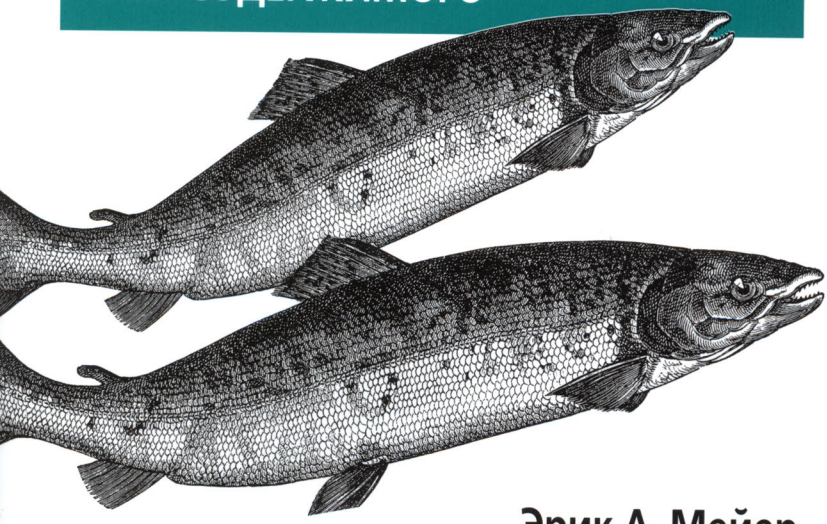
O'REILLY®

4-е издание
Рассмотрен CSS3

CSS
Карманный справочник

ВИЗУАЛЬНОЕ ПРЕДСТАВЛЕНИЕ ВЕБ-СОДЕРЖИМОГО



Эрик А. Мейер

ЧЕТВЕРТОЕ ИЗДАНИЕ

CSS

Карманный справочник

4TH EDITION

CSS

Pocket Reference

Eric A. Meyer

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

ЧЕТВЕРТОЕ ИЗДАНИЕ

CSS

Карманный справочник

Эрик А. Мейер

Москва · Санкт-Петербург · Киев
2016

ББК 32.973.26-018.2.75
М45
УДК 681.3.07

Издательский дом "Вильямс"

Зав. редакцией *С.Н. Тригуб*
Перевод с английского и редакция *И.В. Берштейна*

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Мейер, Эрик А.

М45 CSS. Карманный справочник, 4-е изд. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2016. — 288 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-2081-2 (рус.)

ББК 32.973.26-018.2.75

Научно-популярное издание

Эрик А. Мейер

CSS. Карманный справочник

4-е издание

Литературный редактор *Л.Н. Красножон*
Верстка *М.А. Удалов*
Художественный редактор *В.Г. Павлютин*
Корректор *Л.А. Гордиенко*

Подписано в печать 10.02.2016. Формат 84×108/32.
Гарнитура Times.
Усл. печ. л. 15,21. Уч.-изд. л. 7,6.
Тираж 500 экз. Заказ № 1340

Отпечатано способом ролевой струйной печати
в АО «Первая Образцовая типография»
Филиал «Чеховский Печатный Двор»
142300, Московская область, г. Чехов, ул. Полиграфистов, д.1

ООО "И. Д. Вильямс", 127055, г. Москва, ул. Лесная, д. 43, стр. 1

ISBN 978-5-8459-2081-2 (рус.)
ISBN 978-1-449-39903-0 (англ.)

Содержание

Предисловие

Каскадные таблицы стилей (Cascading Style Sheets — CSS) являются стандартом консорциума W3C на визуальное представление веб-страниц, хотя они могут быть использованы для оформления и других документов. После краткого введения в основные понятия CSS в этом карманном справочнике представлены в алфавитном порядке сначала селекторы, а затем свойства каскадных таблиц стилей по стандарту CSS3.

Условные обозначения, принятые в книге

В этой книге приняты следующие условные обозначения

- *Курсив* и **полужирный**. Служат для обозначения новых терминов и понятий в зависимости от контекста.
- `Моноширинный шрифт`. Служит для обозначения URL, адресов электронной почты, имен и расширений файлов, листингов и таких элементов программ, как переменные, имена функций, типы данных, операторы и ключевые слова.
- **`Моноширинный полужирный шрифт`**. Обозначает команды или другой текст, который должен быть введен пользователем буквально.
- *`Моноширинный наклонный шрифт`*. Обозначает текст, который должен быть заменен значениями, предоставляемыми пользователем или определяемыми по контексту.

Использование примеров кода

Эта книга служит справочным пособием, помогающим читателю решать стоящие перед ним задачи разработки прикладных программ. В общем, приведенные здесь примеры кода

можно использовать в своих программах и документации. Для этого не нужно спрашивать разрешения у автора или издателя. Так, для применения в прикладной программе нескольких фрагментов кода из примеров в этой книге специальное разрешение не требуется. Но для продажи или распространения в иных целях на CD-ROM фрагментов кода из примеров обязательно нужно получить разрешение от издательства O'Reilly. Для цитирования текста и примеров кода из этой книги в ответах на вопросы специальное разрешение не требуется. Но для внедрения значительной части примеров кода в документацию на собственную продукцию обязательно необходимо разрешение от издательства O'Reilly.

Ссылки на эту книгу как на первоисточник желательны, но не обязательны. В ссылке обычно указываются название книги, автор, издатель и ISBN, например *CSS Pocket Reference* by Eric A. Meyer (O'Reilly). Copyright 2011 Media, Inc., ISBN 978-1-449-39903-0.

Если читатель считает, что применение им примеров кода из этой книги выходит за рамки правомерного использования или упомянутых выше разрешений, он может связаться с издательством O'Reilly по адресу `permissions@oreilly.com`.

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо, либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое

мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг.

Наши электронные адреса:

E-mail: info@williamspublishing.com

WWW: http://www.williamspublishing.com

Наши почтовые адреса:

в России: 127055, г. Москва, ул. Лесная, д.43, стр. 1

в Украине: 03150, Киев, а/я 152

Основные понятия

Стилевое оформление HTML- и XHTML-документов

Стили можно применять к документам тремя разными способами, как поясняется в следующих разделах.

Встроенные стили

В разметке HTML- и XHTML-документов сведения о стилевом оформлении отдельного элемента могут быть указаны с помощью атрибута. В качестве значения атрибута указывается блок объявлений без фигурных скобок (см. приведенный далее раздел "Структура правил"), как показано ниже.

```
<p style="color: red; background: yellow;">Внимание!
Этот текст оформлен как предупреждение!</p>
```

Следует иметь в виду, что на момент написания этой книги таблицу стилей нельзя было полностью размещать в атрибуте `style`. В качестве значения этого атрибута можно было указывать лишь единственный блок объявлений. Например, в атрибуте `style` нельзя ни разместить стили наведения, используя объявление `:hover`, ни воспользоваться директивой `@import` в данном контексте.

Несмотря на то что атрибут `style` поддерживается в обычных языках разметки XML-документов (например, в XHTML 1.0, XHTML 1.1 и SVG), такая возможность вряд ли доступна во всех подобных языках. Вследствие этого использовать атрибут `style` обычно не рекомендуется, поскольку это считается ненадлежащей нормой практики авторской разработки веб-страниц и аналогичных документов.

Встроенные таблицы стилей

Таблицу стилей можно встроить в начале HTML- или XHTML-документа, используя элемент разметки `style`. Этот элемент должен быть указан в элементе разметки `head` следующим образом:

```
<html><head><title>Stylin'!</title>
<style type="text/css">
h1 {color: purple;}
p {font-size: smaller; color: gray;}
</style>
</head>
  ...
</html>
```

Аналогичная возможность в одних языках разметки XML-документов может быть предоставлена, а в других — нет. Поэтому для полной уверенности всегда обращайтесь за справкой к спецификации языка DTD (Определение типа документа).

Внешние таблицы стилей

Стили могут быть перечислены в отдельном файле. Главное преимущество отдельного файла состоит в том, что в нем можно собрать все наиболее употребительные стили, а для обновления всех страниц, на которых используется находящаяся в нем таблица стилей, достаточно отредактировать эту единственную таблицу. Еще одно преимущество заключается в том, что внешние таблицы стилей кешируются, что способствует эффективному использованию пропускной способности каналов связи. Обратиться к внешним таблицам стилей можно одним из трех способов, рассматриваемых ниже.

Директива `@import`

В начале любой таблицы стилей можно разместить одну или больше директив `@import`. Для стилевого оформления HTML- и XHTML-документов это обычно делается во встроенной таблице стилей следующим образом:

```
<head><title>My Document</title>
<style type="text/css">
@import url(site.css);
@import url(navbar.css);
@import url(footer.css);
body {background: yellow;}
</style>
</head>
```

Следует иметь в виду, что директивы `@import` могут располагаться в начале (а согласно спецификации — только в начале) любой таблицы стилей. Таким образом, в одну таблицу стилей можно импортировать другую таблицу, а в ту — третью таблицу.

Элемент разметки `link`

Элемент разметки `link` может быть использован в HTML- и XHTML-документах для связывания таблицы стилей с документом. С этой целью допускается указывать несколько элементов разметки `link`. Чтобы ограничить применение таблицы стилей одним или более носителями информации, можно указать атрибут `media` следующим образом:

```
<head>
<title>A Document</title>
<link rel="stylesheet" type="text/css" href="basic.css"
  media="all">
<link rel="stylesheet" type="text/css" href="web.css"
  media="screen">
<link rel="stylesheet" type="text/css" href="paper.css"
  media="print">
</head>
```

Имеется также возможность указать альтернативные таблицы стилей. Если предоставляются альтернативные таблицы стилей, то пользовательский агент (или автор документа) должен предоставить средства для выбора пользователем одного из вариантов таблиц стилей, как показано ниже.

```
<head>
<title>A Document</title>
<link rel="stylesheet" type="text/css" href="basic.css">
```

```
<link rel="alternate stylesheet" title="Classic"
  type="text/css" href="oldschool.css">
<link rel="alternate stylesheet" title="Futuristic"
  type="text/css" href="3000ad.css">
</head>
```

На момент написания данной книги большинство известных пользовательских агентов загружали все связанные таблицы стилей, включая альтернативные, независимо от того, реализовал ли их вообще пользователь. Это могло бы иметь отрицательные последствия для пропускной способности и нагрузки на сервер.

Инструкция обработки `xml-stylesheet`

В XML-документах (например, в XHTML-документах, отсылаемых с указанием типа MIME наподобие `"text/xml"`, `"application/xml"` или `"application/xhtml+xml"` в заголовке) можно воспользоваться инструкцией обработки `xml-stylesheet` для связывания таблицы стилей с документом. С этой целью любые инструкции обработки `xml-stylesheet` должны быть расположены в прологе XML-документа. Допускается указывать несколько подобных инструкций. А для того чтобы ограничить применение таблицы стилей одним или более носителями информации, можно указать псевдоатрибут `media` следующим образом:

```
<?xml-stylesheet type="text/css" href="basic.css"
  media="all"?>
<?xml-stylesheet type="text/css" href="web.css"
  media="screen"?>
<?xml-stylesheet type="text/css" href="paper.css"
  media="print"?>
```

С помощью инструкции обработки `xml-stylesheet` можно также связать альтернативные таблицы стилей, как показано ниже.

```
<?xml-stylesheet type="text/css" href="basic.css"?>
<?xml-stylesheet alternate="yes" title="Classic"
  type="text/css" href="oldschool.css"?>
```

```
<?xml-stylesheet alternate="yes" title="Futuristic"
  type="text/css" href="3000ad.css"?>
```

Заголовки HTTP Link

Последний (и наименее распространенный) способ связывания внешней таблицы стилей со страницами состоит в применении заголовка HTTP Link. С точки зрения CSS этот способ состоит в распространении действия элемента разметки `link` с помощью HTTP-заголовков.

Если ввести строку, подобную следующей, в файл конфигурации **`.htaccess`** на корневом уровне веб-сервера, то внешняя таблица стилей будет связана со всеми страницами на сайте:

```
Header add Link
   "</style.css>;rel=stylesheet;type=text/css;media=all"
```

В качестве альтернативы применению файла конфигурации **`.htaccess`**, что, как известно, может привести к снижению производительности, с той же самой целью можно отредактировать файл конфигурации **`httpd.conf`**, как показано ниже, где путь к каталогу `/usr/local/username/httpdocs` заменяется конкретным путем к начальному каталогу веб-сайта в формате UNIX.

```
<Directory /usr/local/username/httpdocs>
Header add Link
  "</ style.css>;rel=stylesheet;type=text/css;media=all"
</Directory>
```

На момент написания данной книги HTTP-заголовки поддерживались не всеми пользовательскими агентами, среди которых выделяются браузеры Internet Explorer и Safari. Следовательно, применение данного способа, как правило, ограничивается рабочими средами, основанными на других пользовательских агентах и лишь изредка на встроенных скрытых возможностях, подобных так называемым "пасхальным яйцам" для пользователей браузеров Firefox и Opera.

Структура правил

Таблица стилей состоит из одного или нескольких правил, описывающих порядок визуального представления элементов страниц. Каждое правило делится на следующие две основные части: *селектор* и *блок объявлений*. Структура правила приведена на рис. 1.1.

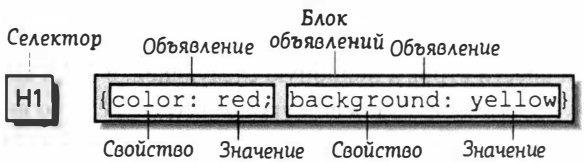


Рис. 1.1. *Структура правила*

В левой части правила находится селектор, предназначенный для выбора тех частей документа, к которым должно быть применено данное правило. А в правой части правила находится блок объявлений, состоящий из одного или более *объявлений*. Каждое объявление, в свою очередь, состоит из *свойства* CSS и *значения* этого свойства.

Блок объявлений всегда заключается в фигурные скобки. Он может состоять из нескольких объявлений, причем каждое объявление завершается точкой с запятой (`;`). Исключением из этого правила является последнее объявление в блоке, где точка с запятой необязательна.

Каждое свойство представляет отдельный параметр стилевого оформления и отделяется от своего значения двоеточием (`:`). Имена свойств в CSS не зависят от регистра букв. Допустимые значения свойств определяются в описании этих свойств. Подробнее о допустимых значениях свойств CSS речь пойдет в главе 4.

Комментарии

Снабдить таблицу CSS комментарием очень просто. Для этого достаточно ввести комментарий между открывающими знаками **/*** и закрывающими знаками ***/** следующим образом:

```
/* Это комментарий! */
```

Комментарии могут быть многострочными:

```
/* Это многострочной комментарий!
   Он продолжается в этой строке
   и завершается в данной строке. */
```

Комментарии можно также вводить в любом месте таблицы стилей, кроме середины обозначения (имени свойства или значения), как показано ниже.

```
h1/* заголовок 1-го уровня */ {color /* цвет фона */:
rgba(23,58,89,0.42) /* цвет RGB + непрозрачность */;}
```

В таблицах стилей допускаются также комментарии к HTML-разметке, а точнее — к SGML-разметке (например, <!-- такой комментарий -->), чтобы скрыть стили от устаревших браузеров, которые уже не распознают разметку в версии HTML 3.2. Они *не* служат в качестве комментариев CSS. Это означает, что содержимое комментария в HTML-разметке будет распознано и интерпретировано синтаксическим анализатором CSS.

Предшествование стилей

Чтобы воспользоваться встроенными стилями, в один HTML- или XHTML-документ можно импортировать и связать с ним несколько внешних таблиц стилей, каждая из которых состоит из одной или более встроенных таблиц стилей. В ходе этого процесса одни правила могут вступить в противоречие с другими. Для разрешения подобных противоречий и получения окончательного ряда стилей, которые можно применить

к документу, в CSS применяется механизм под названием *каскад*. Двумя главными составляющими каскада являются *специфичность* и *наследование*.

Вычисление специфичности

Специфичность описывает вес селектора и любые связанные с ним определения. В следующей таблице сведены слагаемые специфичности.

Тип селектора	Пример	Специфичность
Универсальный селектор	`*`	`0,0,0,0`
Комбинатор	`+`	`0,0,0,1`
Идентификатор элемента	`div`	
Идентификатор псевдоэлемента	`::first-line`	
Идентификатор класса	`.warning`	`0,0,1,0`
Идентификатор псевдокласса	`:hover`	
Идентификатор атрибута	`[type="checkbox"]`	
Идентификатор ID	`#content`	`0,1,0,0`
Встроенный атрибут `style`	`style="color: red;"`	`1,0,0,0`

Значения специфичности накапливаются. Это означает, что селектор, содержащий два идентификатора элементов и один идентификатор класса (например, `div.aside p`), имеет специфичность `0,0,1,2`. Кроме того, значения специфичности сортируются с предшествованием слева направо. А это означает, что селектор, содержащий 11 идентификаторов элементов (`0,0,0,11`), обладает меньшей специфичностью, чем селектор, содержащий только один идентификатор класса (`0,0,1,0`).

В директиве `!important` указывается объявление большего веса, чем у неважных объявлений. Такое объявление сохраняет специфичность своих селекторов и применяется только в сравнении с другими важными объявлениями.

Наследование

Элементы располагаются в форме документа в виде древовидной иерархии, на самом верху которой находится корневой элемент, а ниже него — вся остальная структура документа, благодаря чему она становится похожей на древовидную систему. Элемент разметки `html` располагается на вершине дерева в HTML-документе, ниже него — элементы разметки `head` и `body`, а еще ниже — остальная структура документа. В такой структуре находящиеся ниже элементы являются порожденными от расположенных выше по дереву родительских элементов.

Древовидная структура документа используется в CSS для механизма *наследования*, в котором стиль, применяемый к элементу, наследуется порождаемыми от него элементами. Так, если в атрибуте `red` элемента разметки `body` задан красный цвет, значение этого цвета распространяется вниз по дереву документа к элементам, порожденным от элемента разметки `body`. Наследование прерывается только правилом стиля, применяемым непосредственно к элементу разметки. Наследуемые значения вообще не обладают никакой специфичностью, хотя это совсем не одно и то же, что и обладание нулевой специфичностью.

Следует, однако, иметь в виду, что некоторые элементы вообще не наследуются. При определении свойства всегда указывается, является ни оно наследуемым. К примерам ненаследуемых свойств относятся `padding`, `border`, `margin` и `background`.

Каскад

С помощью механизма каскада в CSS разрешаются конфликты между стилями. Иными словами, этот механизм дает пользовательским агентам возможность выбрать, например, цвет для элемента разметки, когда к нему применяются два разных правила и каждое из них пытается задать свой цвет. Действие каскада можно разделить на следующие стадии.

1. Найти все объявления, содержащие селектор, совпадающий с заданным элементом разметки.
2. Рассортировать по явно заданным весам все объявления, применяемые к заданному элементу разметки. Правила, обозначенные в директиве `!important`, получают больший вес, чем остальные правила. Кроме того, рассортировать по источнику все объявления, применяемые к элементу разметки. Объявления могут иметь следующие источники: автор, читатель и пользовательский агент. Как правило, авторские стили одерживают верх над читательскими стилями. Но читательские стили типа `!important` важнее любых других стилей, в том числе авторских стилей типа `!important`. Как авторские, так и читательские стили заменяют используемые по умолчанию стили пользовательских агентов.
3. Рассортировать по специфичности все объявления, применяемые к заданному элементу разметки. Элементы с большей специфичностью обладают большим весом, чем элементы с меньшей специфичностью.
4. Рассортировать по порядку все объявления, применяемые к заданному элементу разметки. Чем дальше объявление оказывается в таблице стилей или документе, тем больший вес оно получает. Объявления из импортируемой таблицы стилей считаются предшествующими тем объявлениям, которые находятся в импортирующей их таблице стилей, и поэтому они обладают меньшим весом, чем объявления из импортирующей их таблицы стилей.

Любые указания по представлению, источники которых не относятся к CSS (например, диалоговое окно глобальных параметров настройки), получают тот же самый вес, что и используемые по умолчанию стили пользовательских агентов (см. п. 2 приведенной выше процедуры).

Классификация элементов

Вообще говоря, группы элементов разметки CSS делятся на следующие два типа: *незаменяемые* и *заменяемые*. И хотя типы этих элементов могут быть довольно абстрактными, у них имеются некоторые существенные различия. Эти различия подробно рассматриваются в главе третьего издания книги *CSS: The Definitive Guide* того же автора, вышедшей в издательстве O'Reilly (в русском переводе эта книга вышла под названием *CSS — Каскадные таблицы стилей. Подробное руководство* в издательстве "Символ-плюс", Спб., 2008).

Незаменяемые элементы

Большинство элементов разметки HTML- и XHTML-документов являются *незаменяемыми*, а это означает, что их содержимое представлено пользовательским агентом в блоке, формируемом самим элементом. Например, элемент разметки `<span>hi there</span>` является незаменяемым, а размечаемый им текст `"hi there"` отображается пользовательским агентом. Абзацы, заголовки, ячейки таблиц, списки и практически все содержимое HTML- и XHTML-документов относятся к незаменяемым элементам разметки.

Заменяемые элементы

С другой стороны, *заменяемыми* называются такие элементы разметки, содержимое которых заменяется тем, что непосредственно отсутствует в самом документе. Наиболее характерным тому примером служит элемент разметки `img`, заменяемый файлом изображения, являющимся внешним по отношению к документу. В действительности сам элемент разметки `img` ничего не содержит, как следует из приведенного ниже простого примера.

```
<img src="howdy.gif" alt="Hi">
```

Заменяемый элемент, по существу, состоит из своего имени и атрибутов. И только заменив его содержимым, обнаруживаемым другими средствами (в данном случае — путем загрузки изображения из внешнего файла, указываемого в атрибуте `src`), можно что-нибудь вообще представить с помощью такого элемента разметки. Еще одним характерным примером служит элемент разметки `input`, который можно заменить кнопкой-переключателем, флажком или полем ввода текста в зависимости от типа этого элемента. Заменяемые элементы разметки формируют также блоки в местах их отображения.

Роли элементов в отображении

Помимо того что элементы разметки являются заменяемыми или незаменяемыми, они выполняют следующие основные роли в отображении по стандарту CSS3: *на уровне блока* и *на уровне строки*. Все значения свойства `display` по стандарту CSS3 относятся к одной из этих двух категорий. Очень важно знать, к какой именно общей роли относится блок, поскольку некоторые свойства применяются только для одной категории ролей в отображении.

Отображение на уровне блока

На уровне блока (по умолчанию — структурного блока) отображается такой блок элемента, который заполняет область содержимого по всей ширине и не может иметь других элементов по обеим своим сторонам. Иными словами, элементы на уровне блоков формируют "разрывы" до и после блока элемента. Наиболее известными примерами блочных элементов служат элементы `p` и `div` разметки HTML-документов. Заменяемые элементы могут быть блочными, но, как правило, они таковыми не бывают.

Элементы списка являются особым случаем блочных элементов. Помимо того что они ведут себя так же, как и другие

блочные элементы, они формируют маркер (как правило, жирную точку в маркированных списках или номер по порядку в нумерованных списках), который присоединяется к блоку элемента. Кроме маркера, элементы списка во всем остальном похожи на другие блочные элементы.

Для создания структурных блоков имеются следующие значения свойства `display`: `block`, `listitem`, `table`, `table-row-group`, `table-header-group`, `tablefooter-group`, `table-row`, `table-column-group`, `table-column`, `table-cell`, `table-caption`, а также все шаблоны CSS Advanced Layout для усовершенствованной разметки средствами CSS (на момент написания данной книги).

