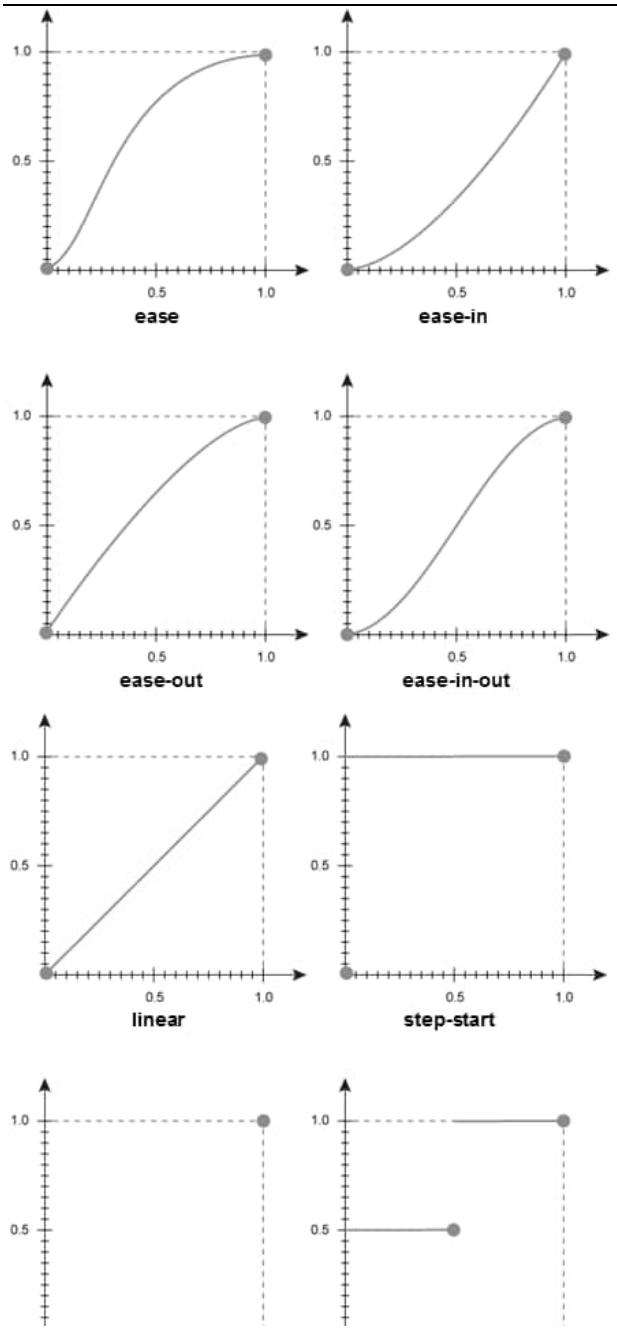

Рис. 12.4. Иллюстрация значений параметра `transition-timing-function`

Скорость перехода в данный момент времени определяется углом между касательной к кривой в этот момент и горизонтальной осью. Чем меньше угол (чем более полога кривая), тем меньше скорость, и наоборот, чем больше угол (чем круче кривая), тем больше скорость перехода. Так например, прямой, параллельной оси времени, соответствует скорость

перехода, равная нулю (мгновенный переход). С помощью функции Безье (`cubic-bezier(`*числовые параметры*`)`) можно задать произвольную кривую перехода, даже выходящую за предельные значения 0 и 1, но только внутри интервала [0, 1] относительного времени.

Теперь рассмотрим применение средств анимации. Прежде всего, для элемента, стилевые параметры которого требуется анимировать, следует задать два состояния — начальное и конечное, различающиеся значениями этих параметров. Затем среди стилевых параметров конечного состояния назначают параметры перехода (transition). Наконец, определяют, каким образом инициируется переход, т. е. собственно анимация.

Типичный способ инициации перехода — применение псевдоэлементов `:hover` или `:active`, для которых стилевые параметры вступают в силу, когда соответственно на элемент наводится указатель мыши или он переходит в активное состояние (*см. разд. 3.2.3*). Другой способ связан с применением скриптов на языке JavaScript для изменения стилевых параметров и здесь не рассматривается.