Отображение на уровне строки

На уровне строки отображается такой блок элемента, который формируется в текстовой строке, не разрывая последовательное расположение этой строки. К числу наиболее известных внутристрочных элементов относится элемент `a` разметки HTML- и XHTML-документов, а также элементы `span` и `em`. Эти элементы разметки не образуют разрывы строки до и после себя, а следовательно, могут присутствовать в содержимом другого элемента, не нарушая его отображение.

Несмотря на то что блочные и внутристрочные элементы CSS имеют немало общего с блочными и внутристрочными элементами разметки HTML- и XHTML-документов, у них все же имеется одно важное отличие. Блочные элементы разметки HTML- и XHTML-документов не могут наследовать от внутристрочных элементов разметки, тогда как в CSS не накладывается никаких ограничений на порядок взаимного вложения ролей элементов при отображении.

Для создания внутристрочных блоков имеются следующие значения свойства `display`: `inline`, `inlineblock`, `inline-table` и `ruby`. На момент написания данной книги еще не было ясно определено, что различные значения данного свойства,

имеющие отношение к оформлению идеографического текста (например, значение `ruby-text`), также способны формировать внутристрочные блоки элементов, хотя это, по-видимому, наиболее вероятный результат их применения.

Вставка

Особый случай представляют собой *вставляемые блоки*, определяемые разметкой `display: run-in`, которая позволяет сформировать структурный или внутристрочной блок в зависимости от конкретной ситуации. Получаемый в итоге результат определяется следующими правилами.

1. Если сам вставляемый блок содержит структурный блок, он формирует такой блок.
2. А если после вставляемого блока сразу же следует родственный структурный блок, который не перемещается свободно и не имеет абсолютного расположения, то вставляемый блок становится первым внутристрочным блоком этого родственного блока.
3. Если же ни одно из упомянутых выше условий не выполняется, то вставляемый блок формирует структурный блок.

Если вставляемый блок вводится в виде первого внутристрочного блока его родственного структурного блока (см. приведенное выше правило 2), то он наследует значения свойств *не* из этого блока, а продолжает наследовать от своего структурного родительского элемента. Так, если в структурном родственном блоке применяется свойство `color: green`, то зеленый цвет не будет непосредственно наследоваться вставляемым элементом, несмотря на то что он визуально является частью этого блока.

Основы визуальной разметки

Для разметки любого элемента документа в CSS определяются специальные алгоритмы. Они служат основанием для визуального представления содержимого в CSS. Имеются следующие две основные разновидности разметки, ведущие себя по-разному: блочная и внутристрочная.

Блочная разметка

Структурный блок формирует в CSS прямоугольный участок, называемый *блоком элемента* и описывающий пространство, занимаемое этим элементом. Различные составляющие блока элемента приведены на рис. 1.2. Блок элемента оформляется по следующим правилам.

- Фон блока элемента простирается до внешнего края границы, тем самым заполняя участки содержимого, внутреннего поля и границы. Если у границы имеются любые прозрачные участки (например, пунктиры и штрихи), фон на них будет просматриваться. Фон не простирается до участков отступов в блоке. Любые контуры, рисуемые на участках отступов, не оказывают влияния на разметку.
- Значение `auto` может быть установлено в свойствах `height` и `width` блока элемента только для отступов.
- Отрицательные значения могут быть заданы только для отступов.
- Ширина внутреннего поля и границы в блоке элемента по умолчанию равна нулю (**0**) и **`none`** соответственно.
- Если свойство `box-sizing` принимает (по умолчанию) значение **`content-box`**, то свойство `width` определяет ширину только участка содержимого, к которому добавляются любые внутренние поля, границы или отступы. То же самое относится и к высоте участка содержимого, определяемой свойством `height`.

- Если свойство `box-sizing` принимает значение **`border-box`**, то свойство `width` определяет общую ширину участков содержимого, внутренних полей и границ, к которым добавляются любые отступы. То же самое относится и к высоте участка содержимого, определяемой свойством `height`.

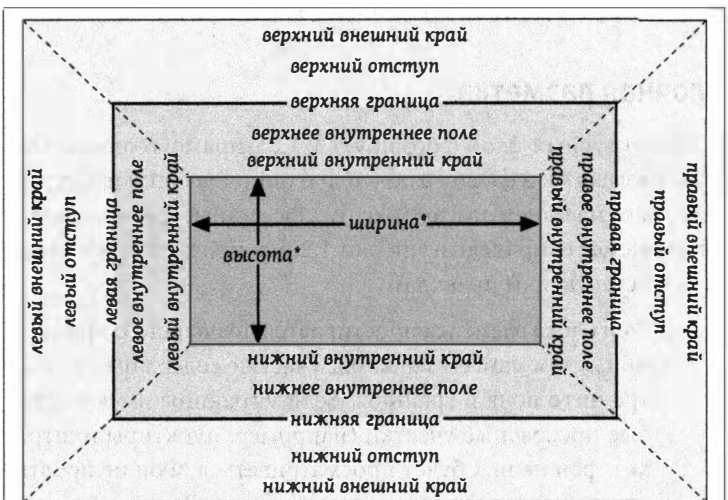


Рис. 1.2. Блочная разметка

Внутристрочная разметка

Внутристрочной блок формирует в CSS один или более прямоугольных участков, называемых *строковыми блоками*, в зависимости от того, разбивается ли строковый блок на несколько строк. Строковый блок оформляется по следующим правилам.

- Любое значение **auto** свойств `left`, `right`, `top`, `bottom`, `margin-left`, `margin-right`, `margin-top` и `margin-bottom` преобразуется в нулевое (**0**).
- Свойства `width` и `height` не применимы к незаменяемым строковым блокам.
- К заменяемым строковым блокам применяются следующие правила.
  - Если оба свойства, `width` и `height`, принимают значение **auto**, а элементу (например, изображению) свойственна ширина, то значение свойства `width` становится равным ширине, свойственной данному элементу. То же самое относится и к свойству `height`.
  - Если оба свойства, `width` и `height`, принимают значение **auto**, а элементу *не* свойственна ширина, то значение свойства `width` становится равным свойственной данному элементу высоте, умноженной на пропорцию (т.е. соотношение ширины и высоты).
  - Если оба свойства, `width` и `height`, принимают значение **auto**, а элементу *не* свойственна высота, но свойственна ширина и пропорция разметки, то значение свойства `height` становится равным свойственной данному элементу ширине, деленной на пропорцию.

Имеется ряд других правил, менее ясных, чем перечисленные выше, но их пояснение слишком обширно, чтобы привести его здесь. Подробнее об этих правилах читайте по адресу `http://w3.org/TR/css3-box/#inline-replaced`.

Все внутристрочные элементы обладают свойством `line-height`, имеющим непосредственное отношение к отображаемым элементам. В частности, высота текстовой строки определяется с учетом следующих факторов.

- **Анонимный текст** — любая символьная строка, не входящая во внутристрочной элемент. Так, в разметке

```
<p> I'm <em>so</em> happy!</p>
```

последовательности строк `"I'm"` и `" happy!"` образуют анонимный текст. Следует иметь в виду, что пробелы являются частью анонимного текста, поскольку пробел — это такой же символ, как и любой другой.

- **Кегельный квадрат** определяется в заданном шрифте, а иначе называется *знакоместом*. Отдельные глифы (наименьшие единицы шрифта) могут быть длиннее или короче, чем их кегельные квадраты, как поясняется в главе 5 упоминавшегося ранее третьего издания книги *CSS: The Definitive Guide*. Значение свойства `font-size` определяется в CSS высотой каждого кегельного квадрата.
- **Область содержимого** в незаменяемых элементах может быть блоком, описываемым связанными вместе кегельными квадратами каждого символа в данном элементе, а иначе — блоком, описываемым глифами символов в данном элементе. В спецификации CSS2.1 пользовательским агентам разрешается выбирать любую из этих разновидностей области содержимого. Ради простоты здесь выбрана первая разновидность области содержимого, описываемая кегельными квадратами. А в заменяемых элементах область содержимого определяется свойственной элементу высотой и любыми отступами, границами или внутренними полями.
- **Интерлиньяж** — это разность значений свойств `fontsize` и `line-height`. Одна половина этой разности приходится на верхнюю часть области содержимого, а другая — на нижнюю часть данной области. Поэтому и не удивительно, что подобные дополнения области содержимого называются *половинным интерлиньяжем*. Интерлиньяж применяется только к незаменяемым элементам.
- **Строковый блок** — это блок, описываемый добавлением интерлиньяжа к области содержимого. Для незаменяемых элементов высота строкового блока элемента

оказывается равной значению свойства `lineheight`, а для заменяемых элементов — величине области содержимого, поскольку интерлиньяж не применяется к заменяемым элементам.

- **Построчный блок** — это самый короткий блок, ограничиваемый наивысшей и наинизшей точками строковых блоков, обнаруживаемых в строке. Иными словами, верхний край построчного блока будет располагаться вдоль верхнего края самого высокого строкового блока, а нижний край — вдоль нижнего края самого низкого строкового блока (рис. 1.3).

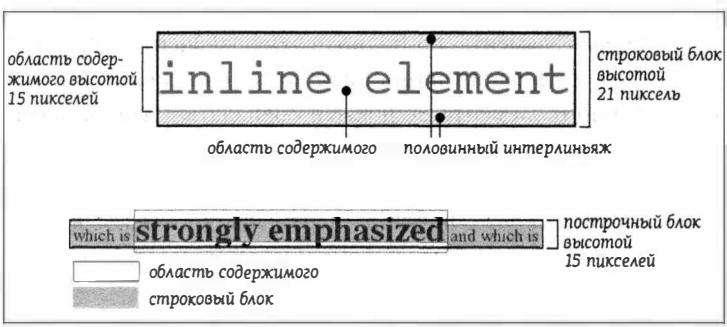


***Рис. 1.3.** Внутристрочная разметка*

Свободное перемещение

Свободное перемещение (или так называемое "плавание") позволяет разместить элемент слева или справа от блока, который его содержит и является ближайшим родственным элементом на уровне блока, а последующее содержимое будет обтекать этот элемент. Любой свободно перемещаемый элемент автоматически формирует структурный блок независимо от типа блока, который был сформирован, если этот элемент не был свободно перемещаемым. Свободно перемещаемый элемент размещается по следующим правилам.

- Левый (или правый) внешний край свободно перемещаемого элемента не должен совпадать с левым (или правым) внутренним краем содержащего его блока.
- Левый (или правый) внешний край свободно перемещаемого элемента должен располагаться справа (или слева) от правого (или левого) внешнего края левого (или правого) левого (или правого) свободно перемещаемого элемента, встречающегося раньше в исходном документе, если только верхний край последнего элемента не оказывается ниже нижнего края первого элемента.
- Правый внешний край левого свободно перемещаемого элемента не должен совпадать с правым внешним краем любого элемента, свободно перемещающегося справа от него. А левый внешний край правого свободно перемещаемого элемента не должен совпадать с левым внешним краем любого элемента, свободно перемещающегося слева от него.
- Верхний край свободно перемещаемого элемента не должен быть выше верхнего внутреннего края содержащего его блока.
- Верхний край свободно перемещаемого элемента не должен быть выше верхнего края любого расположенного прежде свободно перемещаемого или блочного элемента.
- Верхний край свободно перемещаемого элемента не должен быть выше верхнего края построчного элемента с содержимым, предшествующим этому свободно перемещаемому элементу.
- Правый (или левый) внешний край левого (или правого) свободно перемещаемого элемента, слева (или справа) от которого находится другой свободно перемещаемый элемент, не должен располагаться справа (или слева) от правого (или левого) края содержащего его блока.
- Свободно перемещаемый элемент должен располагаться как можно выше.

- Левый свободно перемещаемый элемент должен быть смещен как можно дальше влево, а правый свободно перемещаемый элемент — как можно дальше вправо. Чем дальше влево или вправо смещается элемент, чем выше он должен располагаться.

Расположение элементов

Существует целый ряд специальных правил расположения элементов. Эти правила определяют не только содержащий блок элемента, но и порядок его разметки в данном элементе.

Виды расположения

Ниже перечислены виды расположения элементов.

- **Статическое расположение.** Блок элемента формируется как обычно. Блочные элементы образуют прямоугольный блок, являющийся частью потока документа, а внутристрочные блоки — один или более построчных блоков в их родительском элементе.
- **Относительное расположение.** Блок элемента смещается на некоторое расстояние. Содержащий его блок может считаться областью, которую занял бы этот элемент, если бы он не был расположен. Элемент сохраняет ту форму, которую он имел бы, если бы не был расположен. И за этим элементом сохраняется пространство, которое он обычно занимал бы.
- **Абсолютное расположение.** Блок элемента полностью удаляется из потока документа и располагается относительно содержащего его блока, которым может быть другой элемент документа или начальный содержащий блок, рассматриваемый в следующем разделе. Какое место ни занимал бы элемент в обычном потоке документа, оно изымается, как будто элемент вообще не

существовал. Расположенный таким образом элемент формирует структурный блок независимо от типа блока, который он сформировал бы, если бы находился в обычном потоке документа.

- **Фиксированное расположение.** Блок элемента ведет себя так, как будто он имеет абсолютное расположение, но содержащий его блок сам является областью просмотра.

Содержащий блок

Содержащий блок расположенного элемента определяется следующим образом.

1. Содержащий блок *корневого элемента*, иначе называемый *начальным содержащим блоком*, устанавливается пользовательским агентом. В HTML-разметке корневым является элемент разметки `html`, хотя в некоторых браузерах для этой цели может быть использован элемент разметки `body`.
2. Если свойство `position` некорневого элемента принимает значение `relative` или `static`, то содержащий его блок образуется по краю содержимого ближайшего структурного, табличного, клеточного или внутристрочного родственного блока. Несмотря на это правило, элементы с относительным расположением попрежнему просто смещаются, а не располагаются относительно рассматриваемого здесь содержащего блока, а элементы со статическим расположением не смещаются со своего места в обычном потоке документа.
3. Если свойство `position` некорневого элемента принимает значение `absolute`, то в качестве содержащего блока задается ближайший родительский элемент (любого типа), свойство которого `position` принимает значение, отличающееся от `static`. Это происходит следующим образом.

а) Если родительский элемент является блочным, то содержащий блок доходит до внешнего края внутреннего поля этого элемента. Иными словами, это область, ограниченная границей данного элемента.

б) Если родительский элемент является внутристрочным, то содержащий блок доходит до края содержимого данного элемента. В языках с правосторонним письмом верхний и левый края содержащего блока совпадают с верхним и левым краями первого блока в родительском элементе, а нижний и правый края — с нижним и правым краями содержимого последнего блока. А в языках с левосторонним письмом правый край содержащего блока совпадает с правым краем содержимого первого блока, тогда как левый край выбирается из последнего блока. Верхний и нижний края остаются теми же самыми.

в) Если родительский блок, описанный в пп. 3, *а* и 3, *б*, отсутствует, блок, содержащий элемент с абсолютным расположением, определяется как начальный.

Разметка элементов с абсолютным расположением

В следующих разделах употребляются следующие термины.

- **Подгонка страниц.** Алгоритм, действующий подобно вычислению ширины ячейки таблицы по алгоритму автоматической разметки таблицы. В общем, пользовательский агент пытается найти минимальную ширину элемента, чтобы его содержимое автоматически разбивалось на несколько строк только в том случае, если этого нельзя избежать.
- **Статическое положение.** Место, где мог бы находиться край элемента, если бы его положение было статическим.

Горизонтальная разметка элементов с абсолютным расположением

Горизонтальная разметка подобных элементов осуществляется по следующей формуле:

```
ширина содержащего блока = left + margin-left +
border-left-width +
padding-left + width + padding-right + border-right-width +
margin-right + right + ширина вертикальной полосы прокрутки
(если таковая имеется)
```

Ширина любой вертикальной полосы прокрутки определяется пользовательским агентом и не поддается воздействию со стороны CSS.

Для определения горизонтальной разметки *незаменяемых* элементов служит следующая процедура.

1. Если во всех свойствах `left`, `width` и `right` установлены значения `auto`, то сначала сбросить в нуль (`0`) любые значения `auto` свойств `margin-left` и `margin-right`. И если в свойстве `direction` установлено значение `ltr`, задать статическое положение в свойстве `left` и применить правило, указанное выше, в п. 3, *в*. В противном случае задать статическое положение в свойстве `right` и применить правило, указанное выше, в п. 3, *а*.
2. Если ни в одном из свойств `left`, `width` и `right` не установлены значения `auto`, то выбрать для применения одно из следующих правил.
   а) Если в обоих свойствах, `margin-left` и `margin-right`, установлено значение `auto`, решить приведенное выше уравнение с дополнительным ограничением на равные значения обоих отступов слева и справа.
   б) Если значение `auto` установлено только в свойстве `margin-left` или `margin-right`, решить приведенное выше уравнение для этого значения.

в) Если значения свойств чрезмерно ограничены (ни одно из них не равно auto), пренебречь значением свойства left, если в свойстве direction задано значение rtl, или же значением свойства right, если в свойстве direction задано значение ltr, и тогда решить приведенное выше уравнение для этого значения.

3. Если в одних из свойств left, width и right установлено значение auto, а в других — нет, сбросить в нуль (0) любые значения auto свойств margin-left и margin-right. Выбрать для применения одно из следующих правил.

   а) Если значение auto установлено в свойствах left и width, а в свойстве right — нет, ширина определяется подгонкой страниц. Решить приведенное выше уравнение для значения в свойстве left.

   б) Если значение auto установлено в свойствах left и right, тогда как в свойстве width — нет, задать статическое положение в свойстве left, а иначе — в свойстве right, если в свойстве direction установлено значение ltr. Решить приведенное выше уравнение для значения в свойстве left, если в свойстве direction установлено значение rtl, или для значения в свойстве right, если в свойстве direction задано значение ltr.

   в) Если значение auto установлено в свойствах width и right, а в свойстве left — нет, ширина определяется подгонкой страниц. Решить приведенное выше уравнение для значения в свойстве right.

   г) Если значение auto установлено в свойстве left, а в свойствах width и right — нет, решить приведенное выше уравнение для значения в свойстве left.

   д) Если значение auto установлено в свойстве width, а в свойствах left и right — нет, решить приведенное выше уравнение для значения в свойстве width.

е) Если значение auto установлено в свойстве right, а в свойствах left и width — нет, решить приведенное выше уравнение для значения в свойстве right.

Для определения горизонтальной разметки *заменяемых* элементов служит следующая процедура.

1. Определить значение свойства width, как описано ранее для внутристрочных заменяемых элементов в разделе "Внутристрочная разметка".
2. Если значение auto установлено в обоих свойствах, left и right, задать статическое левое положение в свойстве left, если в свойстве direction установлено значение ltr. А если в свойстве direction установлено значение rtl, задать статическое правое положение в свойстве right.
3. Если значение auto установлено в обоих свойствах, left и right, или в одном из них, сбросить в нуль (0) любые значения auto свойств margin-left и margin-right.
4. Если значение auto не установлено ни в одном из свойств left и right, а в свойствах margin-left и margin-right установлено значение auto, решить приведенное выше уравнение с дополнительным ограничением на равные значения обоих отступов слева и справа.
5. Если значения свойств чрезмерно ограничены (ни одно из них не равно auto), пренебречь значением свойства left, если в свойстве direction задано значение rtl, или же значением свойства right, если в свойстве direction задано значение ltr, и тогда решить приведенное выше уравнение для этого значения.

Вертикальная разметка элементов с абсолютным расположением

Вертикальная разметка подобных элементов осуществляется по следующей формуле:

```
высота содержащего блока = top + margin-top +
border-top-width + padding-top +
height + padding-bottom + border-bottom-width +
margin-bottom + bottom + высота горизонтальной полосы
прокрутки
(если таковая имеется)
```

Высота любой горизонтальной полосы прокрутки определяется пользовательским агентом и не поддается воздействию со стороны CSS.

Для определения вертикальной разметки *незаменяемых* элементов служит следующая процедура.

1. Если значение `auto` установлено во всех свойствах `top`, `height` и `bottom`, задать статическое положение в свойстве `top` и применить правило, указанное выше, в п. 3, *в*.
2. Если значение `auto` не установлено в свойствах `top` и `height`, но установлено в свойстве `bottom`, выбрать для применения одно из следующих правил.
   а) Если значение `auto` установлено в обоих свойствах `margin-top` и `margin-bottom`, решить приведенное выше уравнение с дополнительным ограничением на равные значения обоих отступов слева и справа.
   б) Если значение `auto` установлено только в одном из свойств `margin-top` и `margin-bottom`, решить приведенное выше уравнение для данного значения.
   в) Если значения свойств чрезмерно ограничены (ни одно из них не равно `auto`), пренебречь значением свойства `bottom` и решить приведенное выше уравнение для данного значения.
3. Если значение `auto` установлено в одних свойствах `top`, `height` и `bottom`, а в других — не установлено, выбрать для применения одно из следующих правил.

а) Если значение auto установлено в свойствах top и height, а в свойстве bottom — нет, высота определяется, исходя из содержимого элемента, как это обычно делается в статическом потоке. Сбросить в нуль (0) любые значения auto свойств margin-top и margin-bottom и решить приведенное выше уравнение для значения свойства top.

б) Если значение auto установлено в свойствах top и bottom, а в свойстве height — нет, задать статическое положение в свойстве top. Сбросить в нуль (0) любые значения auto свойств margin-top и margin-bottom и решить приведенное выше уравнение для значения свойства bottom.

в) Если значение auto установлено в свойствах height и bottom, а в свойстве top — нет, высота определяется, исходя из содержимого элемента, как это обычно делается в статическом потоке. Сбросить в нуль (0) любые значения auto свойств margin-top и margin-bottom и решить приведенное выше уравнение для значения свойства bottom.

г) Если значение auto установлено в свойстве top, а в свойствах height и bottom — нет, сбросить в нуль (0) любые значения auto свойств margin-top и margin-bottom и решить приведенное выше уравнение для значения свойства top.

д) Если значение auto установлено в свойстве height, а в свойствах top и bottom — нет, сбросить в нуль (0) любые значения auto свойств margin-top и margin-bottom и решить приведенное выше уравнение для значения свойства height.

е) Если значение auto установлено в свойстве bottom, а в свойствах top и height — нет, сбросить в нуль (0) любые значения auto свойств margin-top и margin-bottom и решить приведенное выше уравнение для значения свойства bottom.

Для определения вертикальной разметки *заменяемых* элементов служит следующая процедура.

1. Определить значение свойства `height`, как описано для внутристрочных заменяемых элементов ранее, в разделе "Внутристрочная разметка".
2. Если значение `auto` установлено в обоих свойствах, `top` и `bottom`, задать статическое верхнее положение в свойстве `top`.
3. Если значения свойств чрезмерно ограничены, пренебречь значением свойства `bottom` и решить приведенное выше уравнение для данного значения.

Разметка таблиц

Разметка таблиц может оказаться довольно сложной, особенно потому что в CSS определяются два разных способа вычисления ширины ячеек таблицы, а также два способа обращения с границами таблиц и их внутренними элементами. Составляющие разметки таблиц приведены на рис. 1.4.

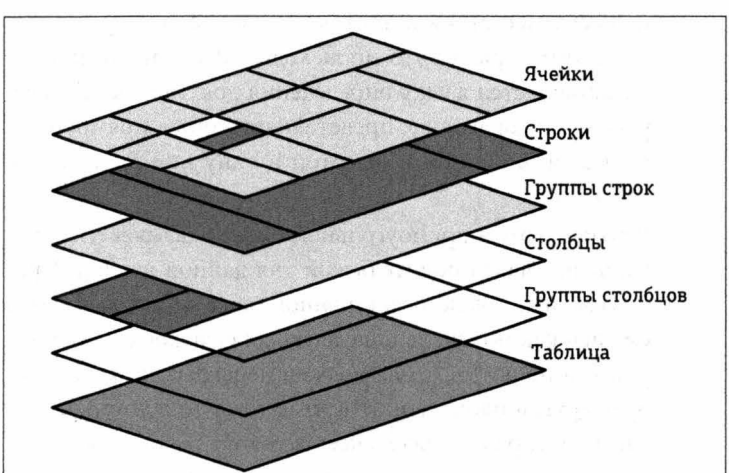


Рис. 1.4. *Составляющие разметки таблиц*

Правила расположения таблиц

В общем, таблица размечается в соответствии со следующими правилами.

- Каждый блок строки охватывает одну строку ячеек таблицы. Все блоки строки в таблице заполняют ее сверху вниз в том порядке, в каком они встречаются в исходном документе. Следовательно, таблица состоит из стольких строк, сколько имеется элементов строк.
- Блок группы строк охватывает те же самые ячейки таблицы, что и блоки строк, которые он содержит.
- Блок столбца охватывает один или более столбцов ячеек таблицы. Блоки столбцов размещаются один за другим в том порядке, в каком встречаются в исходном документе. Первый блок столбца располагается слева для языков с правосторонним письмом или справа для языков с левосторонним письмом.
- Блок группы столбцов охватывает те же самые ячейки таблицы, что и блоки столбцов, которые он содержит.
- Несмотря на то что ячейки таблицы могут простираться на несколько строк или столбцов, в CSS не определяется, каким образом это происходит. Решение этой задачи оставляется языку определения документов. Каждая расширенная ячейка представляет собой прямоугольный блок шириной и высотой в одну или более ячеек таблицы.

  Верхней в этом прямоугольном блоке оказывается строка, являющаяся родительской для данной ячейки. Прямоугольный блок расширенной ячейки должен быть смещен как можно дальше влево для языков с правосторонним письмом, но не должен перекрывать какой-нибудь другой блок ячейки. Кроме того, он должен располагаться справа от всех ячеек в той же самой строке, которые следуют до него в исходном документе на языке с правосторонним письмом. А в языках с левосторонним

письмом расширенная ячейка должна быть смещена как можно дальше вправо, не перекрывая остальные ячейки, и располагаться слева от всех ячеек в той же самой строке, которые следуют после нее в исходном документе.

- Блок ячейки не должен простираться за пределы последнего блока строки или группы строк таблицы. Если структура таблицы является причиной такого состояния, ячейка должна быть сокращена до тех пор, пока не впишется в таблицу или группу строк, которая ее охватывает.

Фиксированная разметка таблиц

Модель фиксированной разметки действует быстро, поскольку не зависит от содержимого ячеек таблицы. Ее действие определяется значениями свойства `width` таблицы, столбцов и ячеек в первой строке таблицы. В модели фиксированной разметки таблиц применяется следующая простая процедура.

1. Любой элемент столбца, значение свойства `width` которого отличается от `auto`, задает ширину этого столбца.
2. Если ширина столбца устанавливается автоматически, а ширина ячейки первой строки таблицы в этом столбце — не автоматически, именно эта ячейка и задает ширину данного столбца. Если же ячейка простирается на несколько столбцов, ее ширина равномерно распределяется среди этих столбцов.
3. Любые столбцы, ширина которых устанавливается автоматически, размечаются таким образом, чтобы иметь как можно более одинаковую длину.

На данном этапе ширина таблицы задается значением свойства `width` таблицы или суммарной шириной столбцов в зависимости от того, что больше. Если же таблица оказывается шире, чем суммарная ширина столбцов, разница делится на количество столбцов и прибавляется к ширине каждого из них.

Автоматическая разметка таблиц

Модель автоматической разметки таблиц действует не так быстро, как модель фиксированной разметки таблиц, тем не менее эта модель в большей степени известна авторам веб-страниц, поскольку она, по существу, аналогична модели HTML-разметки таблиц, которая применяется уже много лет. Применение этой модели в последних пользовательских агентах можно задать, установив значение `auto` в свойстве `width`, независимо от значения свойства `table-layout`, хотя это и не гарантируется.

В модели автоматической разметки таблиц применяется следующая процедура.

1. Вычисляется минимальная и максимальная ширина каждой ячейки в столбце.
2. Определяется минимальная ширина, необходимая для отображения содержимого. С этой целью содержимое может быть разбито на любое количество строк, но не должно выступать за пределы блока ячейки. Если значение свойства `width` ячейки оказывается больше минимально возможной ширины, в этом свойстве устанавливается значение минимальной ширины ячейки. А если значение свойства `width` ячейки оказывается равным `auto`, минимальная ширина ячейки задается по минимальной ширине содержимого.
3. Определяется ширина, необходимая для отображения содержимого без его разбиения на ряд строк, в отличие от принудительного разбиения на ряд строк (например, в силу действия элемента разметки `<br>`). Это значение и определяет максимальную ширину ячейки.
4. Вычисляется минимальная и максимальная ширина каждого столбца следующим образом.
   а) Минимальная ширина столбца определяется наибольшей минимальной шириной всех ячеек в столбце. Если в свойстве `width` столбца явно задано

значение, большее любой минимальной ширины ячеек в этом столбце, его минимальная ширина задается по значению свойства `width`.

б) Максимальная ширина столбца определяется наибольшей максимальной шириной всех ячеек в столбце. Если в свойстве `width` столбца явно задано значение, большее любой максимальной ширины ячеек в этом столбце, его максимальная ширина задается по значению свойства `width`. Оба эти расчета ширины столбца воспроизводят традиционную HTML-разметку таблиц, при которой любой столбец принудительно простирается на ширину самой широкой его ячейки.

5. Если ячейка простирается на несколько столбцов, суммарная минимальная ширина всех столбцов должна быть равна минимальной ширине расширенной ячейки. Аналогично суммарная максимальная ширина всех столбцов должна быть равна максимальной ширине расширенной ячейки. Пользовательские агенты должны равномерно распределять любые различия в ширине столбцов среди всех охватываемых данной ячейкой столбцов.

Кроме того, пользовательский агент должен принять во внимание следующее: если ширина столбца содержит долю в процентах от значения его свойства `width`, то такая доля вычисляется относительно ширины таблицы, даже если эта ширина еще неизвестна. Пользовательский агент должен хранить полученную долю в процентах, чтобы использовать ее в следующей части алгоритма. Как только пользовательский агент определит, насколько шире или *уже* должен быть каждый столбец, он сможет вычислить ширину таблицы. И делается это следующим образом.

1. Если вычисленная ширина таблицы не равна значению `auto`, то она сравнивается с суммарной шириной всех

столбцов, включая любые границы и промежутки между ячейками таблицы. (На данном этапе, скорее всего, вычисляется доля в процентах от ширины столбцов.) Большее из сравниваемых значений и определяет окончательную ширину таблицы. Так, если вычисленная ширина таблицы оказывается больше суммарной ширины всех столбцов, границ и промежутков между ячейками, то ширина столбцов увеличивается на одинаковую величину, чтобы заполнить таблицу по всей ее ширине.

2. Если вычисленная ширина таблицы равна значению `auto`, то окончательная ширина таблицы определяется суммированием ширины всех столбцов, границ и промежутков между ячейками. Это означает, что таблица будет широкой настолько, насколько потребуется для отображения ее содержимого, как и при традиционной HTML-разметке таблиц. А любая доля в процентах от ширины столбцов служит в качестве ограничения, накладываемого на их ширину, но пользовательскому агенту совсем не обязательно соблюдать это ограничение.

Когда завершится последний из приведенных выше этапов вычисления ширины таблицы, тогда (и только тогда) пользовательский агент сможет фактически разметить таблицу.

Сведение границ ячеек таблицы

Модель сведения границ ячеек таблицы описывает, главным образом, традиционную HTML-разметку таблиц без промежутков между ячейками (так называемого межклеточного расстояния). Эта модель действует по следующим правилам.

- Элементы таблицы не могут иметь никаких внутренних полей, хотя им разрешается иметь отступы. Это, по существу, означает отсутствие разделения между границей, проходящей по ее внешнему краю, и крайними ячейками.

- Границы могут накладываться на ячейки, строки, группы строк, столбцы и группы столбцов. И сам элемент разметки `table` может, как обычно, иметь границу.
- Разделение между границами ячеек отсутствует. В действительности границы сводятся одна в другую там, где они соприкасаются, чтобы нарисованной оказалась лишь одна из сведенных границ. Это сродни сведению отступов, когда верх одерживает самый крупный отступ. А при сведении границ ячеек верх одерживает та граница, которая представляет наибольший интерес.
- Как только границы между ячейками будут свернуты, они центруются по гипотетическим линиям сетки, условно проведенным между ячейками.

Сведение границ

Если рядом находятся две или более границ, они сводятся одна в другую, как показано на рис. 1.5. Какая именно граница одерживает при этом верх, определяют следующие строгие правила.

1. Если свойство `border-style` одной из сводимых границ принимает значение `hidden`, именно ей отдается наибольший приоритет над всеми остальными сводимыми границами, которые в данном месте скрываются.
2. Если свойство `border-style` одной из сводимых границ принимает значение `none`, ей отдается наименьший приоритет. Никакой границы не рисуется, если свойство `border-style` всех границ, сходящихся в данном месте, также принимает значение `none`. Следует иметь в виду, что значение `none` устанавливается в свойстве `border-style` по умолчанию.
3. Если свойство `border-style` хотя бы одной из сводимых границ принимает значение `none` или `hidden`, узкие границы уступают место более широким. Если две или более сводимых границ имеют одинаковую ширину, стиль границы принимает следующий порядок, начиная

с наименее предпочтительного: double, solid, dashed, dotted, ridge, outset, groove, inset. Так, если две сводимые границы имеют одинаковую длину, одна из них — стиль dashed, а другая — стиль inset, то граница в данном месте оформляется штрихом.

4. Если сводимые границы имеют одинаковые стиль и ширину, но различаются цветом, то цвет выбирается из следующего списка, начиная с наиболее предпочтительного элемента: ячейки, строки, группы строк, столбца, группы столбцов, таблицы. Так, если границы ячейки и столбца оказываются сходными, кроме цвета, они сводятся и выбирается цвет (стиль и ширина) границы ячейки. А если сводимые границы происходят от элемента одного и того же типа (например, границы двух строк одинакового стиля и ширины, но разного цвета), то верх одерживает граница крайней слева и сверху

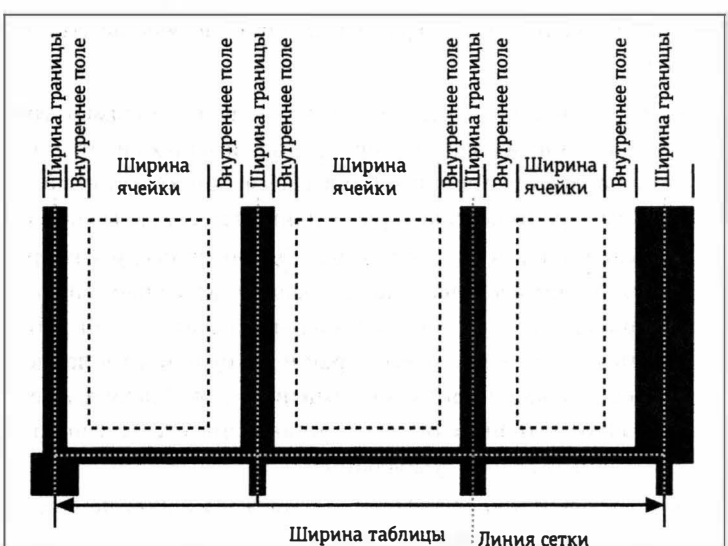


Рис. 1.5. Модель сведения границ ячеек

ячейки в языках с правосторонним письмом или граница крайней справа и сверху ячейки в языках с левосторонним письмом.

Выравнивание содержимого ячеек по вертикали

Ниже подробно описан процесс выравнивания содержимого ячеек таблицы по вертикали в пределах строки таблицы.

1. Если любая из ячеек выравнивается по базовой линии, то определяется базовая линия строки и размещается содержимое ячеек, выравниваемых по базовой линии.
2. Размещается содержимое любой выравниваемой сверху ячейки. Теперь у строки имеется ориентировочная высота, которая определяется по нижнему краю самой нижней ячейки, содержимое которой уже размещено.
3. Если любые из оставшихся ячеек выравниваются посредине или снизу, а высота их содержимого оказывается больше ориентировочной высоты строки, то высота строки увеличивается опусканием базовой линии вниз с целью охватить самые высокие ячейки.
4. Размещается содержимое всех остальных ячеек. Внутренне поле любой ячейки, содержимое которой оказывается более коротким, чем высота строки, увеличивается для совпадения со строкой по высоте.

Значения

В спецификации CSS определяются самые разные типы значений, в большинстве из которых употребляются единицы измерения. Сочетая основные типы значений (например, числовых) с единицами измерения (например, пикселями), можно добиться практически любых результатов привлекательного оформления документов средствами CSS.

Ключевые слова

Ключевые слова определяются на основе свойств и имеют назначение, характерное только для данного свойства. Например, ключевое слово **`normal`** имеет совершенно особое назначение в свойствах `font-variant` и `letter-spacing`. Как и в именах свойств, в ключевых словах регистр букв не учитывается. Особое место принадлежит ключевому слову **`inherit`**, которое допускается во всех свойствах и всегда имеет одно и то же назначение: получить связанное со свойством значение из родительского элемента. Имеется еще одно ключевое слово **`initial`**, которое специально предназначено для представления исходного или устанавливаемого по умолчанию значения данного свойства. Таким образом, в результате объявления `font-family: initial` возвращается гарнитура шрифта (Times для большинства пользователей), устанавливаемая среди глобальных параметров браузера по умолчанию. Состояние и применение ключевого слова **`initial`** не были ясны на момент написания этой книги, и поэтому могут быть ненадежны.

Значения цвета

Ниже перечислены формы обозначения значений цвета.

#RRGGBB

Это обозначение в виде пар шестнадцатеричных значений хорошо известно тем, кто пользуется традиционной HTML-разметкой в авторской разработке веб-страниц. В данной форме записи первая пара цифр соответствует интенсивности красного цвета, вторая пара — интенсивности зеленого цвета, а третья — интенсивности синего цвета. В шестнадцатеричном обозначении каждая пара цифр указывается в пределах **00-FF**. Таким образом, чистый синий цвет обозначается как **#0000FF**, чистый красный цвет — как **#FF0000** и т.д.

#RGB

Это более краткая форма описанного выше обозначения цвета шестью шестнадцатеричными цифрами. В этой форме каждая цифра дублируется для получения в конечном итоге эквивалентного значения из шести цифр. Таким образом, шестнадцатеричное значение цвета **#F8C** становится значением цвета **#FF88CC**.

rgb(rrr,ggg,bbb)

В этой форме значения цвета допускается указывать в пределах **0-255**, где разрешаются только целые числа. И совсем не случайно, что эти пределы являются десятичным эквивалентом пределов **00-FF** в шестнадцатеричной форме. Так, чистый зеленый цвет обозначается в данной форме как **rgb(0,255,0)**, а белый цвет — как **rgb(255,255,255)**.

rgb(rrr.rr%,ggg.gg%,bbb.bb%)

В этой форме значения цвета RGB допускается указывать в процентах от **0%** до **100%**, где разрешаются десятичные числа (например, **75.5%**). Так, значение черного цвета обозначается как **rgb(0%,0%,0%)**, тогда как значение чистого синего цвета — как **rgb(0%,0%,100%)**.

hsl(hhh.hh,sss.ss%,lll.ll%)

В этой форме допускается обозначать цвет по его оттенку (углу цветового тона на цветовом круге), насыщенности и яркости (т.е. по цветовой модели HSL — "оттенок-насыщен-

ность–яркость"). Оттенок всегда указывается безразмерным числом в пределах от **0** до **360**, а насыщенность и яркость — в процентах. Оттенки **0** и **360** равнозначны и оба относятся к красному цвету. Допускается указывать оттенки свыше **360**, но они все равно приводятся к пределам **0–360**. Таким образом, обозначение оттенка **454** равнозначно установке оттенка **94**. Любое значение цвета HSL, независимо от его оттенка, будет воспроизведено как оттенок серого, если значение насыщенности равно **0%**. Конкретный оттенок серого зависит от значения яркости. А любое значение цвета HSL, независимо от его оттенка, будет воспроизведено сплошным черным цветом, если значение яркости равно **0%**, или сплошным белым цветом, если значение яркости равно **100%**. Обычное для большинства цветов значение яркости составляет **50%**.

rgba(rrr,ggg,bbb,a.aa)
rgba(rrr.rr%,ggg.gg%,bbb.bb%,a.aa)
hsla(hhh.hh,sss.ss%,lll.ll%,a.aa)

Эта форма служит расширением предыдущих форм, включая в себя значение альфа-канала (т.е. непрозрачности цвета). Значение непрозрачности должно быть указано вещественным числом в пределах от **0** до **1** включительно, но только не в процентах. Таким образом, обозначения **rgba(0,0,255,0.5)**, **rgba(0,0,100%,0.5)** и **hsla(0,100%,50%,0.5)** равнозначны и определяют один и тот же полупрозрачный красный цвет. А шестнадцатеричное обозначение для цвета RGBA не предусмотрено.

<ключевое слово>

На основании исходных цветов по стандарту VGA в Windows для распознавания названий отдельных цветов определены 17 следующих ключевых слов: `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `orange`, `purple`, `red`, `silver`, `teal`, `white` и `yellow`. Как правило, цвета распознаются в браузерах и по другим ключевым словам, например цвета 147 X11, описанные в разделе 4.3 спецификации цвето-

вого модуля W3C CSS3 Color Module (по адресу http://www.w3.org/TR/css3-color). В этом цветовом модуле из списка основных распознаваемых ключевых слов исключено слово orange. Тем не менее оно присутствует в списке цветов X11 и поддерживается во всех известных браузерах по исторически сложившимся причинам.

currentColor

Этим специальным ключевым словом обозначается текущее вычисляемое значение свойства color отдельного элемента. Это означает, что с помощью объявления background-color: currentColor можно задать такой же цвет заднего плана, как и цвет переднего плана, что, в общем, не рекомендуется делать. Применительно к свойству color это равнозначно объявлению border: 1px solid currentColor. Этим ключевым словом удобно пользоваться для установки и отмены цвета границы, составляя документы по модели DOM с помощью сценариев.

transparent

Это специальное ключевое слово служит сокращенной формой для обозначения прозрачного цвета **rgba(0,0,0,0)**, которое автоматически вычисляется всякий раз, когда употребляется ключевое слово transparent.

Числовые значения

Числовое значение обозначается как положительное или отрицательное, если это разрешается. Числа могут быть вещественными и представленными типом данных <number> или целыми и представленными типом данных <integer>. Они могут также иметь ограниченный диапазон допустимых значений, как, например, значения цвета в пределах **0–255**. Чаще всего диапазон чисел ограничивается только неотрицательными числами, но иногда такие числа могут быть представлены как <non-negative number> или <non-negative integer>, т.е. как неотрицательное вещественное или целое число.

Значения в процентах

Значение в процентах обозначается как число типа `<number>` со знаком процента (`%`). Между числом и знаком процента не должно быть никаких пробелов. Значение в процентах всегда вычисляется относительно какого-то другого значения. Например, в объявлении `font-size: 120%;` для элемента задается шрифт размером **`120%`** от вычисляемого значения свойства `font-size` в его родительском элементе. В некоторых свойствах может допускаться употребление только неотрицательных значений.

Значения длины

Значение длины обозначается как положительное или отрицательное число, если это разрешено, после которого сразу же следует двухбуквенное сокращение, обозначающее используемые единицы измерения длины. Между числом и обозначением единицы измерения не должно быть никаких пробелов. При обозначении нулевого (`0`) значения длины указывать единицы измерения не нужно. Единицы измерения длины разделяются на следующие два вида: *абсолютные единицы измерения*, которые теоретически измеряются одинаково, и *относительные единицы измерения*, которые измеряются относительно чего-то другого.

Абсолютные единицы измерения длины

Ниже перечислены абсолютные единицы измерения длины, употребляемые в CSS.

Дюймы (`in`)

Это типичные единицы измерения длины, указываемые на измерительных линейках в США. Преобразование дюймов в единицы, применяемые для измерения длины на экранах мониторов и прочих устройств отображения, обычно выполняется в

лучшем случае приблизительно, поскольку отображение информации на рабочих участках экранов во многих системах никак не связано с такими "настоящими" единицами измерения, как дюймы. Поэтому пользоваться дюймами следует крайне осторожно, разрабатывая документы, предназначенные для вывода на экран.

***Сантиметры* (cm)**

Сантиметры обычно указываются на измерительных линейках в большинстве стран мира. В каждом дюйме насчитывается **2,54** сантиметра, а **1** сантиметр равен **0,394** дюйма. В отношении применения сантиметров в качестве единиц измерения длины отдельных элементов документов на экранах мониторов действуют те же самые предостережения, что и для дюймов.

***Миллиметры* (mm)**

В одном сантиметре насчитывается **10** миллиметров, а в дюйме — **25,4** миллиметра, и **1** миллиметр равен **0,0394** дюйма. Но в отношении миллиметров следует иметь в виду упомянутые выше предупреждения по поводу измерения длины отдельных элементов документов на экранах мониторов.

***Пункты* (pt)**

Пункты являются стандартными в полиграфии единицами измерения длины на печатных и наборных машинах и в программах обработки и верстки документов. В соответствии с современным определением в одном дюйме насчитывается **72** пункта. Следовательно, заглавные буквы текста, набранные кеглем (т.е. размером шрифта) **12** пунктов, должны быть высотой в одну шестую дюйма. Например, объявление `p {font-size: 18pt;}` равнозначно объявлению `p {fontsize: 0.25in;}` при надлежащем преобразовании единиц измерения длины в среде отображения информации, как пояснялось выше.

***Цицеро* (pc)**

Это еще одна единица измерения, применяемая в полиграфии. Один цицеро равнозначно **12** пунктам, т.е. в одном дюйме насчитывается **6** цицеро. Заглавные буквы текста, набранные

кеглем в 1 цицеро, должны быть высотой в одну шестую дюйма. Например, в объявлении `p {font-size: 1.5pc;}` задается такой же размер шрифта для набора текста, как и в объявлениях, приведенных выше при пояснении пунктов. Применяя цицеро, следует также принимать во внимание упомянутые выше предостережения.

Относительные единицы измерения длины

Ниже перечислены относительные единицы измерения длины, используемые в CSS.

***Высота шрифта* (em)**

Эта единица измерения обозначает высоту кегельной площадки заданного шрифта. Высота шрифта в CSS равнозначна знакоместу для заданного шрифта, а, по существу, — вычисляемому значению свойства `font-size`. В единицах измерения высоты шрифта можно указывать относительные размеры шрифтов. Например, **`1.2em`** означает то же самое, что и **`120%`** при указании шрифта.

***Высота шрифта корневого элемента* (rem)**

Эта единица измерения обозначает высоту шрифта корневого элемента (т.е. элемента разметки `html` в HTML- и XHTML-документах). А в остальном она подобна единице измерения **`em`**.

***X-height* (ex)**

Эта единица измерения обозначает x-высоту (т.е. размер строчных букв) шрифта. Но в определение подавляющего большинства шрифтов x-высота не входит, и поэтому во многих браузерах применяется следующее весьма приблизительное соотношение единиц измерения высоты шрифта: **`1ex`** равно **`0.5em`**. Исключением из этого правила является версия 5 браузера Internet Explorer для компьютеров Macintosh, в которой предпринимается попытка определить фактическую x-высо-

ту шрифта путем внутреннего растрирования очень большой буквы **x** и подсчета пикселей!