Допустим, требуется анимировать изменение цвета прямоугольника, заданного тегом `<div>`, которое должно произойти при наведении на него указателя мыши. Цвет определим посредством параметра `background`. Пусть в начальном (обычном) состоянии прямоугольник имеет следующие стилевые параметры:

```
div {width:150px; height:100px;
      background:red /* красный */
}
```

Таким образом, мы определили размеры и красный цвет элемента `<div>`. Конечное состояние прямоугольника, а заодно и способ инициации перехода, определим следующим образом:

```
div:hover {
    background:blue; /* синий */
    transition:background 1s ease
}
```

В конечном состоянии, переход к которому инициируется наведением указателя мыши на элемент `<div>`, задан синий цвет и, кроме того, указано (в значении `transition`), что переход к данному цвету должен произойти по правилам, перечисленным в значении параметра `transition`. А именно, переход должен произойти за 1 с, причем не равномерно, а медленно в начале и в конце и быстро в середине, точнее — в соответствии с временной функцией `ease`.

В рассмотренном примере вместо одного интегрального параметра `transition:background 1s ease`, задающего анимацию цвета фона `(background)` в течение 1 с согласно временной функции `ease`, можно использовать несколько частных параметров:

```
transition-property:background;  /* анимируемый параметр */
transition-duration:1s;          /* длительность перехода*/
transition-timing-function:ease  /* временная функция */
```

В рассмотренном случае, при наведении указателя мыши на прямоугольник его цвет плавно (но не равномерно) изменится от красного к синему, но как только указатель мыши покинет прямоугольник, его цвет вернется к начальному значению мгновенно, поскольку обратный переход не специфицирован. Чтобы возврат к начальному состоянию тоже был плавным, достаточно определить наборы стилевых параметров как в листинге 12.2.

Листинг 12.2. Стилевые параметры для плавного изменения цвета

```
div {width:150px; height:100px;
  background:red;
  transition:background 0.5s ease-in
}
div:hover {
  background:blue;
  transition:background 1s ease
}
```

В следующем примере (листинг 12.3) задается анимация сразу нескольких стилевых параметров: при наведении указателя мыши на элемент `<div>` плавно изменяются его размеры, цвета фона и содержащегося в нем текста.

Листинг 12.3. Анимация нескольких параметров

```
div {width:150px; height:100px;
      background:red;
      color:black;
 }
div:hover {
      width:300px; height:200px;
      background:blue;
      color:white;
      transition: background 1s ease, color 0.5s, width 1s, height 1s
}
```

Примечание

Анимацию можно применить и к стилевому параметру `transform`, однако в настоящее время это могут выполнить браузеры Firefox 26 и Internet Explorer 11. Chrome 32 и Opera 18 не анимируют `transform`, а для анимации других параметров перед `transition` пока требуется указывать фирменные префиксы `-webkit-` и `-o-` соответственно.

В листинге 12.4 приведен пример плавно раскрывающегося и закрывающегося вертикального меню, которое размечено посредством тегов `<div>` и `<span>`. Здесь анимируется стилевой параметр `height` (высота меню), при этом раскрытие и закрытие происходят равномерно, но с разными скоростями.

Листинг 12.4. Плавно раскрывающееся меню

```
<!DOCTYPE html>
<html>
<head>
<title>Меню плавное</title>
<style type="text/css">
.mainmenu { /* меню
        border:solid 1px #000000;
        position:absolute;
        width:170px;
```

```
        height:20px; /* высота такая, чтобы опции не помещались */
        background:#00aaff;
        overflow:hidden; /* все, что не помещается, не видно */
        text-align:center;
        color:white;
        z-index:100;
        font:bold 14px "Arial";
        transition:height 0.5s;  /* переход при уходе мыши */
        -webkit-transition:height 0.5s;
        -o-transition:height 0.5s;
}
.mainmenu:hover { /* при наведении мыши */
    height:100px;
        box-shadow:4px 4px rgba(100,100,100,0.5); /* тень */
        transition:height 1s; /* переход при наведении мыши */
        -webkit-transition:height 1s;
        -o-transition:height 1s;
}
.submenu { /* блок для опций меню */
        position:relative;
        top:10px;
        width:170px;
        height:100px;
        background:#00ddff;
}
.submenu span {display:block;}
.submenu span:hover {background:#00bbee}
.submenu a {
        font-size:14px;
        color:#000000;
        text-decoration:none;
}
/* ссылки для опций меню: */
.submenu a:visited {color:#aaaaaa}
.submenu a:hover{color:#ffff00}
.submenu a:active {color:#00ffff}
</style>
</head>
<body>
<!-- Разметка меню -->
<div class="mainmenu">
Меню
<div class="submenu">
        <span><a href="http://www.rambler.ru">Rambler</a></span>
        <span><a href="http://www.yandex.ru">Яндекс</a></span>
        <span><a href="http://www.google.ru">Google</a></span>
        <span><a href="http://ru.wikipedia.org/wiki/">Википедия</a></span>
```

```
</div>
</div>
</body>
</html>
```