***Ширина НУЛЯ* (ch)**

Эта единица измерения обозначает ширину одного нуля (**Unicode +0300**) в текущей гарнитуре шрифта и в его размере.

***Пиксели* (px)**

Пиксель представляет собой, как правило, прямоугольный и реже квадратный элемент изображения на экране, но в CSS пиксели определяются более абстрактно. В точки зрения CSS размер пикселя определяется таким образом, чтобы на один дюйм приходилось **96** пикселей. Многие пользовательские агенты пренебрегают этим определением в пользу простого обращения к пикселям на экране монитора. При увеличении, уменьшении изображения или печати страниц в силу вступают масштабные коэффициенты, где элемент шириной **100px** может быть воспроизведен на устройстве вывода больше 100 точек по ширине.

***Единица измерения ширины области просмотра* (vw)**

Эта единица измерения вычисляется относительно ширины области просмотра, которая делится на **100**. Так, если ширина области просмотра составляет **937** пикселей, то **1vw** равно **9.37px**. Если же ширина области просмотра изменяется, например, в результате того, что окно браузера делается шире или уже, то вместе с ней изменяется и величина **vw**.

***Единица измерения высоты области просмотра* (vh)**

Эта единица измерения вычисляется относительно высоты области просмотра, которая делится на **100**. Так, если ширина области просмотра составляет **650** пикселей, то **1vh** равно **6.5px**. Если же высота области просмотра изменяется, например, в результате того, что окно браузера делается длиннее или короче, то вместе с ней изменяется и величина **vh**.

***Минимальная единица измерения области просмотра* (vm)**

Эта единица измерения составляет **1/100** часть ширины или высоты области просмотра в зависимости от того, какая из этих

величин *меньше*. Так, если имеется область просмотра шириной **937** пикселей и высотой **650** пикселей, то **1vm** равно **6.5px**.

URI

Значение URI типа `<uri>` является ссылкой на файл, содержащий, например, графическое изображение или другую таблицу стилей. URI определяется в CSS относительно таблицы стилей, которая его содержит. Сокращение "URI" означает "Uniform Resource Identifier" (Универсальный идентификатор ресурса) и в последнее время чаще употребляется вместо URL. (Формально URL являются подмножеством URI.) Тем не менее в спецификации CSS, которая была впервые определена, когда URI еще обозначались как URL, это означает, что ссылки на URI будут фактически появляться в форме `url(<uri>)`. Забавно!

Углы

Углы выражаются в форме `<angle>`, где после числа типа `<number>` сразу же следует единица измерения угла. Существуют следующие разновидности единиц измерения углов: градусы (`deg`), грады (`grad`), радианы (`rad`) и обороты (`turn`). Например, прямой угол можно объявить одним из следующих способов: **90deg**, **100grad**, **1,571rad** или **0,25turn**, и в каждом случае указанные значения прямого угла преобразуются в градусы в пределах от **0** до **360**. То же самое относится и к отрицательным значениям углов, которые вполне допустимы. Так, величины **-90deg** и **270deg** обозначают один и тот же прямой угол.

Время

Величина времени типа `<time>` выражается в виде неотрицательного числа типа `<number>`, после которого сразу же следует

единица измерения времени. Существуют следующие разновидности единиц измерения времени: секунды (`s`) и миллисекунды (`ms`). Величины времени употребляются в акустических стилях, которые не нашли широкого признания, а также в намного лучше поддерживаемых монтажных переходах и в анимации.

Частоты

Величина частоты типа `<frequency>` выражается в виде неотрицательного числа типа `<number>`, за которым сразу же следует единица измерения частоты. Существуют следующие разновидности единиц измерения частоты: герцы (`Hz`) и килогерцы (`kHz`). Они обозначаются без учета регистра букв, и поэтому величины **`6kHz`** и **`6khz`** равнозначны. На момент написания данной книги величины частоты использовались в акустических стилях, которые не нашли широкой поддержки.

Символьные строки

Символьная строка типа `<string>` состоит из последовательности символов, заключаемых в одинарные или двойные кавычки. Если в состав символьной строки входит та же самая кавычка, в которую она заключается, то такая кавычка должна быть экранирована, как, например, в следующих строках: `'That \'s amazing!'` или `"Deploy the \"scare quotes\" at once!"`. Если же в символьной строке необходимо указать знак новой строки, его следует обозначить как **`\A`**, что в Уникоде означает кодовую точку для знака новой строки. Любой символ в Уникоде может быть представлен с помощью экранированной ссылки на кодовую точку. Так, левая закругленная двойная кавычка (“) может быть представлена в Уникоде как **`\201C`**. Если символьная строка все же содержит знак перевода строки из соображений удобочитаемости, этот знак должен быть экранирован, а при обработке строки — удален.

Глава 3

Селекторы

Обычные селекторы

Ниже перечислены обычные селекторы и коротко описано их назначение.

Универсальный селектор

Образец:

*

Описание:

Этот селектор обеспечивает совпадение с именем любого элемента на языке документа. Если правило не предусматривает явного указания селектора, то автоматически выводится универсальный селектор.

Примеры:

```
* {color: red;}
div * p {color: blue;}
```

Селектор по типу

Образец:

```
element1
```

Описание:

Этот селектор обеспечивает совпадение с именем указанного элемента на языке документа. Совпадение происходит с каждым экземпляром имени искомого элемента. (В спецификации CSS1 такие селекторы назывались селекторами элементов.)

Примеры:

```
body {background: #FFF;}
p {font-size: 1em;}
```

Селектор порожденных элементов

Образец:

```
element1 element2
```

Описание:

Этот селектор позволяет выбрать элемент по его состоянию как порожденного от другого элемента. Совпавший элемент может быть потомком, внуком, правнуком и так далее родительского элемента. (В спецификации CSS1 такие селекторы назывались контекстными селекторами).

Примеры:

```
body h1 {font-size: 200%;}
table tr td div ul li {color: purple;}
```

Селектор потомков

Образец:

```
element1 > element2
```

Описание:

Этот тип селектора служит для совпадения с элементом по его состоянию как потомка другого элемента. Он более строгий, чем селектор порожденных элементов, поскольку обеспечивает совпадение только с потомком.

Примеры:

```
div > p {color: cyan;}
ul > li {font-weight: bold;}
```

Селектор смежных родственных элементов

Образец:

```
element1 + element2
```

Описание:

Этот селектор позволяет выбрать следующий смежный элемент, родственный другому элементу. (У родственных элементов, как подразумевает их название, имеется общий родительский элемент.) Любой текст, расположенный между двумя родственными элементами, игнорируется, а во внимание принимаются только сами элементы и их положение в дереве документа.

Примеры:

```
table + p {margin-top: 2.5em;}
h1 + * {margin-top: 0;}
```

Селектор следующих родственных элементов

Образец:

```
element1 ~ element2
```

Описание:

Этот селектор позволяет выбрать элемент, родственный другому элементу и следующий после него в дереве документа. Любой текст, расположенный между двумя родственными элементами, игнорируется, а во внимание принимаются только сами элементы и их положение в дереве документа.

Примеры:

```
h1 ~ h2 {margin-top: 2.5em;}
div#navlinks ~ div {margin-top: 0;}
```

Селектор по классу

Образец:

```
element1.classname
element1.classname1.classname2
```

Описание:

В тех языках, в которых допускается запись через точку (например, в HTML, XHTML, SVG и MathML), селектор по классу позволяет выбрать элементы с атрибутом класса, содержащим конкретные значения. Наименование значения класса должно быть указано сразу же после точки. Несколько значений класса можно связывать в цепочку, хотя поддержка такой возможности вызывает затруднения в браузере Internet Explorer до версии 7. Если же имя перед точкой отсутствует, селектор по классу обеспечивает совпадение со всеми элементами, содержащими значения данного класса.

Примеры:

```
p.urgent {color: red;}
a.external {font-style: italic;}
.example {background: olive;}
.note.caution {background: yellow;}
```

Примечание:

В браузере Internet Explorer до версии 7 не поддерживается синтаксис связывания в цепочку значений, указываемых в селекторе по классу, хотя при разметке допускается указывать несколько слов в значениях класса.

Селектор по идентификатору

Образец:

```
element1#idname
```

Описание:

В тех языках, в которых допускается запись через знак решетки (например, в HTML или XHTML), селектор по идентификатору позволяет выбрать элементы, содержащие идентификатор с конкретными значениями. Наименование значения идентификатора должно быть указано сразу же после знака решетки (#). Если же имя элемента перед знаком решетки отсутствует, селектор по идентификатору обеспечивает совпадение со всеми элементами, содержащими значение этого идентификатора.

Примеры:

```
h1#page-title {font-size: 250%;}
body#home {background: silver;}
#example {background: lime;}
```

Простой селектор по атрибутам

Образец:

```
element1[attr]
```

Описание:

Этот селектор позволяет выбрать любой элемент по наличию в нем заданного атрибута, независимо от значения этого атрибута.

Примеры:

```
a[rel] {border-bottom: 3px double gray;}
p[class] {border: 1px dotted silver;}
```

Селектор по точным значениям атрибутов

Образец:

```
element1[attr="value"]
```

Описание:

Этот селектор позволяет выбрать любой элемент по точному и полному значению атрибута.

Примеры:

```
a[rel="Start"] {font-weight: bold;}
p[class="urgent"] {color: red;}
```

Селектор по частичным значениям атрибутов

Образец:

```
element1[attr~="value"]
```

Описание:

Этот селектор позволяет выбрать любой элемент по части значения атрибута, разделяемой пробелами. Следует, однако, иметь в виду, что выражения `[class~="value"]` и `.value` равнозначны, как пояснялось выше.

Примеры:

```
a[rel~="friend"] {text-transform: uppercase;}
p[class~="warning"] {background: yellow;}
```

Селектор по начальным подстрокам в значениях атрибутов

Образец:

```
element1[attr^="substring"]
```

Описание:

Этот селектор позволяет выбрать любой элемент по любой подстроке в самом начале значения атрибута.

Примеры:

```
a[href^="/blog"] {text-transform: uppercase;}
p[class^="test-"] {background: yellow;}
```

Селектор по конечным подстрокам в значениях атрибутов

Образец:

```
element1[attr$="substring"]
```

Описание:

Этот селектор позволяет выбрать любой элемент по любой подстроке в самом конце значения атрибута.

Пример:

```
a[href$=".pdf"] {font-style: italic;}
```

Селектор по произвольным подстрокам в значениях атрибутов

Образец:

```
element1[attr*="substring"]
```

Описание:

Этот селектор позволяет выбрать любой элемент по любой подстроке, обнаруживаемой на любом месте в значении атрибута.

Примеры:

```
a[href*="oreilly.com"] {font-weight: bold;}
div [class*="port"] {border: 1px solid red;}
```

Селектор по языковым атрибутам

Образец:

```
element1[lang|="lc"]
```

Описание:

Этот селектор позволяет выбрать любой элемент по атрибуту `lang`, значением которого является разделяемый дефисами список, начиная со значения, указанного в селекторе.

Пример:

```
html[lang|="tr"] {color: red;}
```

Структурные псевдоклассы

Строго говоря, все псевдоклассы, как и все селекторы, являются структурными. В конечном счете все они так или иначе зависят от структуры документа. Рассматриваемые здесь псевдоклассы отличаются тем, что они, по существу, действуют по

шаблонам, обнаруживаемым в структуре документа, выбирая, например, текст через абзац или элементы, являющиеся последними потомками своего родительского элемента.

:empty

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементами, у которых отсутствуют порожденные узлы, т.е. отсутствуют порожденные элементы *или* узлы содержимого, которые определяются как любой текст, пробелы, ссылка на объект или узлы CDATA. Так, элемент   *не* является пустым и не окажется таковым, даже если заменить в нем пробел знаком новой строки. Следует, однако, иметь в виду, что этот псевдокласс *не* применяется к пустым элементам наподобие `br`, `img`, `input` и т.д.

Примеры:

```
p:empty {padding: 1em; background: red;}
li:empty {display: none;}
```

:first-child

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементом, если он оказывается первым потомком другого элемента. Так, выражение `div:first-child` позволяет выбрать любой

элемент div, который оказывается первым потомком другого элемента, но *не* первым потомком любого элемента div.

Примеры:

```
td:first-child {border-left: 1px solid;}
p:first-child {text-indent: 0; margin-top: 2em;}
```

:first-of-type

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементом, если он оказывается первым потомком другого элемента для своего типа. Так, объявление div:first-of-type позволяет выбрать любой элемент div, который оказывается первым потомком типа div другого элемента.

Примеры:

```
td:first-of-type {border-left: 1px dotted;}
h2:first-of-type {color: fuchsia;}
```

:lang

Применяется:

К любому элементу с соответствующей информацией о языковой кодировке.

Описание:

Этот псевдокласс обеспечивает совпадение с элементами по их языковой кодировке. Подобная информация должна находиться в самих элементах или быть иным образом связанной с документом, поскольку ее нельзя назначить из CSS. Обработка

псевдокласса :lang выполняется таким же образом, как и селекторов по языковым атрибутам |=. Например, в HTML-документе язык элемента определяется его атрибутом lang. Если такой атрибут отсутствует в документе, то язык элемента определяется атрибутом lang его ближайшего родственного элемента, а если и у него такой атрибут отсутствует, то полем ответного HTTP-заголовка Content-Language (или соответствующим атрибутом meta http-equiv) для документа.

Примеры:

```
html:lang(en) {background: silver;}
*:lang(fr) {quotes: '&#171; ' ' &#187;';}
```

:last-child

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементом, если он оказывается последним потомком другого элемента. Так, объявление div:last-child позволяет выбрать любой элемент div, который оказывается последним потомком другого элемента, но *не* последним потомком любого элемента div.

Примеры:

```
td:last-child {border-right: 1px solid;}
p:last-child {margin-bottom: 2em;}
```

:last-of-type

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементом, если он оказывается последним потомком другого элемента для своего типа. Так, объявление `div:last-of-type` позволяет выбрать любой элемент `div`, который оказывается последним потомком типа `div` другого элемента.

Примеры:

```
td:last-of-type {border-right: 1px dotted;}
h2:last-of-type {color: fuchsia;}
```

:nth-child(an+b)

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с каждым *n*-м потомком по шаблону выбора, определяемому по формуле **an+*b***, где **a** и ***b*** — целые числа типа `<integer>`, а **n** — бесконечный ряд целых чисел, отсчитываемый *вперед* от *первого* потомка. Следовательно, чтобы был выбран каждый четвертый потомок элемента разметки `body`, начиная с первого потомка, достаточно составить следующий селектор: `body > *:nth-child(4n+1)`. В итоге будут выбраны первый, пятый, девятый, четырнадцатый и так далее потомки элемента разметки `body`. Если же требуется выбрать четвертый, восьмой, двенадцатый и так далее потомки этого элемента, то достаточно составить такой селектор: `body > *:nthchild(4n)`. Имеется также возможность указывать отрицательное значение параметра ***b***, и тогда селектор `body > *:nth-child(4n-1)` выберет третий, седьмой, одиннадцатый, пятнадцатый и так далее потомки элемента разметки `body`.

Вместо формулы **an+b** допускается употреблять ключевые слова **even** и **odd**. Эти ключевые слова равнозначны **2n** и **2n+1** соответственно.

Примеры:

```
*:nth-child(4n+1) {font-weight: bold;}
tbody tr:nth-child(odd) {background-color: #EEF;}
```

:nth-last-child(an+b)

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с каждым *n*-м потомком по шаблону выбора, определяемому по формуле **an+b**, где **a** и **b** — целые числа типа <integer>, а **n** — бесконечный ряд целых чисел, отсчитываемый *назад* от *последнего* потомка. Следовательно, чтобы был выбран каждый четвертый потомок элемента разметки body, начиная с последнего потомка, достаточно составить следующий селектор: `body > *:nth-last-child(4n+1)`. Этот псевдокласс, по существу, является зеркальным отражением псевдокласса `:nth-child`.

Вместо формулы **an+b** допускается употреблять ключевые слова **even** и **odd**. Эти ключевые слова равнозначны **2n** и **2n+1** соответственно.

Примеры:

```
*:nth-last-child(4n+1) {font-weight: bold;}
tbody tr:nth-last-child(odd) {background-color: #EEF;}
```

:nth-last-of-type(an+b)

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с каждым *n*-м потомком того же самого типа, что и указанный элемент, по шаблону выбора, определяемому по формуле **an+*b***, где ***a*** и ***b*** — целые числа типа <integer>, а **n** — бесконечный ряд целых чисел, отсчитываемый *назад* от *последнего* элемента *такого типа*. Следовательно, чтобы выбрать каждый третий от конца абзац (p), являющийся потомком элемента разметки body, начиная с первого такого абзаца, достаточно составить следующий селектор: `body > p:nth-last-of-type(3n+1)`. Этот критерий выбора остается в силе даже в том случае, если между различными абзацами встречаются другие элементы, в том числе списки, таблицы и пр.

Вместо формулы **an+*b*** допускается использовать ключевые слова **even** и **odd**. Эти ключевые слова равнозначны **2n** и **2n+1** соответственно.

Примеры:

```
td:nth-last-of-type(even) {background-color: #FCC;}
img:nth-last-of-type(3n) {float: left; border: 2px solid;}
```

:nth-of-type(an+b)

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с каждым *n*-м потомком того же самого типа, что и указанный элемент, по шаблону выбора, определяемому по формуле **an+*b***, где ***a*** и ***b*** — целые числа типа <integer>, а **n** — бесконечный ряд целых чисел, отсчитываемый *вперед* от *первого* элемента *такого типа*. Следовательно, чтобы выбрать каждый третий абзац

(p), являющийся потомком элемента разметки body, начиная с первого такого абзаца, достаточно составить следующий селектор: `body > p:nth-oftype(3n+1)`. В итоге будут выбраны первый, четвертый, седьмой, десятый и так далее абзацы, являющиеся потомками элемента разметки body. Этот критерий выбора остается в силе даже в том случае, если между различными абзацами встречаются другие элементы, в том числе списки, таблицы и пр.

Вместо формулы **an+b** допускается использовать ключевые слова **even** и **odd**. Эти ключевые слова равнозначны **2n** и **2n+1** соответственно.

Примеры:

```
td:nth-of-type(even) {background-color: #FCC;}
img:nth-of-type(3n) {float: right; margin-left: 1em;}
```

:only-child

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементом, являющимся единственным потомком своего родительского элемента. Такой псевдокласс чаще всего применяется для удаления границы из любого привязываемого по ссылке изображения при условии, что оно является единственным элементом в ссылке. Следует иметь в виду, что элемент может быть выбран с помощью псевдокласса `:only-child` даже в том случае, если у него имеются свои потомки. Но сам элемент должен быть *единственным* потомком своего родительского элемента.

Примеры:

```
a img:only-child {border: 0;}
table div:only-child {margin: 5px;}
```

:only-of-type

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементом того же самого типа, являющимся единственным потомком своего родительского элемента. Следует иметь в виду, что элемент может быть выбран с помощью псевдокласса `:only-of-type` даже в том случае, если у него имеются свои потомки (например, элементы `div` внутри элемента разметки `div`). Но сам элемент должен быть *единственным* потомком того же самого типа для своего родительского элемента.

Примеры:

```
p em:only-of-type {font-weight: bold;}
section article:only-of-type {margin: 2em 0 3em;}
```

:root

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с корневым элементом документа, которым в HTML- и XHTML-документах всегда является элемент разметки `html`. А в форматах XML-разметки корневой элемент может иметь любое имя, и поэтому

для его выбора требуется обобщенный селектор корневых элементов.

Примеры:

```
:root {font: medium serif;}
:root > * {margin: 1.5em 0;}
```

Псевдоклассы отрицания

Имеется лишь один псевдокласс, предназначенный для отрицания. Но он настолько особенный, что заслуживает выделения в отдельную категорию.

`:not(e)`

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с каждым элементом, который *не* описывается простым селектором ***e***. Это дает возможность выбрать, например, каждый элемент, который не является абзацем, с помощью следующего селектора: `*:not(p)`. Более полезное применение отрицание может найти в контексте селекторов порожденных элементов. Примером тому может служить выбор из таблицы каждого элемента, который не является ячейкой данных, с помощью такого селектора: `*:not(td)`. Еще одним примером отрицания может служить выбор каждого элемента без идентификатора `"search"` с помощью следующего селектора: `[id]:not([id="search"])`.

Из приведенного выше простого определения простого селектора ***e*** имеется следующее исключение: он не может быть псевдоклассом отрицания. Это означает, что нельзя составить

следующий селектор: `:not(:not(div))`. Но это не такая уж и большая потеря, поскольку такое двойное отрицание равнозначно выбору элемента `div`.

В связи с тем что выражение `:not()` является псевдоклассом, его можно связывать в цепочку с другими псевдоклассами, а также с его собственными экземплярами. Например, чтобы выбрать любой элемент, на котором установлен фокус ввода и который не является элементом разметки `a`, достаточно воспользоваться следующим селектором: `*:focus:not(a)`. А для того чтобы выбрать любой элемент, который не является ни абзацем, ни элементом разметки `div`, можно составить такой селектор: `*:not(p):not(div)`.

На момент написания данной книги ограничение на простой селектор означало, что в выражениях `:not()` не допускалось употреблять селекторы групп и порожденных элементов. Это ограничение, вероятнее всего, будет послаблено в последующих версиях модуля CSS Selectors.

Примеры:

```
ul *:not(li) {text-indent: 2em;}
fieldset *:not([type="checkbox"]):not([type="radio"])
    {margin: 0 1em;}
```

Псевдоклассы взаимодействия

Все перечисленные ниже псевдоклассы имеют отношение к взаимодействию пользователя с документом. Такое взаимодействие происходит независимо от стилевого оформления разных состояний ссылок, выделения элемента, являющегося целевым для идентификатора фрагмента, или от стилевого оформления элементов формы по состоянию их доступности или недоступности.

:active

Применяется:

К активизируемому элементу.

Описание:

Этот псевдокласс применяется к элементу во время его активизации. Наиболее характерным примером тому служит щелчок кнопкой мыши на гиперссылке в HTML-документе. Эта ссылка остается активной до тех пор, пока кнопка мыши нажата. Имеются и другие способы активизации элементов. И хотя теоретически активизированы могут быть и другие элементы, в CSS они не определены.

Примеры:

```
a:active {color: red;}
*:active {background: blue;}
```

:checked

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с любым элементом пользовательского интерфейса, который не "включен", например, с установленным флажком или выбранной кнопкой-переключателем.

Примеры:

```
input:checked {outline: 3px solid rgba(127,127,127,0.5);}
input[type="checkbox"]:checked {box-shadow: red 0 0 5px;}
```

:disabled

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементами пользовательского интерфейса, которые не в состоянии принять вводимые пользователем данные из-за языковых атрибутов или других непредставительных средств, например разметки `<input type="text" disabled>` недоступного поля ввода текста в HTML5. Следует, однако, иметь в виду, что псевдокласс `:disabled` *не* применяется, если элемент разметки `input` просто удален из области просмотра с помощью таких свойств, как `position` и `display`.

Пример:

```
input:disabled {opacity: 0.5;}
```

:enabled

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементами пользовательского интерфейса, которые способны принимать вводимые пользователем данные и могут быть установлены в активное или неактивное состояние средствами самого языка разметки. К их числу относятся любые формы элементов разметки ввода пользовательских данных в (X)HTML, кроме гиперссылок.

Пример:

```
input:enabled {background: #FCC;}
```

:focus

Применяется:

К элементу, находящемуся в фокусе ввода.

Описание:

Этот псевдокласс применяется к элементу в тот момент, когда он находится в фокусе ввода. Примером тому служит поле ввода текста в HTML-документе, на котором находится указатель, и когда пользователь начинает набирать текст, он вводится в этом поле. В фокусе ввода могут находиться и другие элементы (например, гиперссылки), но в CSS точно не определено, какие именно элементы могут иметь фокус ввода.

Примеры:

```
a:focus {outline: 1px dotted red;}
input:focus {background: yellow;}
```

Примечание:

Поддержка псевдокласса `:focus` в браузере Internet Explorer распространяется только на гиперссылки, но не на элементы управления в формах.

:hover

Применяется:

К элементу, находящемуся в состоянии, когда на него наведен курсор.

Описание:

Этот псевдокласс применяется к элементу в тот момент, когда на него *наведен* курсор, т.е. пользователь указывает на элемент, но не активизирует его. Наиболее характерным тому примером служит перемещение указателя мыши по гиперссылке в HTML-документе. Теоретически указатель может быть наведен и на другие элементы, хотя в CSS они точно не определены.

Примеры:

```
a[href]:hover {text-decoration: underline;}
p:hover {background: yellow;}
```

Примечание:

Поддержка псевдокласса `:hover` в браузере Internet Explorer до версии 7 распространяется только на гиперссылки.

`:link`

Применяется:

К гиперссылке или другому не посещенному ресурсу.

Описание:

Этот псевдокласс применяется к ссылке на URI не посещенного ресурса, т.е. к ссылке на такой URI, который отсутствует в предыстории посещений пользовательского агента. Такое состояние совершенно противоположно состоянию для применения псевдокласса `:visited`.

Примеры:

```
a:link {color: blue;}
*:link {text-decoration: underline;}
```

:target

Применяется:

К любому элементу.

Описание:

Этот псевдокласс обеспечивает совпадение с элементом, который сам сопоставлен с частью идентификатора фрагмента URI, используемого для доступа к веб-странице. Следовательно, ссылка `http://www.w3.org/TR/css3-selectors/#target-pseudo` будет сопоставлена по шаблону `:target`, чтобы применить объявленные селекторы к любому элементу с идентификатором `target-pseudo`. Если бы этот элемент был абзацем, он был бы сопоставлен также по шаблону `p:target`.

Пример:

```
:target {background: #EE0;}
```

:visited

Применяется:

К гиперссылке или другому посещенному ресурсу.

Описание:

Этот псевдокласс применяется к ссылке на URI посещенного ресурса, т.е. к ссылке на такой URI, который присутствует в предыстории посещений пользовательского агента. Такое состояние совершенно противоположно состоянию для применения псевдокласса `:link`.

Примеры:

```
a:visited {color: purple;}
*:visited {color: gray;}
```

Псевдоэлементы

В спецификациях CSS1 и CSS2 имена псевдоэлементов предварялись одним двоеточием, как, впрочем, и имена псевдоклассов. А в спецификации CSS3 имена псевдоэлементов предваряются двумя двоеточиями, чтобы как-то отличать их от псевдоклассов. По причинам исторического характера в браузерах поддерживаются псевдоэлементы как с одним, так и с двумя двоеточиями, хотя все же рекомендуется использовать два двоеточия.

::after

Формирует:

Псевдоэлемент, содержащий сформированное содержимое, размещаемое после содержимого элемента разметки.

Описание:

Этот псевдоэлемент позволяет ввести сформированное содержимое в конце содержимого элемента разметки. По умолчанию этот псевдоэлемент является внутристрочным, но может быть изменен с помощью свойства `display`.

Примеры:

```
a.external:after {content: " " url(/icons/globe.gif);}
p:after {content: " | ";}
```

::before

Формирует:

Псевдоэлемент, содержащий сформированное содержимое, размещаемое до содержимого элемента разметки.

Описание:

Этот псевдоэлемент позволяет ввести сформированное содержимое в начале содержимого элемента разметки. По умолчанию этот псевдоэлемент является внутристрочным, но может быть изменен с помощью свойства `display`.

Примеры:

```
a[href]:before {content: "[LINK] ";)
p:before {content: attr(class);}
a[rel|="met"]:after {content: " *";}
```

::first-letter

Формирует:

Псевдоэлемент, содержащий первую букву элемента.

Описание:

Этот псевдоэлемент служит для стилевого оформления первой буквы элемента. Вместе с первой буквой соответствующим стилем должен быть оформлен и любой начальный знак препинания. В некоторых языках имеются сочетания букв, которые должны интерпретироваться как один символ, и пользовательский агент может применить к обеим буквам стиль, заданный для оформления первой буквы. До версии CSS2.1 псевдоэлемент `::first-letter` можно было присоединять только к блочным элементам. А начиная с версии CSS2.1, область действия этого псевдоэлемента расширена и включает блоки, элементы списков, ячейки таблицы, надписи на таблицах и внутристрочные элементы. К первой букве элемента может быть применен ограниченный ряд свойств.

Примеры:

```
h1:first-letter {font-size: 166%;}
p:first-letter {text-decoration: underline;}
```

::first-line

Формирует:

Псевдоэлемент, содержащий первую отформатированную строку элемента.

Описание:

Этот псевдоэлемент служит для стилевого оформления первой текстовой строки в элементе, независимо от количества слов в этой строке. Псевдоэлемент `::first-line` можно присоединять только к блочным элементам. К первой строке элемента может быть применен ограниченный ряд свойств.

Пример:

```
p.lead:first-line {font-weight: bold;}
```

Мультимедийные запросы

С помощью мультимедийных запросов можно определить среду распространения информации, в которой браузер применяет заданную таблицу стилей. В прошлом для этой цели служили типы носителя информации, которые задавались в атрибуте `media` элементов разметки `link` или в дескрипторе носителя информации в объявлениях `@import`. Мультимедийные запросы расширяют этот принцип, позволяя выбирать таблицы стилей, исходя из возможностей конкретного типа носителя информации.

Основные понятия

Размещение мультимедийных запросов окажется очень знакомым для тех, кому приходилось раньше задавать тип носителя информации. Ниже демонстрируются два способа применения внешней таблицы стилей для воспроизведения документа на цветном печатающем устройстве.

```
<link href="print-color.css" type="text/css"
media="print and (color)" rel="stylesheet">

@import url(print-color.css) print and (color);
```

Мультимедийный запрос можно применять везде, где допускается задавать тип носителя информации. Это означает, что несколько подобных запросов можно перечислить в списке через запятую следующим образом:

```
<link href="print-color.css" type="text/css"
    media="print and (color), projection and (color)"
    rel="stylesheet">

@import url(print-color.css) print and (color),
    projection and (color);
```

Соответствующая таблица стилей применяется всякий раз, когда один из мультимедийных запросов вычисляется как "истинный". Так, в приведенном выше объявлении `@import` таблица стилей `print-color.css` применяется в том случае, если документ нужно воспроизвести на цветном печатающем устройстве или в среде проецирования цветных изображений. А если предполагается печать на черно-белом печатающем устройстве, то оба запроса вычисляются как "ложные", и поэтому таблица стилей `print-color.css` не применяется в документе. То же самое относится и к любому экранному носителю информации, среде проецирования полутоновых изображений, среде акустического воспроизведения информации и т.д.

Каждый мультимедийный запрос состоит из типа носителя информации и одного или более перечисляемых свойств этого носителя. Каждое свойство носителя информации заключается в круглые скобки, а несколько таких свойств связываются вместе с помощью ключевого слова `and`. В мультимедийных запросах применяются следующие логические ключевые слова.

`and`

Связывает два или более свойства носителя информации таким образом, чтобы сам запрос оказался истинным, если истинны все перечисленные свойства. Например, запрос

(color) and (orientation: landscape) and (min-device-width: 800px) означает, что для применения заданной таблицы стилей должны быть удовлетворены все следующие условия: среда должна воспроизводить информацию в цвете, носитель должен иметь альбомную ориентацию, а устройство вывода должно быть в состоянии отображать информацию шириной не меньше **800** пикселей.

not

Отрицает весь запрос таким образом, чтобы таблица стилей *не* применялась, если истинны *все* условия запроса. Например, запрос not (color) and (orientation: landscape) and (min-devicewidth: 800px) означает, что если удовлетворяются все три условия, то запрос отклоняется. Так, если среда способна воспроизводить информацию в цвете, носитель имеет альбомную ориентацию, а устройство вывода в состоянии отображать информацию шириной не меньше 800 пикселей, то заданная таблица стилей *не* применяется. А во всех остальных случаях она будет применяться. Следует, однако, иметь в виду, что ключевое слово **not** должно быть указано только в начале мультимедийного запроса, а запросы наподобие (color) and not (mid-device-width: 800px) недопустимы. В таких случаях запрос будет проигнорирован. Следует также иметь в виду, что устарелые браузеры, не воспринимающие мультимедийные запросы, будут всегда пропускать таблицу стилей, в которой дескриптор носителя информации начинается с ключевого слова **not**.

Для указания условий в мультимедийных запросах не предусмотрено ключевое слово **or**. Его функцию выполняют запятые, через которые перечисляются условия запроса. Так, запрос screen, print означает, что таблица стилей применяется в том случае, если документ выводится на экран *или* печать. Следовательно, вместо недостоверного и поэтому игнорируемого запроса screen and (max-color: 2) or (mono chrome) нужно составить такой запрос: screen and (max-color: 2), screen and (monochrome).

Имеется еще одно ключевое слово **only**, предназначенное для целей обратной совместимости. Ниже приведено его краткое описание.

only

Служит для сокрытия таблицы стилей от тех браузеров, которые слишком устарели, чтобы воспринимать мультимедийные запросы. Например, чтобы применить таблицу стилей на всех носителях информации, но только в тех браузерах, которые способны воспринимать мультимедийные запросы, достаточно составить следующий запрос: `@import url(new.css) only all`. В тех браузерах, которые способны воспринимать мультимедийные запросы, ключевое слово **only** игнорируется. Следует, однако, иметь в виду, что ключевое слово **only** можно указывать *только* в начале мультимедийного запроса.

Параметры мультимедийных запросов

В качестве параметров мультимедийных запросов внедрены два новых типа значений, которые не применяются в каком-либо другом контексте. Эти значения описаны ниже.

<ratio>

Значение параметра пропорций состоит из двух положительных целочисленных значений типа `<integer>`, разделяемых знаком косой черты (**/**) и необязательными пробелами. Первое значение обозначает ширину, а второе — высоту. Следовательно, чтобы выразить пропорции 16:9 в виде соотношения ширины и высоты, в запросе достаточно указать **16/9** или **16 / 9**.

<resolution>

Значение параметра разрешения состоит из положительного целочисленного значения типа `<integer>` и обозначения единицы измерения **dpi** или **dpcm** заданного разрешения. Как правило, пробелы между целочисленным значением типа

<integer> и обозначением единицы измерения заданного разрешения не допускаются.

Свойства носителей информации

Ниже перечислены свойства носителей информации, которые можно указывать в мультимедийных запросах. Следует, однако, иметь в виду, что значения всех этих свойств *не* должны быть отрицательными.

width, min-width, max-width
***Значения:* <length>**

Обозначают ширину области отображения для пользовательского агента. Для веб-браузера, рассчитанного на экранные носители информации, это ширина области просмотра, включая любые полосы прокрутки. А для печатного носителя информации это ширина страничного блока. Следовательно, свойство `(min-width: 850px)` применяется в том случае, если область просмотра оказывается шире **850** пикселей.

device-width, min-device-width, max-device-width
***Значения:* <length>**

Обозначают ширину всей области воспроизведения на устройстве вывода. Для экранных носителей информации это ширина экрана, а для печатных носителей информации — ширина страницы. Следовательно, свойство `(max-device-width: 1200px)` применяется в том случае, если область вывода на устройстве оказывается меньше **1200** пикселей в ширину.

height, min-height, max-height
***Значения:* <length>**

Обозначают высоту области отображения для пользовательского агента. Для веб-браузера, рассчитанного на экранные носители информации, это высота области просмотра, включая любые полосы прокрутки. А для печатного носителя информации это высота страничного блока. Следовательно,

свойство (height: 567px) применяется в том случае, если высота области просмотра оказывается равной **567** пикселям.

device-height, min-device-height, max-device-height
Значения: <length>

Обозначают высоту всей области воспроизведения на устройстве вывода. Для экранных носителей информации это высота экрана, а для печатных носителей информации — высота страницы. Следовательно, свойство (max-device-height: 400px) применяется в том случае, если область вывода на устройстве оказывается меньше **400** пикселей в высоту.

aspect-ratio, min-aspect-ratio, max-aspect-ratio
Значения: <ratio>

Обозначают пропорции, получаемые в результате сравнения свойств width и height носителя информации (см. приведенное выше описание значения типа <ratio>). Следовательно, свойство (min-aspect-ratio: 2/1) применяется к любой области просмотра, в которой соотношение ширины и высоты составляет хотя бы **2:1**.

device-aspect-ratio, min-device-aspect-ratio,
max-deviceaspect-ratio
Значения: <ratio>

Обозначают пропорции, получаемые в результате сравнения свойств device-width и device-height носителя информации (см. приведенное выше описание значения типа <ratio>). Следовательно, свойство device-aspect-ratio: 16/9) применяется к любому устройству вывода, в котором соотношение ширины и высоты в области отображения точно равно **16:9**.

color, min-color, max-color
Значения: <integer>

Обозначают наличие возможности воспроизводить информацию в цвете на устройстве вывода, причем необязательное целое число представляет количество битов, используемых в каждой составляющей цвета. Следовательно, свойство

(color) применяется к любому устройству с какой угодно глубиной цвета, тогда как свойство (min-color: 4) означает, что на каждую составляющую цвета должно приходиться не меньше **4** бит. Любое устройство, не поддерживающее воспроизведение информации в цвете, возвратит нулевое значение (**0**).

color-index, min-color-index, max-color-index
***Значения:* <integer>**

Обозначают общее количество цветов, доступных в справочной таблице цветов на устройстве вывода. Любое устройство, на котором справочная таблица цветов не применяется, возвратит нулевое значение (**0**). Следовательно, свойство (min-color-index: 256) применяется к любому устройству вывода, в котором доступно для воспроизведения не меньше **256** цветов.

monochrome, min-monochrome, max-monochrome
***Значения:* <integer>**

Обозначают наличие возможности воспроизводить информацию в монохромном виде, причем необязательное целое число представляет количество битов, приходящихся на каждый пиксель в буфере кадров устройства вывода. Следовательно, свойство (monochrome) применяется к любому монохромному устройству вывода, тогда как свойство (min-monochrome: 2) означает любое монохромное устройство вывода, в котором на каждый пиксель в буфере кадров приходится не меньше **2** бит.

resolution, min-resolution, max-resolution
***Значения:* <resolution>**

Обозначают разрешение устройства вывода, выражаемое как плотность пикселей и измеряемое в точках на дюйм (**dpi**) или в точках на сантиметр (**dpcm**). (См. приведенное выше описание значения типа <resolution>.) Если пиксели на устройстве вывода *не* оказываются квадратными, то для обозначения требуемого разрешения используется наименьшая из осей плотности. Так, если плотность по одной оси на устройстве вывода составляет **100dpcm**, а по другой оси — **120dpcm**,

то возвращается значение **100**. Кроме того, запрос не будет удовлетворен, если в нем просто указать свойство `resolution`, хотя он будет удовлетворен, если указать свойства `min-resolution` и `max-resolution`.

`Orientation`
Значения: **`portrait | landscape`**

Обозначает общую область вывода на устройстве, которое возвратит значение **`portrait`** (книжная ориентация), если значение свойства `height` носителя информации окажется больше, чем значение свойства `width` этого же носителя. В противном случае возвратится значение **`landscape`** (альбомная ориентация).

`scan`
Значения: **`progressive | interlace`**

Обозначает процесс (построчной или чересстрочной) развертки, применяемый на устройстве вывода с типом носителя информации **`tv`**.

`Grid`
Значения: `0 | 1`

Обозначает наличие (или отсутствие) сеточного устройства вывода, например терминала типа `tty`. Сеточное устройство вывода возвратит значение **1**, а иначе — нулевое значение (**0**).

Глава 4

Справочник свойств

Универсальные значения

Любой пользовательский агент, полностью реализовавший модуль, действующий по принципу каскадирования и наследования, будет учитывать значения **inherit** и **initial** во всех свойствах. На практике поддержка значения **inherit** нашла более широкое распространение, чем поддержка значения **initial**.

inherit

Вынуждает наследовать значение свойства от родительского элемента данного элемента, даже если рассматриваемое свойство не наследуется (например, свойство background-image). Такое наследование можно рассматривать и как копирование значений из родительского элемента.

initial

Вынуждает интерпретировать значение свойства как исходное, определяемое в соответствующем модуле CSS. Например, в объявлении font-style: initial устанавливается значение **normal** свойства font-style независимо от того, было ли унаследовано значение этого свойства от родительского элемента. В тех случаях, когда исходное значение определяется как задаваемое пользовательским агентом, например значение свойства font-size, оно устанавливается как определяемое по умолчанию среди глобальных параметров пользовательского агента.

Свойства визуальных носителей информации

Ниже перечислены все свойства визуальных носителей информации с кратким пояснением их назначения.

animation

Значения:

```
[< animation-parameters >] [, [<animation-parameters >]]*
```

Расширения:

<animation-parameters>

```
<'animation-name'> || <'animation-duration'> ||
<'animation-timing-function'> || <'animation-delay'> ||
<'animation-iteration-count'> || <'animation-direction'>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Это сокращенная форма свойства, охватывающая все остальные свойства видов CSS-анимации, указываемых списком через запятую. Отдельные части значения этого свойства могут быть указаны в любом порядке, поэтому следует избегать возможной неоднозначности значений задержки и

продолжительности анимации. На момент написания данной книги первой обычно указывалась продолжительность анимации, а второй — ее задержка, хотя это и нельзя гарантировать.

Примеры:

```
div#slide {animation: 'slide' 2.5s linear 0 1 normal;}
h1 {animation: 'bounce' 0.5s 0.33s
    ease-in-out infinite alternate;}
```

animation-delay

Значения:

<time> [, <time>]*

Исходное значение:

0ms

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет период времени, в течение которого пользовательский агент ожидает начала заданных видов CSS-анимации. Таймер запускается, когда пользовательский агент применяет CSS-анимацию. Для неинтерактивного элемента это, скорее всего, должно произойти по окончании загрузки вебстраницы, хотя и не гарантируется.

Примеры:

```
body {animation-delay: 1s, 2000ms, 4s;}
a:hover {animation-delay: 400ms;}
```

Примечание:

Согласно спецификации, на момент написания данной книги конкретное значение этого свойства по умолчанию было равно нулю (**0**). Но ради ясности оно указано как **0ms**, поскольку в качестве безразмерных нулевых значений разрешается указывать только числовые значения длины.

animation-direction

Значения:

normal | **alternate** [, **normal** | **alternate**]*

Исходное значение:

normal

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Обозначает, должна ли многоцикловая CSS-анимация (см. ниже описание свойства `animation-iteration-count`) всегда воспроизводиться в одном и том же направлении или же она должна менять его на обратное через цикл. Так, если задать значение **alternate** данного свойства, то в ходе анимации элемент переместится на **300** пикселей вправо в одном цикле и на **300** пикселей влево в другом цикле, по существу, возвратившись в исходное положение. А если задать значение **normal** данного свойства, то в ходе анимации элемент переместится на **300** пикселей вправо, затем возвратится в исходное положение и снова переместится на **300** пикселей вправо, и так будет

продолжаться до завершения анимации, если это вообще предполагается.

Примеры:

```
body {animation-direction: alternate, normal, normal;}
#scanner {animation-direction: normal;}
```

animation-duration

Значения:

```
<time> [, <time>]*
```

Исходное значение:

0ms

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет продолжительность каждого цикла CSS-анимации от начала и до конца ее выполнения. Следовательно, для видов одноцикловой анимации это свойство определяет общую их продолжительность. Устанавливаемое по умолчанию значение **0ms** данного свойства означает, что анимация элемента ограничивается только его переходом из начального состояния в конечное. Отрицательные значения данного свойства автоматически приводятся к значению **0ms**.

Примеры:

```
h1 {animation-duration: 10s, 5s, 2.5s, 1250ms;}
.zip {animation-duration: 90ms;}
```

Примечание:

Согласно спецификации, на момент написания данной книги конкретное значение этого свойства по умолчанию было равно нулю (**0**). Но ради ясности оно указано как **0ms**, поскольку в качестве безразмерных нулевых значений разрешается указывать только числовые значения длины.

animation-iteration-count

Значения:

infinite | <number> [, **infinite** | <number>]*

Исходное значение:

1

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет количество циклов в анимации. Исходное значение **1** этого свойства означает, что анимация будет выполнена только один раз с переходом из начального состояния в конечное; дробное значение (например, **2.75**) означает, что анимация будет остановлена посредине своего завершающего цикла; а нулевое значение (**0**) означает, что анимация вообще не состоится. Отрицательные значения данного свойства автоматически приводятся к нулевому значению (**0**). И наконец, значение **infinite** данного свойства означает, что анимация никогда не завершится, поэтому пользоваться этим значением следует осторожно.

Примеры:

```
body {animation-iteration-count: 2, 1, 7.5875;}
ol.dance {animation-iteration-count: infinite;}
```

animation-name

Значения:

none | **IDENT** [, **none** | **IDENT**]*

Исходное значение:

none

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет объявляемые имена видов CSS-анимации. Каждое имя **IDENT** обозначает @-правило анимации по ключевым кадрам. Если имя **IDENT** не объявлено или если указано ключевое слово **none**, то анимация не выполняется независимо от значений любых других ее свойств. Так, если задать свойство `animation-name: 'bounce', none, 'jumper';`, но не определить имя анимации `jumper`, то первая анимация будет выполнена, а вторая и третья — нет.

Примеры:

```
html {animation-name: 'turn', 'slide', none;}
h2 {animation-name: 'flip';}
```

animation-play-state

Значения:

running | **paused** [, **running** | **paused**]*

Исходное значение:

running

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет состояние выполнения одного или более видов CSS-анимации. Устанавливаемое по умолчанию состояние **running** оказывается самым полезным в статических средах CSS, но им можно воспользоваться, чтобы легко останавливать или начинать анимацию, изменяя модель DOM с помощью сценариев или применяя интерактивные средства CSS (например, псевдокласс `:hover`).

Примеры:

```
pre {animation-play-state: running, paused, running;}
table {animation-play-state: running;}
```

Примечание:

На момент написания данной книги это свойство предполагалось удалить из модуля CSS Animations.

animation-timing-function

Значения:

`<timing-function> [, < timing-function >]*`

Расширения:

<timing-function>

ease | **linear** | **ease-in** | **ease-out** | **ease-in-out** | **cubicbezier**(< number>, <number>, <number>, <number>)

Исходное значение:

ease

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок выполнения анимации в полном ее цикле или в отдельном ключевом кадре в зависимости от того, где именно применяется данное свойство. У всех определенных выше ключевых слов должны быть свои эквиваленты в вызове функции `cubicbezier()`. Например, эквивалентом ключевого слова **`linear`** служит вызов функции `cubicbezier(0,0,1,1)`. Следовательно, они должны иметь согласованные анимационные эффекты среди всех пользовательских агентов, хотя на такую согласованность не следует особенно полагаться.

Примеры:

```
h1 {animation-timing-function:
    ease, ease-in, cubic-bezier(0.13,0.42,0.67,0.75)}
p {animation-timing-function: linear;}
```

backface-visibility

Значения:

visible | **hidden**

Исходное значение:

visible

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, будет ли видна обратная сторона элемента, если повернуть его в имитируемом трехмерном пространстве лицевой стороной от наблюдателя. Если задано значение **hidden** данного свойства, элемент будет, по существу, не виден до тех пор, пока снова не будет повернут лицевой стороной к наблюдателю.

Примеры:

```
div.card {backface-visibility: hidden;}
span.cubeside {backface-visibility: visible;}
```

background

Значения:

```
[ <bg-layer> , ]* <final-bg-layer>
```

Расширения:

<bg-layer>

```
<bg-image> || <bg-position> [ / <bg-size> ]? || <bg-repeat>
|| <bgattachment'> || < bg-box>{1,2}
```

<final-bg-layer>

```
<bg-image> || <bg-position> [ / <bg-size> ]? || <bg-repeat>
|| <bgattachment'> || < bg-box>{1,2} || <bg-color>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

Допускается для значения <bg-position> (см. ниже описание свойства background-position) и обозначения обоих измерений участка заднего плана элемента, а также измерений исходного изображения.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Определяет краткий способ выражения различных свойств одного или более задних планов (или фонов) элемента в одном объявлении. Как и во всех остальных сокращенных формах свойств, все допустимые значения (например, повторение, расположение заднего плана и т.д.) устанавливаются в данном свойстве по умолчанию, если они не заданы явно. Следовательно, оба следующих объявления обусловят одинаковый внешний вид заднего плана:

```
background: yellow;
background: yellow none top left repeat;
```

Более того, эти значения по умолчанию могут отменять предыдущие объявления, сделанные с помощью более конкретных свойств заднего плана. Например, в объявлениях

```
h1 {background-repeat: repeat-x;}
h1, h2 {background: yellow url(headback.gif);}
```

устанавливается по умолчанию значение **repeat** для повторения заднего плана обоих элементов, h1 и h2, отменяя тем самым объявленное ранее значение repeat-x.

Если объявлено несколько задних планов, то цвет будет иметь только последний задний план. В тех случаях, когда изображения нескольких задних планов перекрываются, они накладываются одно на другое, причем первое изображение — наверху, а последнее — внизу. Такой порядок совершенно противоположен перекрытию при расположении изображений в CSS, и поэтому он может показаться нелогичным.

Примеры:

```
body {background: white url(bg41.gif)
      fixed center repeat-x;}
p {background:
   url(/pix/water.png) center repeat-x,
   top left url(/pix/stone.png) #555;}
pre {background: yellow;}
```

background-attachment

Значения:

```
<bg-attachment> [ , <bg-attachment> ]*
```

Расширения:

<bg-attachment>

scroll | fixed | local

Исходное значение:

scroll

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли прокручивать изображения заднего плана вместе с элементом при просмотре документа. С помощью этого свойства можно создавать "выравниваемые" задние планы. Подробнее об этом читайте в главе 9 упоминавшегося ранее третьего издания книги *CSS: The Definitive Guide* (издательство O'Reilly).

Примеры:

```
body {background-attachment: scroll, scroll, fixed;}
div.fixbg {background-attachment: fixed;}
```

Примечание:

В браузере Internet Explorer до версии 7 это свойство поддерживается только для элемента разметки body.

background-clip

Значения:

<bg-box> [, <bg-box>]*

Расширения:

<bg-box>

border-box | **padding-box** | **content-box**

Исходное значение:

border-box

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет в блоке элемента границу, по которой обрезается задний план, т.е. дальше он не воспроизводится. Исторически сложилось так, что это было равнозначно установке по умолчанию значения **border-box**, при котором задний план доходит до внешнего края области границы. Это свойство допускает плотную обрезку блоков по внешнему краю области внутреннего поля и по краю самого содержимого.

Примеры:

```
body {background-clip: content-box;}
.callout {
    background-clip: content-box, border-box,
    padding-box;}
```

background-color

Значения:

<color>

Исходное значение:

transparent

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет сплошной цвет заднего плана элемента. Этим цветом заполняется блок, определяемый по умолчанию значением свойства background-clip, а также области содержимого, внутреннего поля и границ элемента, доходя до внешнего

края границы элемента. На прозрачных участках границ (например, пунктирных или штриховых) цвет заднего плана проявляется сквозь эти участки в тех случаях, когда заполнение цветом заднего плана простирается на область границы.

Примеры:

```
h4 {background-color: white;}
p {background-color: rgba(50%,50%,50%,0.33);}
pre {background-color: #FF9;}
```

background-image

Значения:

<bg-image> [, <bg-image>]*

Расширения:

<bg-image>

<image> | **none**

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютный URI.

Описание:

Размещает одно или более изображений на заднем плане элемента. В зависимости от значения свойства background-repeat изображение может бесконечно располагаться мозаикой по одной оси или вообще не повторяться. Первоначальное

(т.е. исходное) изображение заднего плана размещается в соответствии со значением свойства background-position.

Примеры:

```
body {
     background-image: url(bg41.gif), url(bg43.png),
     url(bg51.jpg);}
h2 {background-image: url(http://www.pix.org/dots.png);}
```

background-origin

Значения:

<bg-box> [, <bg-box>]*

Расширения:

<bg-box>

border-box | **padding-box** | **content-box**

Исходное значение:

padding-box

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет в блоке элемента границу, относительно которой вычисляется расположение изображения заднего плана. Исторически сложилось так, что это было равнозначно установке по умолчанию значения **padding-box**. А это свойство допускает разные контексты расположения. Следует, однако,

иметь в виду, что если значение свойства background-origin позволяет изображению заднего плана простираться еще дальше, чем допускает значение свойства backgroundclip, то изображение располагается по краю, а часть его обрезается, как демонстрируется в следующем примере:

```
div#example {background-origin: border-box;
    background-clip: content-box;
    background-position: 100% 100%;}
```

В данном примере изображение будет размещаться таким образом, чтобы его правый нижний угол оказался вровень с правым нижним углом внешнего края границы. Но видны будут только те части изображения, которые приходятся на область содержимого.

Примеры:

```
html, body {background-origin: border-box;}
h1 {background-origin: content-box, padding-box;}
```

background-position

Значения:

```
<bg-position> [ , <bg-position> ]*
```

Расширения:

<bg-position>

```
[ [ top | bottom ] | [ <percentage> | <length> | left |
center | right ] [ <percentage> | <length> | top | center
| bottom ]? | [ center | [ left | right ] [ <percentage> |
<length> ]? ] && [ center | [ top | bottom ] [ <percentage> |
<length> ]? ] ]
```

Исходное значение:

0% 0%

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По соответствующей точке как на элементе, так и на исходном изображении.

Вычисляемое значение:

Смещения по абсолютной длине, если объявлено значение типа `<length>`, а иначе — значения, задаваемые в процентах.

Описание:

Определяет положение одного или более исходных изображений задних планов, как задано в свойстве `background-image`. Это точка, с которой начнется повторение или мозаичное расположение любого изображения заднего плана. Значения, задаваемые в процентах, определяют не только эту точку внутри элемента, но и ту же самую точку в самом исходном изображении. Благодаря этому изображение можно, например, отцентровать, объявив его расположение равным **`50% 50%`**. Если используются значения, задаваемые в процентах или по длине, то первое значение всегда обозначает расположение по горизонтали, а второе — по вертикали. А если задается только одно значение, то оно обозначает расположение по горизонтали, тогда как отсутствующее значение предполагается равным **`center`** или **`50%`**. Допускается указывать отрицательные значения, при которых изображение может оказаться расположенным за пределами области содержимого элемента, и тогда оно фактически не будет воспроизведено. На контекст, в котором размещается исходное изображение, может оказывать влияние значение свойства `background-origin`.

Примеры:

```
body {background-position: top center;}
div#navbar {background-position: right, 50% 75%, 0 40px;}
pre {background-position: 10px 50%;}
```

background-repeat

Значения:

<bg-repeat-style> [, <bg-repeat-style>]*

Расширения:

<bg-repeat-style>

repeat-x | **repeat-y** | {**repeat** | **space** | **round** | **no-repeat**}{1,2}

Исходное значение:

repeat

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет мозаичный рисунок расположения одного или более изображений заднего плана. Повторение начинается с исходного изображения, которое определяется значением свойства background-image и размещается в соответствии со значением свойства background-position, а возможно, и свойства background-origin. Если указаны ключевые слова **space** и **round**, то изображение располагается мозаикой столько раз, сколько оно помещается в области заднего плана без обрезки, а затем первое и последнее изображения размещаются относительно краев соответствующих задних планов. Различие этих ключевых слов заключается в том, что ключевое слово **space** обусловливает расположение перемежающихся изображений с равномерными промежутками, тогда как ключевое слово **round** — их растягивание до взаимного соприкосновения. Следует также иметь в виду, что значение

repeat-x равнозначно значениям **repeat no-repeat**, а значение **repeat-y** — значениям **no-repeat repeat**.

Примеры:

```
body {background-repeat: no-repeat;}
h2 {background-repeat: repeat-x, repeat-y, space;}
ul {background-repeat: repeat-y, round space, repeat;}
```

background-size

Значения:

<bg-size> [, <bg-size>]*

Расширения:

<bg-repeat-style>

[**<length>** | **<percentage>** | **auto**]{1,2} | **cover** | **contain**

Исходное значение:

auto

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. приведенное ниже описание.

Описание:

Определяет размеры одного или более изображений заднего плана. Если указаны два значения данного свойства (например, **50px 25%**), то первое из них определяет размер изображения по горизонтали, а второе — по вертикали. Исходное изображение может быть деформировано таким образом, чтобы полностью покрыть задний план по размерам **100% 100%**. С другой стороны, значение **cover** обозначает масштабирование изображения

с целью полностью покрыть задний план, даже если его часть простирается за пределы области заднего плана, и тогда оно обрезается. А значение **contain** обозначает масштабирование исходного изображения таким образом, чтобы заполнить им область заднего плана хотя бы по одному размеру.

Примеры:

```
body {background-size: 100% 90%;}
div.photo {background-size: cover;}
```

border

Значения:

```
<'border-width'> || <'border-style'> || <color>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Это сокращенная форма свойства, определяющая ширину, цвет и стиль границ элемента. Несмотря на то что ни одно из значений данного свойства фактически указывать не нужно, следует все же иметь в виду, что если опустить стиль границы, то обрамление элемента будет отсутствовать, поскольку стиль границы по умолчанию выбирается равным **none**.

Примеры:

```
h1 {border: 2px dashed olive;}
a:link {border: blue solid 1px;}
p.warning {border: double 5px red;}
```

border-bottom

Значения:

<'border-width'> || <'border-style'> || <color>

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая ширину, цвет и стиль нижней границы элемента. Если опустить стиль границы, как и в свойстве border, то обрамление элемента будет отсутствовать.

Примеры:

```
ul {border-bottom: 0.5in groove green;}
a:active {border-bottom: purple 2px dashed;}
```

border-bottom-color

Значения:

<color>

Исходное значение:

Значение свойства color, задаваемое для элемента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Если ни одно из значений на объявлено, используется вычисляемое значение свойства `color`, задаваемое для аналогичного элемента, а иначе — такое же, как и объявляемое значение.

Описание:

Определяет цвет видимых участков нижней границы элемента. Стиль этой границы должен отличаться от задаваемого значением **`none`** или **`hidden`**, чтобы она стала видимой.

Примеры:

```
ul {border-bottom-color: green;}
a:active {border-bottom-color: purple;}
```

border-bottom-left-radius

Значения:

```
[ <length> | <percentage> ] [ <length> | <percentage> ]?
```

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По размерам блока границы элемента.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет радиус скругления левого нижнего угла границы элемента. Если заданы два значения данного свойства, то первое из них обозначает горизонтальный радиус, а второе — вертикальный радиус. Порядок создания формы скругления приведен ниже, в описании свойства `border-radius`.

Примеры:

```
h1 {border-bottom-left-radius: 10%;}
h2 {border-bottom-left-radius: 1em 10px;}
```

border-bottom-right-radius

Значения:

```
[ <length> | <percentage> ] [ <length> | <percentage> ]?
```

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По размерам блока границы элемента.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет радиус скругления правого нижнего угла границы элемента. Если заданы два значения данного свойства, то первое из них обозначает горизонтальный радиус, а второе — вертикальный радиус. Порядок создания формы скругления приведен ниже, в описании свойства `border-radius`.

Примеры:

```
h1 {border-bottom-right-radius: 10%;}
h2 {border-bottom-right-radius: 1em 10px;}
```

border-bottom-style

Значения:

none | **hidden** | **dotted** | **dashed** | **solid** | **double** | **groove** | **ridge** | **inset** | **outset**

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет стиль нижней границы элемента. Значение данного свойства должно отличаться от **none**, чтобы эта граница стала видимой.

Примеры:

```
ul {border-bottom-style: groove;}
a:active {border-bottom-style: dashed;}
```

border-bottom-width

Значения:

thin | **medium** | **thick** | <length>

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютная длина. Нулевое значение (**0**), если задан стиль границы **none** или **hidden**.

Описание:

Определяет ширину нижней границы элемента, что возымеет действие только в том случае, если задан любой стиль границы, кроме **none**. Если же задан стиль границы **none**, то ее ширина, по существу, становится равной нулю (**0**). Отрицательные значения длины в данном свойстве не допускаются.

Примеры:

```
ul {border-bottom-width: 0.5in;}
a:active {border-bottom-width: 2px;}
```

border-collapse

Значения:

collapse | **separate**

Исходное значение:

separate

Применяется:

К элементам со значением **table** или **inline-table** свойства display.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет модель разметки, применяемую для расположения в таблице границ вокруг ячеек, строк и т.д. И хотя данное свойство применяется только к таблицам, оно наследуется всеми элементами в таблице, в которой оно, по существу, и применяется.

Пример:

```
table {border-collapse: separate; border-spacing: 3px 5px;}
```

Примечание:

В спецификации CSS2.0 по умолчанию устанавливалось значение **collapse** данного свойства.

border-color

Значения:

<color>{**1**,**4**}

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, задающая цвет видимых участков общей границы элемента или разный цвет границы по каждой из четырех сторон. Не следует, однако, забывать, что стиль границы должен отличаться от **none** или **hidden**, чтобы она стала видимой.

Примеры:

```
h1 {border-color: purple;}
a:visited {border-color: maroon;}
```

border-image

Значения:

```
<'border-image-source'> || <'border-image-slice'>
[ / <'borderimage-width'>? [ / <'border-image-outset'> ]? ]? ||
<'border-imagerepeat'>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

Ко всем элементам, *кроме* табличных элементов, в которых свойство `border-collapse` принимает значение **collapse**.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая источник, образец нарезки, ширину и повторение границы, формируемой на основе изображения. Синтаксис этого свойства не совсем обычен по сравнению с остальной частью CSS и

поэтому требует дополнительного времени на изучение. Например, три из пяти возможных значений данного свойства разделяются знаком косой черты и должны быть перечислены в определенном порядке.

Следует, однако, иметь в виду, что организовать повторение простого изображения (например, звезды) по краям элемента практически невозможно. Чтобы добиться такого эффекта, придется составить одно изображение из девяти копий того изображения, которое требуется повторить, в виде сетки **3×3**. Возможно, также придется задать достаточно большим значение `border-width` (но *не* значение `border-image-width`), чтобы изображение было видимым, хотя это зависит от значения `border-image-outset`.

Примеры:

```
div.starry {border-image: url(stargrid.png) 5px repeat;}
aside {
    border-image: url(asides.png) 100 50 150 / 8 3 13 /
    2 stretch round;}
```

Примечание:

На момент написания данной книги поддержка свойства `border-image` в браузерах была неполной и несогласованной, тогда как ни одно из связанных с ним свойств (например, `border-image-source`) вообще не поддерживалось. Тем не менее описание этих свойств было включено в книгу, поскольку в дальнейшем предполагалось согласовать их поддержку в браузерах.

border-image-outset

Значения:

`[ <length> | <number> ]{1,4}`

Исходное значение:

0

Применяется:

Ко всем элементам, *кроме* табличных элементов, в которых свойство `border-collapse` принимает значение **`collapse`**.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение, тем не менее читайте приведенное ниже описание.

Описание:

Определяет расстояние, на которое изображение границы может простираться за пределы области границы элемента. Значения этого свойства определяют расстояние от верхнего, правого, нижнего и левого краев изображения границы (именно в таком порядке). Числовые значения вычисляются относительно внутренней системы координат изображения. Следовательно, для растрового изображения числовое значение **7** означает расстояние в семь пикселей. А изображения в других форматах, например SVG, могут иметь другую систему координат. Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
aside {border-image-outset: 2;}
div#pow {border-image-outset: 10 17 13 5;}
```

Примечание:

На момент написания данной книги поддержка свойства `border-image-outset` в браузерах отсутствовала. Тем не менее его описание включено в эту книгу, поскольку свойство `border-image`, охватывающее свойство `border-image-outset`, *уже* поддерживалось в браузерах, а в дальнейшем предполагалась также поддержка свойства `border-image-outset`.

border-image-repeat

Значения:

[**stretch** | **repeat** | **round**]{1,2}

Исходное значение:

stretch

Применяется:

Ко всем элементам, *кроме* табличных элементов, в которых свойство `border-collapse` принимает значение **collapse**.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет образец повторения (или его отсутствие) по сторонам изображения границы. В частности, значение **stretch** обусловливает растягивание одной копии изображения по всему участку (верхней, правой, нижней или левой стороне) границы. Значение **repeat** обусловливает мозаичное расположение изображения границы аналогично изображению заднего плана, хотя изображения границ могут располагаться мозаикой только по одной оси. А значение **round** обусловливает мозаичное расположение изображения границы столько раз, сколько оно может уместиться без обрезки, после чего все расположенные мозаикой изображения масштабируются, если нужно, чтобы они точно поместились на участке границы.

Примеры:

```
div.starry {border-image-repeat: repeat;}
aside {border-image-repeat: stretch round;}
```

Примечание:

На момент написания данной книги поддержка свойства `border-image-repeat` в браузерах отсутствовала. Тем не менее его описание включено в эту книгу, поскольку свойство `border-image`, охватывающее свойство `border-image-repeat`, *уже* поддерживалось в браузерах, а в дальнейшем предполагалась также поддержка свойства `border-image-repeat`.

border-image-slice

Значения:

`[<number> | <percentage>]{1,4} && fill?`

Исходное значение:

100%

Применяется:

Ко всем элементам, *кроме* табличных элементов, в которых свойство `border-collapse` принимает значение **collapse**.

Наследуется:

Нет.

Задание в процентах:

По размерам изображения границы.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет "расстояния нарезки", т.е. смещения от верхнего, правого, нижнего и левого краев изображения границы. Совместно эти расстояния делят изображение на девять областей, которые соответствуют восьми участкам (четырем углам и четырем сторонам) границы элемента и области заднего плана элемента.

В тех случаях, когда две противоположные области нарезки объединяются и превышают общий для них размер, они обе становятся совершенно прозрачными. Так, если значение смещения нарезки по верхнему краю составляет **10**, а значение смещения нарезки по нижнему краю — **20**, но исходное изображение оказывается высотой лишь **25** пикселей, то обе нарезки превышают высоту изображения. Следовательно, верхний и нижний участки границы окажутся совершенно прозрачными. То же самое относится и к нарезкам по левому и правому краям границы. Но углы изображения не становятся прозрачными, даже если их нарезки перекрываются в исходном изображении.

Примеры:

```
div.starry {border-image-slice: 5px;}
aside {border-image-slice: 100 50 150;}
```

Примечание:

На момент написания данной книги поддержка свойства `border-image-slice` в браузерах отсутствовала. Тем не менее его описание включено в эту книгу, поскольку свойство `border-image`, охватывающее свойство `border-image-slice`, *уже* поддерживалось в браузерах, а в дальнейшем предполагалась также поддержка свойства `border-image-slice`.

border-image-source

Значения:

none | <uri>

Исходное значение:

none

Применяется:

Ко всем элементам, *кроме* табличных элементов, в которых свойство `border-collapse` принимает значение **collapse**.

Наследуется:

Нет.

Задание в процентах:

По размерам изображения границы.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Предоставляет местоположение изображения, применяемого в качестве изображения границы элемента.

Примеры:

```
div.starry {border-image-source: url(stargrid.png);}
aside {border-image-source: url(asides.png);}
```

Примечание:

На момент написания данной книги поддержка свойства `border-image-source` в браузерах отсутствовала. Тем не менее его описание включено в эту книгу, поскольку свойство `border-image`, охватывающее свойство `border-image-source`, *уже* поддерживалось в браузерах, а в дальнейшем предполагалась также поддержка свойства `border-image-source`.

border-image-width

Значения:

`[ <length> | <percentage> | <number> | auto ]{1,4}`

Исходное значение:

1

Применяется:

Ко всем элементам, *кроме* табличных элементов, в которых свойство `border-collapse` принимает значение **collapse**.

Наследуется:

Нет.

Задание в процентах:

По размерам области изображения границы.

Вычисляемое значение:

Такое же, как и объявляемое значение, тем не менее читайте приведенное ниже описание.

Описание:

Определяет ширину изображения границы по каждой из его четырех сторон. Нарезки изображения границы, отличающиеся по ширине от самого изображения границы, масштабируются, чтобы вписаться в пределы границы, что может повлиять на порядок их повторения. Так, если правый край изображения границы составляет 10 пикселей в ширину, а ширина изображения границы объявлена как `border-image-width: 3px;`, то изображения границы по правому краю будут уменьшены до трех пикселей в ширину.

Следует, однако, иметь в виду, что свойство `border-image-width` отличается от свойства `border-width` тем, что ширина изображения границы может быть иной, чем ширина области границы. В тех случаях, когда изображение оказывается шире или выше области границы, оно обрезается по умолчанию, хотя этому может воспрепятствовать установка свойства `border-image-outset`. А если изображение оказывается уже или короче области границы, его масштаб не увеличивается.

Примеры:

```
aside {border-image-width: 8 3 13;}
div#pow{border-image-width: 25px 35;}
```

Примечание:

На момент написания данной книги поддержка свойства `border-image-width` в браузерах отсутствовала. Тем не менее его описание включено в эту книгу, поскольку свойство

border-image, охватывающее свойство border-image-width, *уже* поддерживалось в браузерах, а в дальнейшем предполагалась также поддержка свойства border-image-width.

border-left

Значения:

<'border-width'> || <'border-style'> || <color>

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая ширину, цвет и стиль левой границы элемента. Если опустить стиль границы, как и в свойстве border, то обрамление элемента будет отсутствовать.

Примеры:

```
p {border-left: 3em solid gray;}
pre {border-left: double black 4px;}
```

border-left-color

Значения:

<color>

Исходное значение:

Значение свойства `color`, задаваемое для элемента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Если ни одно из значений не объявлено, используется вычисляемое значение свойства `color`, задаваемое для аналогичного элемента, а иначе — такое же, как и объявляемое значение.

Описание:

Определяет цвет видимых участков левой границы элемента. Стиль этой границы должен отличаться от задаваемого значением **`none`** или **`hidden`**, чтобы она стала видимой.

Примеры:

```
p {border-left-color: gray;}
pre {border-left-color: black;}
```

border-left-style

Значения:

`none` | **`hidden`** | **`dotted`** | **`dashed`** | **`solid`** | **`double`** | **`groove`** | **`ridge`** | **`inset`** | **`outset`**

Исходное значение:

`none`

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет стиль левой границы элемента. Значение данного свойства должно отличаться от **none**, чтобы эта граница стала видимой.

Примеры:

```
p {border-left-style: solid;}
pre {border-left-style: double;}
```

border-left-width

Значения:

thin | **medium** | **thick** | <length>

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютная длина. Нулевое значение (**0**), если задан стиль границы **none** или **hidden**.

Описание:

Определяет ширину левой границы элемента, что возымеет действие только в том случае, если задан любой стиль границы, кроме **none**. Если же задан стиль границы **none**, то ее ширина, по существу, становится равной нулю (**0**). Отрицательные значения длины в данном свойстве не допускаются.

Примеры:

```
p {border-left-width: 3em;}
pre {border-left-width: 4px;}
```

border-radius

Значения:

```
[ <length> | <percentage> ]{1,4} [ / [ <length> |
<percentage> ] {1,4}
```

Исходное значение:

0

Применяется:

Ко всем элементам, *кроме* табличных элементов, в которых свойство `border-collapse` принимает значение **`collapse`**.

Наследуется:

Нет.

Задание в процентах:

По размерам блока границы элемента (см. приведенное ниже описание).

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Это сокращенная форма свойства, определяющая радиус скругления правого нижнего угла границы элемента. Конкретные углы объявляются в нем по высоте и ширине. Рассмотрим в качестве примера следующее объявление:

```
.callout {border-radius: 10px;}
```

Каждый угол элемента с выноской, определяемой селектором класса в этом объявлении, будет скруглен на 10 пикселей по ширине, если отсчитывать от начала скругления до

внешнего края элемента, и точно так же по высоте. Это все равно как если бы по углам элемента были нарисованы окружности радиусом **10** пикселей, т.е. диаметром **20** пикселей, а его граница была затем изогнута по краям этих окружностей.

Следует заметить, что если при таком синтаксисе задаются два значения, то первое из них относится к левому верхнему и правому нижнему углам, а второе — к правому верхнему и левому нижнему углам. Чтобы создать овальное скругление, задав одно значение для горизонтального радиуса каждого угла, а другое значение для вертикального радиуса, следует разделить эти значения знаком косой черты, как показано ниже:

```
.callout {border-radius: 10px / 20px;}
```

В итоге все четыре угла будут скруглены на **10** пикселей по горизонтали и на **20** пикселей по вертикали. А задать скругление каждого из четырех углов по отдельности можно следующим образом:

```
.callout {border-radius: 10px 20px 30px 40px /1em 2em 3em 4em;}
```

И это равнозначно следующему объявлению:

```
.callout {border-top-left-radius: 10px 1em;
    border-top-right-radius: 20px 2em;
    border-bottom-right-radius: 30px 3em;
    border-bottom-left-radius: 40px 4em;}
```

Если задать меньше четырех значений, то указанные значения будут повторяться в известном (для отступов, внутренних полей и т.д.) порядке, но с некоторым смещением. Порядок следования значений свойств для скругления углов должен быть таким: левый верхний, правый верхний, правый нижний, левый нижний угол. В противном случае образец повторения останется тем же самым, т.е. значение **1em** будет аналогично последовательности значений **1em 1em 1em 1em**, а значения **1em 2em** — последовательности значений **1em 2em 1em 2em** и т.д. Таким образом, значения, указываемые по разные стороны от знака косой черты, могут быть разными. Так, оба следующие объявления равнозначны:

```
.callout {border-radius: 2em 3em 4em / 5%;}
.callout {border-radius: 2em 3em 4em 3em / 5% 5% 5% 5%;}
```

Если значения данного свойства указываются в процентах, то они вычисляются относительно размера блока границы элемента, определяемого внешними краями области границы элемента, по соответствующей оси. Следовательно, в приведенных выше объявлениях значения **5%** вычисляются как составляющие 5% от высоты блока границы элемента, поскольку значения, указанные после знака косой черты, определяют вертикальные радиусы скругления углов. А любые значения, указываемые в процентах до знака косой черты, вычисляются в процентах от ширины блока границы элемента.

Примеры:

```
a[href] {border-radius: 0.5em 50%;}
.callout {
    border-radius: 10px 20px 30px 40px /
    1em 2em 3em 4em;}
```

border-right

Значения:

```
<'border-width'> || <'border-style'> || <color>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая ширину, цвет и стиль правой границы элемента. Если опустить стиль границы, как и в свойстве `border`, то обрамление элемента будет отсутствовать.

Примеры:

```
img {border-right: 30px dotted blue;}
h3 {border-right: cyan 1em inset;}
```

border-right-color

Значения:

`<color>`

Исходное значение:

Значение свойства `color`, задаваемое для элемента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Если ни одно из значений на объявлено, используется вычисляемое значение свойства `color`, задаваемое для аналогичного элемента, а иначе — такое же, как и объявляемое значение.

Описание:

Определяет цвет видимых участков правой границы элемента. Стиль этой границы должен отличаться от задаваемого значением **`none`** или **`hidden`**, чтобы она стала видимой.

Примеры:

```
img {border-right-color: blue;}
h3 {border-right-color: cyan;}
```

border-right-style

Значения:

none | **hidden** | **dotted** | **dashed** | **solid** | **double** | **groove** | **ridge** | **inset** | **outset**

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет стиль правой границы элемента. Значение данного свойства должно отличаться от **none**, чтобы эта граница стала видимой.

Примеры:

```
img {border-right-style: dotted;}
h3 {border-right-style: inset;}
```

border-right-width

Значения:

thin | **medium** | **thick** | <length>

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютная длина. Нулевое значение (**0**), если задан стиль границы **none** или **hidden**.

Описание:

Определяет ширину правой границы элемента, что возымеет действие только в том случае, если задан любой стиль границы, кроме **none**. Если же задан стиль границы **none**, то ее ширина, по существу, становится равной нулю (**0**). Отрицательные значения длины в данном свойстве не допускаются.

Примеры:

```
img {border-right-width: 30px;}
h3 {border-right-width: 1em;}
```

border-spacing

Значения:

```
<length> <length>?
```

Исходное значение:

0

Применяется:

К элементам со значением **table** или **inline-table** свойства display.

Наследуется:

Да.

Вычисляемое значение:

Две абсолютные длины.

Описание:

Определяет расстояние между границами ячеек таблицы в модели разметки разделяемых границ таблицы. Первое из двух значений длины обозначает разделение по горизонтали, а второе — по вертикали. Это свойство принимается во внимание лишь в том случае, если задано значение **separate** свойства border-collapse, а иначе оно игнорируется. И хотя данное свойство применяется только к таблицам, оно наследуется всеми элементами в таблице, в которой оно, по существу, и применяется.

Примеры:

```
table {border-collapse: separate; border-spacing: 0;}
table {border-collapse: separate;
       border-spacing: 3px 5px;}
```

border-style

Значения:

[**none** | **hidden** | **dotted** | **dashed** | **solid** | **double** | **groove** | **ridge** | **inset** | **outset**]{**1**,**4**}

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая стили общей границы элемента или каждой его стороны в отдельности.

Значение данного свойства для любой границы должно отличаться от **none**, чтобы эта граница стала видимой. Следует, однако, иметь в виду, что если установить значение **none** свойства `border-style`, что делается по умолчанию, то любое значение свойства `border-width` будет проигнорировано, а ширина границы станет равной нулю (**0**). Любое нераспознанное значение из списка заданных значений должно быть поновому интерпретировано как **solid**.

Примеры:

```
h1 {border-style: solid;}
img {border-style: inset;}
```

border-top

Значения:

```
<'border-width'> || <'border-style'> || <color>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая ширину, цвет и стиль верхней границы элемента. Если опустить стиль границы, как и в свойстве `border`, то обрамление элемента будет отсутствовать.

Примеры:

```
ul {border-top: 0.5in solid black;}
h1 {border-top: dashed 1px gray;}
```

border-top-color

Значения:

<color>

Исходное значение:

Значение свойства color, задаваемое для элемента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Если ни одно из значений на объявлено, используется вычисляемое значение свойства color, задаваемое для аналогичного элемента, а иначе — такое же, как и объявляемое значение.

Описание:

Определяет цвет видимых участков верхней границы элемента. Стиль этой границы должен отличаться от задаваемого значением **none** или **hidden**, чтобы она стала видимой.

Примеры:

```
ul {border-top-color: black;}
h1 {border-top-color: gray;}
```

border-top-left-radius

Значения:

[<length> | <percentage>] [<length> | <percentage>]?

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По размерам блока границы элемента.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет радиус скругления левого верхнего угла границы элемента. Если заданы два значения данного свойства, то первое из них обозначает горизонтальный радиус, а второе — вертикальный радиус. Порядок создания формы скругления приведен выше, в описании свойства `border-radius`.

Примеры:

```
h1 {border-top-left-radius: 10%;}
h2 {border-top-left-radius: 1em 10px;}
```

border-top-right-radius

Значения:

```
[ <length> | <percentage> ] [ <length> | <percentage> ]?
```

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По размерам блока границы элемента.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет радиус скругления правого верхнего угла границы элемента. Если заданы два значения данного свойства, то первое из них обозначает горизонтальный радиус, а второе — вертикальный радиус. Порядок создания формы скругления приведен выше, в описании свойства `border-radius`.

Примеры:

```
h1 {border-top-right-radius: 10%;}
h2 {border-top-right-radius: lem 10px;}
```

border-top-style

Значения:

none | **hidden** | **dotted** | **dashed** | **solid** | **double** | **groove** | **ridge** | **inset** | **outset**

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет стиль верхней границы элемента. Значение данного свойства должно отличаться от **none**, чтобы эта граница стала видимой.

Примеры:

```
ul {border-top-style: solid;}
hl {border-top-style: dashed;}
```

border-top-width

Значения:

thin | **medium** | **thick** | <length>

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютная длина. Нулевое значение (**0**), если задан стиль границы **none** или **hidden**.

Описание:

Определяет ширину верхней границы элемента, что возымеет действие только в том случае, если задан любой стиль границы, кроме **none**. Если же задан стиль границы **none**, то ее ширина, по существу, становится равной нулю (**0**). Отрицательные значения длины в данном свойстве не допускаются.

Примеры:

```
ul {border-top-width: 0.5in;}
hl {border-top-width: lpx;}
```

border-width

Значения:

[**thin** | **medium** | **thick** | <length>]{**1,4**}

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая ширину общей границы элемента или каждой его стороны в отдельности. Ширина заданной границы вступит в действие только в том случае, если задан любой стиль границы, кроме **none**. Если же задан стиль границы **none**, то ее ширина, по существу, становится равной нулю (**0**). Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
h1 {border-width: 2ex;}
img {border-width: 5px thick thin 1em;}
```

bottom

Значения:

<length> | <percentage> | **auto**

Исходное значение:

auto

Применяется:

К размещаемым элементам, т.е. к таким элементам, для которых задано значение свойства position, отличающееся от **static**.

Наследуется:

Нет.

Задание в процентах:

По высоте содержащего блока.

Вычисляемое значение:

Для относительно размещаемых элементов читайте приведенное ниже описание; для статически размещаемых элементов (т.е. таких элементов, для которых задано значение **static** свойства position) вычисляемое значение равно **auto**; для значений длины оно равно соответствующей абсолютной длине; для значений в процентах — объявляемому значению, а иначе — **auto**.

Описание:

Определяет смещение внешнего края нижнего отступа размещаемого элемента относительно нижнего края содержащего его блока. Если для относительно размещаемых элементов задано значение **auto** обоих свойств, bottom и top, то их вычисляемые значения равны нулю (**0**). Если же только одно из этих свойств принимает значение **auto**, то его вычисляемое значение становится отрицательным значением другого свойства. А если ни одно из этих свойств не принимает значение **auto**, то вычисляемое значение свойства bottom становится отрицательным значением свойства top.

Примеры:

```
div#footer {position: fixed; bottom: 0;}
sup {position: relative; bottom: 0.5em;
     vertical-align: baseline;}
```

box-align

Значения:

stretch | **start** | **end** | **center** | **baseline**

Исходное значение:

stretch

Применяется:

Ко всем элементам со значением **box** или **inline-box** свойства display.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок размещения гибких блоков по оси, перпендикулярной оси ориентации (см. ниже описание свойства box-orient). Устанавливаемое по умолчанию значение **stretch** этого свойства означает, что блоки, порожденные данным блоком (т.е. его потомки), растягиваются по его высоте, если задано значение **horizontal** его свойства box-orient, или же по его ширине, если задано значение **vertical** того же свойства. Значения **start** и **end** данного свойства обозначают верхние и нижние края горизонтальных блоков, и то же самое, вероятнее всего, произойдет с их левыми и правыми краями в языках с правосторонним письмом, хотя такое поведение не определено. Если же задано значение **center** данного свойства, то центр гибких блоков выравнивается по центральной линии оси ориентации.

Примеры:

```
div#layout {box-align: stretch;}
.icicle {box-align: start;}
```

Примечание

Это свойство определено в спецификации модуля Flexible Box 2009 года. В новой версии этой спецификации предполагается сделать данное свойство не рекомендованным к применению, после чего поддержка в браузерах будет, вероятнее всего, прекращена.

box-decoration-break

Значения:

slice | **clone**

Исходное значение:

clone

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, применяются ли воспроизводимые многими частями элементы оформления блока, включая задний план, внутреннее поле, границы, скругленные углы, изображение заднего плана и тень от блока, к каждой части или ко всему блоку в целом, прежде чем разделять его.

Наиболее характерным примером применения данного свойства служит внутристрочной элемент, охватывающий один или более разрывов строк. При стандартном поведении, когда задано значение **slice** данного свойства, отдельные части внутристрочного элемента воспроизводятся так, как будто единый элемент сначала размещен в одной строке, а затем

разбит по каждому разрыву строки. А если задано значение **clone** данного свойства, то каждая часть элемента оформляется так, как будто отдельные элементы разделяют одни и те же общие стили. Свойство box-decoration-break применяется также к структурным блокам, разделяемым по столбцам или страницам.

Примеры:

```
span {box-decoration-break: clone;}
a {box-decoration-break: slice;}
```

box-direction

Значения:

normal | **reverse**

Исходное значение:

normal

Применяется:

Ко всем элементам со значением **box** или **inline-box** свойства display.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет направление, в котором размещаются потомки блока. Если задано значение **reverse** данного свойства, потомки блока размещаются справа налево в горизонтальном блоке и снизу вверх в вертикальном блоке.

Пример:

```
#tower {box-direction: reverse;}
```

Примечание:

Это свойство определено в спецификации модуля Flexible Box 2009 года. В новой версии этой спецификации предполагается сделать данное свойство не рекомендованным к применению, после чего поддержка в браузерах будет, вероятнее всего, прекращена.

box-flex

Значения:

`<number>`

Исходное значение:

0

Применяется:

К потомкам элемента из обычного потока со значением **box** или **inline-box** свойства `display`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет "гибкость" элемента, являющегося потомком блока. Значения свойства `box-flex` всех гибких блоков в группе сначала складываются, а затем каждое из них делится на общую сумму, чтобы получить степень гибкости элемента. Так, если все три гибких блока в группе имеют значение **1** данного свойства, то степень гибкости каждого из этих блоков составит **0,33**. Если же значение свойства `box-flex` одного из блоков изменится на **2**, то степень его гибкости составит **0,5**, а двух других блоков — **0,25**. А нулевое значение (**0**) данного

свойства, устанавливаемое по умолчанию, обозначает, что блок является негибким.

После того как гибкие блоки будут размещены обычным образом, любое лишнее пространство, оставшееся в родительском блоке, распределяется по гибким блокам в соответствии со степенью его гибкости. Так, если в предыдущем примере останется **100** пикселей лишнего пространства, то при значении **2** свойства `flex-box` один элемент будет расширен на **50** пикселей, а два других — на **25** пикселей каждый. А если гибкие блоки переполняют родительский блок, их размеры пропорционально сокращаются.

Пример:

```
#nav li {box-flex: 1;}
```

Примечание:

Это свойство определено в спецификации модуля Flexible Box 2009 года. В новой версии этой спецификации предполагается сделать данное свойство не рекомендованным к применению, после чего поддержка в браузерах будет, вероятнее всего, прекращена.

box-lines

Значения:

single | **multiple**

Исходное значение:

single

Применяется:

К элементам со значением **box** или **inline-box** свойства `display`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок размещения гибких блоков, если они оказываются слишком широкими в горизонтальном блоке (см. ниже описание свойства `box-orient`). Если задано значение **`multiple`** данного свойства, гибкие блоки будут размещены в стольких "строках", сколько потребуется для отображения всех этих блоков. И это напоминает плавающую разметку, когда свободно перемещаемые (т.е. плавающие) блоки не могут располагаться рядом, хотя механизм в данном случае несколько иной.

Пример:

```
#nav li {box-flex: 1;}
```

Примечание:

Это свойство определено в спецификации модуля Flexible Box 2009 года. В новой версии этой спецификации предполагается сделать данное свойство не рекомендованным к применению, после чего поддержка в браузерах будет, вероятнее всего, прекращена.

box-ordinal-group

Значения:

`<integer>`

Исходное значение:

`1`

Применяется:

К потомкам элемента из обычного потока со значением **`box`** или **`inline-box`** свойства `display`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядковую группу, к которой относятся гибкие блоки. Гибким блокам можно присваивать произвольные номера групп. При разметке блоков группы располагаются по порядку номеров, причем гибкие блоки в каждой группе размещаются в их исходном порядке и в соответствии со значением свойства `box-direction`. Это дает возможность располагать гибкие блоки совершенно независимо от их исходного порядка.

Примеры:

```
.sticky {box-ordinal-group: 1;}
.footer {box-ordinal-group: 13;}
```

Примечание

Это свойство определено в спецификации модуля Flexible Box 2009 года. В новой версии этой спецификации предполагается сделать данное свойство не рекомендованным к употреблению, после чего поддержка в браузерах будет, вероятнее всего, прекращена.

box-orient

Значения:

horizontal | **vertical** | **inline-axis** | **block-axis**

Исходное значение:

inline-axis

Применяется:

К элементам со значением **box** или **inline-box** свойства `display`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет направление, в котором гибкие блоки размещаются в их родительском блоке. Если задано значение **horizontal** данного свойства, гибкие блоки располагаются слева направо, а если задано значение **vertical**, то сверху вниз. Установка значений **inline-axis** и **block-axis** имеет последствия, зависящие от конкретного языка. Так, в языке с правосторонним письмом, например в английском, эти значения равнозначны значениям **horizontal** и **vertical**.

Примеры:

```
#nav {box-orient: horizontal;}
#sidebar {box-orient: vertical;}
```

Примечание:

Это свойство определено в спецификации модуля Flexible Box 2009 года. В новой версии этой спецификации предполагается сделать данное свойство не рекомендованным к применению, после чего поддержка в браузерах будет, вероятнее всего, прекращена.

box-pack

Значения:

start | **end** | **center** | **justify**

Исходное значение:

start

Применяется:

К элементам со значением **box** или **inline-box** свойства display.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок размещения гибких блоков, если их суммарные размеры по оси ориентации (см. приведенное выше описание свойства box-orient) оказываются меньше общей величины доступного пространства.

Примеры:

```
#gallery {box-pack: center;}
.subcolumns {box-pack: left;}
```

Примечание:

Это свойство определено в спецификации модуля Flexible Box 2009 года. В новой версии этой спецификации предполагается сделать данное свойство не рекомендованным к применению, после чего поддержка в браузерах будет, вероятнее всего, прекращена.

box-shadow

Значения:

none | <shadow> [, <shadow>]*

Расширения:

<shadow>

inset? && [<length>{**2**,**4**} && <color>?]

Исходное значение:

`none`

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение, но с абсолютными длинами и вычисляемыми цветами.

Описание:

Определяет одну или более несколько, отбрасываемых формой блока элемента. Данное свойство позволяет определить внешнюю (т.е. падающую) или внутреннюю тень, причем последнюю с помощью ключевого слова **`inset`**. Если же это ключевое слово не указано, тень оказывается внешней.

Четыре значения длины могут быть объявлены для данного свойства в следующем порядке: смещение по горизонтали, смещение по вертикали, расстояние размытия и расстояние распространения тени. При положительных значениях обоих видов смещения тень отбрасывается вниз и вправо, а при отрицательных значениях — вверх и влево. При положительных значениях расстояния распространения тени ее размеры увеличиваются, а при отрицательных значениях — уменьшаются. И наконец расстояние размытия не может иметь отрицательных значений.

Следует, однако, иметь в виду, что тени ограничиваются краями границ элемента. Следовательно, внешняя тень воспроизводится только до внешнего края границы. На полупрозрачном или полностью прозрачном фоне элемента внешняя тень от него *не* проявляется. А внутренняя тень видна только на самом краю границы и вообще не выходит за его пределы.

Примеры:

```
h1 {box-shadow: 5px 10px gray;}
table th {
    box-shadow: inset 0.5em 0.75em 5px -2px
    rgba(255,0,0,0.5);}
```

box-sizing

Значения:

content-box | border-box

Исходное значение:

content-box

Применяется:

Ко всем элементам со свойствами `width` и `height`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, должны ли свойства `height` и `width` элемента задавать размеры (высоту и ширину) блока содержимого (исторически сложившееся поведение) или блока границы. Если речь идет о блоке границы, то значение свойства `width` определяет расстояние от левого внешнего края границы до ее правого внешнего края. Аналогично свойство `height` определяет расстояние от верхнего внешнего края границы до ее нижнего внешнего края. Ширина любого внутреннего поля или границы "вычитается" из этих размеров, а не "прибавляется" к ним согласно исторически сложившемуся поведению. Так, если имеется объявление

```
body {box-sizing: border-box; width: 880px;
      padding: 0 20px;}
```

то ширина области содержимого составит **840** пикселей (**880 - 20 - 20**).

Пример:

```
body {box-sizing: border-box;}
```

caption-side

Значения:

top | **bottom**

Исходное значение:

top

Применяется:

К элементам со значением **table-caption** свойства display.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет расположение заголовка таблицы относительно блока таблицы. Заголовок воспроизводится так, как будто он является блочным элементом, расположенным непосредственно перед таблицей или после нее.

Пример:

```
caption {caption-side: top;}
```

Примечание:

Значения left и right данного свойства появились в спецификации CSS2, но были исключены из спецификации CSS2.1

из-за отсутствия их широкой поддержки (хотя в некоторых версиях браузера Firefox значения `left` и `right` данного свойства все еще поддерживаются).

clear

Значения:

left | right | both | none

Исходное значение:

none

Применяется:

К блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, на какой именно стороне (или сторонах) элемента нельзя располагать свободно перемещаемый элемент. При обычной разметке очищаемого элемента свободно перемещаемый элемент появляется у очищенной стороны, а очищаемый элемент смещается вниз до тех пор, пока не расположится ниже свободно перемещаемого элемента (т.е. очистится). В спецификациях CSS1 и CSS2 это делалось автоматически благодаря увеличению верхнего отступа очищаемого элемента. А в спецификации CSS2.1 добавлялся просвет над верхним отступом элемента, тогда как сам отступ не изменялся. Но в любом случае верхний внешний край границы элемента оказывается сразу же под внешним краем нижнего отступа свободно перемещаемого элемента на объявленной стороне.

Примеры:

```
h1 {clear: both;}
p + h3 {clear: right;}
```

clip

Значения:

rect(*top*, *right*, *bottom*, *left*) | **auto**

Исходное значение:

auto

Применяется:

К абсолютно располагаемым элементам (в спецификации CSS2 свойство `clip` применяется к блочным и заменяемым элементам).

Наследуется:

Нет.

Вычисляемое значение:

Вычисляемая длина четырех сторон, представляющих края отсекающего прямоугольника, а иначе — такое же, как и объявляемое значение.

Описание:

Определяет отсекающий прямоугольник, внутри которого отображается содержимое абсолютно располагаемого элемента. А содержимое за пределами области отсечения интерпретируется в соответствии со значением свойства `overflow`. Область отсечения может быть меньше или больше области содержимого элемента. В последнем случае она определяется отрицательными значениями длины сторон отсекающего прямоугольника.

В современных браузерах область отсечения определяется значением функции `rect()`, обозначающим смещение верхнего, правого, нижнего и левого краев областей отсечения

относительно левого верхнего угла элемента. Следовательно, значение функции `rect(5px, 10px, 40px, 5px)` обозначает расположение верхнего края области отсечения на **5** пикселей ниже верхнего края элемента, правого края области отсечения — на **10** пикселей правее левого края элемента, нижнего края области отсечения — на **40** пикселей ниже верхнего края элемента, а левого края области отсечения — на **5** пикселей правее левого края элемента.

Примеры:

```
div.sidebar {overflow: scroll; clip: 0 0 5em 10em;}
img.tiny {overflow: hidden; clip: 5px 5px 20px 20px;}
```

color

Значения:

`<color>`

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

См. приведенное ниже описание.

Описание:

Определяет цвет переднего плана элемента, что в HTML-разметке означает текст элемента. Свойство `color` не оказывает влияния на растровые изображения. Оно задает также цвет любых границ элемента, если только оно не замещается

свойством border-color или одним из свойств цвета краев границы (border-top-color и т.д.).

Если цвет обозначается ключевыми словами (например, **navy**) или шестнадцатеричными значениями RGB (например, **#008800** или **#080**), то вычисляемое значение данного свойства равно эквивалентному значению функции rgb(). Если же задано значение **transparent**, то вычисляемое значение данного свойства равно значению функции rgba(0,0,0,0). А если задано значение **currentColor**, то вычисляемое значение данного свойства равно **inherit**. Для всех остальных значений цвета вычисляемое значение данного свойства остается таким же, как и объявляемое.

Примеры:

```
strong {color: rgb(255,128,128);}
h3 {color: navy;}
p.warning {color: #ff0000;}
pre.pastoral {color: rgba(0%,100%,0%,0.33334);}
```

column-count

Значения:

<integer> | **auto**

Исходное значение:

auto

Применяется:

К незаменяемым блочным элементам, кроме табличных элементов, к ячейкам таблицы и внутристрочным блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет количество столбцов, используемых в многостолбцовой разметке элемента. Помимо исходного значения **auto**, устанавливаемого по умолчанию, допускаются только положительные ненулевые целочисленные значения данного свойства.

Пример:

```
body {column-count: 2;}
```

column-fill

Значения:

auto | **balance**

Исходное значение:

balance

Применяется:

К элементам, размещаемым с помощью нескольких столбцов.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли уравновешивать (или не уравновешивать) столбцы по высоте при разметке элемента с помощью нескольких столбцов. Значение этого свойства удерживается в тех случаях, когда столбцы каким-то образом ограничены по

длине. Очевидным тому примером служит высота элемента, явно задаваемая с помощью свойства `height`. Значение **`auto`** данного свойства означает, что столбцы заполняются последовательно, т.е. каждый столбец заполняется по всей высоте до конца (недостаточно заполняется или перезаполняется по мере надобности).

Пример:

```
body {height: 50em; column-fill: auto;}
```

column-gap

Значения:

`<length>` | **`normal`**

Исходное значение:

`normal`

Применяется:

К элементам, размещаемым с помощью нескольких столбцов.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет ширину промежутка между соседними столбцами в элементе, размечаемом с помощью нескольких столбцов. Любая линейка для разделения столбцов (см. ниже описание свойства `column-rule`) центруется в пределах каждого промежутка между столбцами. Длина промежутка между столбцами не может быть отрицательной.

Пример:

```
body {column-gap: 2em;}
```

column-rule

Значения:

```
<'column-rule-width'> || <'border-style'> || [ <color>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

К элементам, размещаемым с помощью нескольких столбцов.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Это сокращенная форма свойства, определяющая ширину, стиль и цвет так называемых *линеек* (т.е. вертикальных линий), воспроизводимых между столбцами в элементе, размечаемом с помощью нескольких столбцов. Любое пропущенное значение устанавливается равным значению по умолчанию соответствующего свойства. Следует, однако, иметь в виду, что если стиль границы не определяется, то по умолчанию он принимает значение **none** и линейка для разделения столбцов не воспроизводится.

Примеры:

```
#d01 {column-rule: 5px solid red;}
#d02 {column-rule: 2em dashed green;}
```

column-rule-color

Значения:

`<color>`

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

К элементам, размещаемым с помощью нескольких столбцов.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет цвет так называемых *линеек* (т.е. вертикальных линий), воспроизводимых между столбцами в элементе, размечаемом с помощью нескольких столбцов.

Примеры:

```
#d01 {column-rule-color: red;}
#d02 {column-rule-color: green;}
```

column-rule-style

Значения:

`<'border-style'>`

Исходное значение:

none

Применяется:

К элементам, размещаемым с помощью нескольких столбцов.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет стиль так называемых *линеек* (т.е. вертикальных линий), отображаемых между столбцами в элементе, размечаемом с помощью нескольких столбцов. Значения этого свойства являются такими же, как и у свойства `border-style`. Любое из значений **`none`** (по умолчанию) или **`hidden`** означает, что линейки для разделения столбцов не отображаются.

Примеры:

```
#d01 {column-rule-style: solid;}
#d02 {column-rule-style: dashed;}
```

column-rule-width

Значения:

`thin` | **`medium`** | **`thick`** | `<length>`

Исходное значение:

`medium`

Применяется:

К элементам, размещаемым с помощью нескольких столбцов.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютная длина. Нулевое значение (**`0`**), если задан стиль границы **`none`** или **`hidden`**.

Описание:

Определяет ширину так называемых *линеек* (т.е. вертикальных линий), отображаемых между столбцами в элементе, размечаемом с помощью нескольких столбцов.

Примеры:

```
#d01 {column-rule-width: 5px;}
#d02 {column-rule-width: 2em;}
```

column-span

Значения:

none | **all**

Исходное значение:

none

Применяется:

К статически размещаемым, не свободно перемещаемым элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет количество столбцов, на которое простирается элемент, размечаемый с помощью нескольких столбцов. Для этого имеются только две возможности: вообще не простирать элемент ни на один из столбцов (значение **none**) или же простирать его на все столбцы (значение **all**). Если элемент простирается на несколько столбцов, то содержимое, которое находится перед этим элементом, равномерно распределяется по столбцам над ним. А содержимое, которое следует после

данного элемента, равномерно распределяется по столбцам под ним.

Пример:

```
h2 {column-span: all;}
```

column-width

Значения:

<integer> | **auto**

Исходное значение:

auto

Применяется:

К незаменяемым блочным элементам, кроме табличных элементов, к ячейкам таблицы и внутристрочным блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютная длина.

Описание:

Определяет абсолютную ширину столбцов в элементе, размечаемом с помощью нескольких столбцов. Это означает *оптимальную* ширину столбцов, а следовательно, пользовательские агенты могут видоизменить ее значение или пренебречь ею, если сочтут нужным. Очевидным тому примером служит несоответствие общей ширины столбов и промежутков между ними ширине многостолбцового элемента. И это в какой-то степени похоже на изменение ширины ячеек таблицы с целью подогнать столбцы по общей ширине таблицы.

Значения абсолютной длины должны быть больше нуля. Но по какой-то непонятной причине указывать эти значения в процентах не разрешается.

Пример:

```
body {column-width: 23em;}
```

columns

Значения:

```
<'column-width'> || <'column-count'>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

К незаменяемым блочным элементам, кроме табличных элементов, к ячейкам таблицы и внутристрочным блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Это сокращенная форма свойства, определяющего количество и ширину столбцов в элементе, размечаемом с помощью несколько столбцов. Пропущенные значения устанавливаются равными значениям по умолчанию соответствующих свойств.

Примеры:

```
body {columns: 3 23em;}
div {columns: 200px 5;}
```

content

Значения:

normal | **none** | [<string> | <uri> | <counter> | **attr**(<identifier>) | **open-quote** | **close-quote** | **no-open-quote** | **no-close-quote**]+

Исходное значение:

normal

Применяется:

К псевдоэлементам ::before и ::after.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютный URI для значений <uri> или результирующая символьная строка для ссылок на атрибуты, а иначе — такое же, как и объявляемое значение.

Описание:

Определяет расположение генерируемого содержимого до или после элемента. По умолчанию это, скорее всего, внутристрочное содержимое, но тип блока, создаваемого для содержимого, определяется с помощью свойства display.

Примеры:

```
p::before {content: "Paragraph...";}
a[href]::after {content: "(" attr(href) ")";
    font-size: smaller;}
```

counter-increment

Значения:

`[<identifier> <integer>? ]+ | none`

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

С помощью этого свойства счетчики могут инкрементироваться (или декрементироваться) на любое значение: положительное, отрицательное или нулевое (**0**). Если тип `<integer>` значения не указан, то по умолчанию устанавливает значение **1** данного свойства.

Примеры:

```
h1 {counter-increment: section;}
*.backward li {counter-increment: counter -1;}
```

counter-reset

Значения:

`[<identifier> <integer>? ]+ | none`

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

С помощью этого свойства счетчики могут сбрасываться (или устанавливаться в первый раз), принимая любое значение: положительное или отрицательное. Если тип `<integer>` значения не указан, то по умолчанию устанавливает нулевое значение (`0`) данного свойства.

Примеры:

```
h1 {counter-reset: section;}
h2 {counter-reset: subsec 1;}
```

cursor

Значения:

`[<uri> [<number> <number>]?,]* [auto | default | auto | default | none | context-menu | help | pointer | progress | wait | cell | cross hair | text | vertical-text | alias | copy | move | no-drop | notallowed | e-resize | n-resize | ne-resize | nw-resize | s-resize | se-resize | sw-resize | w-resize | ew-resize | ns-resize | nesw-resize | nwse-resize | col-resize | row-resize | all-scroll]`

Исходное значение:

`auto`

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Единственный абсолютный URI с дополнительными, хотя и не обязательными, координатами *X*,*Y* для значений `<uri>`, если эти значения приводятся к поддерживаемому типу файлов, а иначе — такое же значение, как и объявляемое ключевым словом.

Описание:

Определяет форму курсора, появляющегося на экране при размещении указателя мыши на границе элемента, хотя в спецификации CSS2.1 край этой границы не определяется. Не следует, однако, забывать, что пользователи обычно привыкают к форме курсора, и поэтому ее неоправданное изменение может легко привести их в замешательство. Такое может, например, произойти, если изменить состояние курсора на **`pointer`**, когда пользователь наведет его на неинтерактивный элемент.

Следует также иметь в виду, что синтаксис значений данного свойства допускает указание необязательных значений URI, хотя ключевое слово должно быть указано обязательно. Следовательно, можно указать любое количество URI внешних ресурсов для курсора, но соответствующее значение должно *непременно* оканчиваться ключевым словом. В отсутствие ключевого слова пользовательские агенты, поддерживающие данное свойство, полностью опустят его объявление.

В спецификации CSS3 допускается снабжать значение `<uri>` двумя дополнительными, хотя и не обязательными числами. Эти числа обозначают координаты *X*,*Y* "активной точки" курсора, т.е. такой точки расположения курсора, которая служит для определения состояний наведения, активных действий и т.п. Если эти числа не предоставляются, а изображение курсора не содержит "внутренней активной точки", как сказано в спецификации, то в качестве такой точки выбирается левый верхний угол изображения, что равнозначно обозначению **`0 0`**.

Следует также иметь в виду, что эти числа обозначают безразмерные величины и интерпретируются относительно "системы координат курсора", как, опять же, сказано в спецификации.

Примеры:

```
a.moreinfo {cursor: help;}
a[href].external {cursor: url(globe.png), auto;}
```

direction

Значения:

ltr | rtl

Исходное значение:

ltr

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет базовое направление записи блоков, а также направление встраивания и замещения текста для двунаправленного алгоритма в Уникоде. Кроме того, данное свойство изменяет способ интерпретации целого ряда свойств и решений, принимаемых при разметке, включая размещение ячеек в строке таблицы и выбор алгоритмов разметки структурных блоков, но не ограничиваясь только этим.

По самым разным причинам вместо свойства `direction` настоятельно рекомендуется использовать атрибут `dir` разметки HTML-документов. Пользовательским агентам, не

поддерживающим двунаправленный текст, разрешается пренебрегать этим свойством.

Примеры:

```
*:lang(en) {direction: ltr;}
*:lang(ar) {direction: rtl;}
```

display

Значения в CSS2.1:

none | inline | block | inline-block | list-item | table | inlinetable | table-row-group | table-header-group | table-footergroup | table-row | table-column-group | table-column | tablecell | table-caption

Значения в CSS3:

none | inline | block | inline-block | list-item | run-in | compact | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | tablecolumn | table-cell | table-caption | ruby | ruby-base | ruby-text | ruby-base-container | ruby-text-container

Исходное значение:

inline

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Отличается для свободно перемещаемых, располагаемых и корневых элементов (см. раздел 9.7 спецификации CSS2.1), а иначе — такое же, как и объявляемое значение.

Описание:

Определяет вид блока отображения, формируемого во время разметки элемента. Использование свойства `display` в HTML-документах не всегда оправдано, поскольку это нарушает иерархию отображения, уже определенную в HTML-разметке, хотя данное свойство может быть очень полезным. Что же касается XML-документов, в которых подобная встроенная иерархия отсутствует, то для них свойство `display` просто незаменимо.

Значение **`none`** данного свойства нередко служит для сокрытия элементов, поскольку оно удаляет элемент и все производные от него элементы из представления. И это справедливо не только для визуальных, но и для всех остальных носителей информации. Следовательно, объявление `display: none` в разметке элемента не даст возможности озвучить этот элемент в озвучивающем браузере.

Значение **`run-in`** данного свойства долго входило в спецификацию CSS2.1, но в начале 2011 года было исключено из-за несогласованностей в разных браузерах. Тем не менее оно попрежнему входит в спецификацию CSS3. А значения **`compact`** и **`marker`** появились в спецификации CSS2, но были исключены из спецификации CSS2.1 из-за отсутствия широкой поддержки.

Примеры:

```
h1 {display: block;}
li {display: list-item;}
img {display: inline;}
.hide {display: none;}
tr {display: table-row;}
```

empty-cells

Значения:

`show | hide`

Исходное значение:

show

Применяется:

К элементам со значением **table-cell** свойства display.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет представление ячеек таблицы без содержимого. Если такие ячейки отображаются, то воспроизводятся их границы и задний план. Это свойство принимается во внимание лишь в том случае, если задано значение **separate** свойства border-collapse, а иначе оно просто игнорируется.

Пример:

```
th, td {empty-cells: show;}
```

float

Значения:

left | **right** | **none**

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет направление, в котором свободно перемещается элемент. По традиции это свойство применялось к изображениям, чтобы текст облекал их, но в CSS свободно перемещаться разрешается любому элементу. Свободно перемещаемый элемент будет формировать структурный блок независимо от типа элемента, для которого он предназначен. Для свободно перемещаемых незаменяемых элементов должна быть явно задана их ширина, иначе они станут максимально узкими. Свободное перемещение элементов обычно в достаточной степени поддерживается во всех браузерах, но сам характер таких элементов может привести к неожиданным результатам, когда они применяются в качестве механизма разметки страниц. И это объясняется, главным образом, незначительными отличиями в интерпретации определений наподобие "элементы станут максимально узкими".

Примеры:

```
img.figure {float: left;}
p.sidebar {float: right; width: 15em;}
```

font

Значения:

```
[[ <'font-style'> || <'font-variant'> || <'font-weight'>
]? <'fontsize'> [ / <'line-height'> ]? <'font-family'>] |
caption | icon | menu | message-box | small-caption | status-bar
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

Ко всем элементам.

Наследуется:

Да.

Задание в процентах:

Вычисляется относительно родительского элемента для размера шрифта <font-size> и относительно размера шрифта <font-size> элемента для высоты строки <line-height>.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, предназначенная для установки сразу всех свойств шрифта размечаемого элемента. Может также использоваться для установки шрифта размечаемого элемента с учетом вычислительной среды пользователя, для чего служит ключевое слово **icon**. Если ключевые слова отсутствуют в объявлении этого свойства, то в качестве минимальных типографских параметров должны быть *непременно* заданы размер и гарнитура шрифта (именно в *таком* порядке), а любой типографский параметр, не обозначаемый ключевым словом, *должен* оканчиваться гарнитурой шрифта. В противном случае объявление шрифта будет проигнорировано.

Примеры:

```
p {font: small-caps italic bold small/
   1.25em Helvetica,sans-serif;}
p.example {font: 14px Arial;} /* технически правильно,
  хотя общая форма font-families зарезервирована
  для дальнейшего использования */
.figure span {font: icon;}
```

font-family

Значения:

```
[ <family-name> | <generic-family> ] [, <family-name>|
<genericfamily> ]*
```

Расширения

<generic-family>

serif | **sans-serif** | **monospace** | **cursive** | **fantasy**

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет гарнитуру шрифта, применяемую для отображения текста элемента. Следует, однако, иметь в виду, что применение конкретной гарнитуры шрифта (например, Geneva) полностью зависит от ее наличия в системе пользователя или в виде загружаемого файла, а также от присутствия в ней глифов, необходимых для отображения содержимого. Поэтому в качестве запасного варианта настоятельно рекомендуется указывать наименования типичных гарнитур шрифтов. Наименования шрифтов, содержащие пробелы или небуквенные символы, должны быть заключены в кавычки, чтобы свести к минимуму возможные недоразумения. В то же время наименования запасных типичных гарнитур *вообще не* следует заключать в кавычки.

Примеры:

```
p {font-family: Helvetica, Arial, sans-serif;}
li {font-family: Times, TimesNR, "New Century Schoolbook",
    serif;}
pre {font-family: Courier, "Courier New", "Andale Mono",
Monaco, monospace;}
```

font-size

Значения:

xx-small | **x-small** | **small** | **medium** | **large** | **x-large** | **xx-large** | **smaller** | **larger** | <length> | <percentage>

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Да.

Задание в процентах:

По размеру шрифта родительского элемента.

Вычисляемое значение:

Абсолютная длина.

Описание:

Определяет размер шрифта как абсолютный или относительный размер, а также в виде длины, вычисляемой как абсолютно, так и в процентах. Указывать отрицательные значения абсолютной длины и в процентах не разрешается. В присваивании размера шрифта скрывается немало разных опасностей. Так, в веб-разработке не рекомендуется указывать размер шрифта в пунктах из-за отсутствия определенной взаимосвязи между пунктами и пикселями на экране монитора.

Исторически сложившиеся недоразумения заключались в том, что установка значения **medium** в свойстве font-size приводила к разным результатам в первых версиях браузеров Internet Explorer и Navigator 4.x. Некоторые из этих недоразумений подробно рассматриваются в главе 5 упоминавшейся ранее книги *CSS: The Definitive Guide*, третье издание (издательство O'Reilly), а дополнительные сведения по данному вопросу

приведены по адресу http://style.cleverchimp.com/. Для достижения наилучших результатов рекомендуется указывать размер шрифта в процентах или кегельных шпациях, а в крайнем случае — в пикселях, хотя такой подход накладывает серьезные ограничения на специальные возможности, поскольку это препятствует пользователям изменять размер текста в браузере Internet Explorer под управлением Windows, даже в том случае, если этот текст слишком мелкий, чтобы быть удобочитаемым. А в большинстве других браузеров пользователям разрешается изменять размер текста независимо от его исходного размера.

Примеры:

```
h2 {font-size: 200%;}
code {font-size: 0.9em;}
p.caption {font-size: 9px;}
```

font-size-adjust

Значения:

<number> | **none**

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет *аспект шрифта* элемента, употребляемый для масштабирования шрифтов таким образом, чтобы они точнее соответствовали друг другу в тех случаях, когда применяются запасные шрифты. Аспект шрифта определяется путем деления x-высоты шрифта на его размер.

С помощью свойства `font-size-adjust` размер шрифта корректируется по x-высоте, т.е. по высоте строчных букв шрифта. Рассмотрим в качестве примера гипотетический шрифт (назовем его, скажем, CSSType) размером **100** пикселей и x-высотой **60** пикселей, которая означает, что высота строчной буквы ***x*** составляет **60** пикселей. Соответствующее значение свойства `font-size-adjust` для шрифта CSSType будет равно **0,6** (**60 / 100**). Так, объявление

```
p {font: 20px "CSSType", sans-serif;
   font-size-adjust: 0.6;}
```

означает, что размер текста в абзаце должен быть таким, чтобы строчные буквы оказались высотой **12** пикселей (**20 × 0,6 = = 12**) независимо от применяемой гарнитуры шрифта. Если же шрифт CSSType отсутствует, а пользовательский агент обращается к запасному шрифту, например Helvetica, то размер текста, набранного шрифтом Helvetica, также приводится к высоте строчных букв **12** пикселей, тогда как высота прописных букв будет соответствовать заданному размеру шрифта. Аспект шрифта Helvetica составляет **0,53**, и поэтому высота строчных букв этого шрифта будет равна **22,6** пикселя (или округленной величине, если пользовательский агент не в состоянии обращаться с дробными пикселями). Если же применяется какой-нибудь другой шрифт без засечек с аспектом **0,7**, то высота прописных букв составит **17,1** пикселя, а высота строчных букв — **12** пикселей.

Примеры:

```
body {font-family: Helvetica, sans-serif;
      font-size-adjust: 0.53;}
```

font-style

Значения:

italic | **oblique** | **normal**

Исходное значение:

normal

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет вид начертания шрифта: курсивное, наклонное или обычное. Набранный курсивом текст обычно определяется как отдельное начертание в гарнитуре шрифта. Теоретически пользовательский агент может вычислить наклонное начертание шрифта из его обычного начертания. Но на практике пользовательские агенты редко отличают курсивный текст от наклонного, если они вообще на это способны, и почти всегда воспроизводят оба вида текста одинаково.

Примеры:

```
em {font-style: oblique;}
i {font-style: italic;}
```

font-variant

Значения:

small-caps | **normal**

Исходное значение:

normal

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, задается ли для текста стиль малых прописных букв. Теоретически пользовательский агент может вычислить начертание шрифта малыми прописными буквами из обычного начертания.

Примеры:

```
h3 {font-variant: small-caps;}
p {font-variant: normal;}
```

font-weight

Значения:

normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 |600 | 700 | 800 | 900

Исходное значение:

normal

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Одно из абсолютных числовых значений (**100** и т.д.) или же абсолютных числовых значений плюс относительные значения (**bolder** или **lighter**).

Описание:

Определяет вес шрифта, употребляемый при воспроизведении текста элемента. Числовое значение **400** этого свойства равнозначно ключевому слову **normal**, а числовое значение **700** — ключевому слову **bold**. Если же у шрифта имеются только два веса (обычный и полужирный), то в пределах от **100** до **500** своего веса шрифт будет обычным, а в пределах от **600** до **900** — полужирным. В общем, чем меньше числовое значение веса, тем светлее оказывается шрифт, а чем оно больше, тем шрифт темнее.

Примеры:

```
b {font-weight: 700;}
strong {font-weight: bold;}
.delicate {font-weight: lighter;}
```

height

Значения:

<length> | <percentage> | **auto**

Исходное значение:

auto

Применяется:

Ко всем элементам, кроме незаменяемых внутристрочных элементов, отдельных столбцов и групп столбцов таблицы.

Наследуется:

Нет.

Задание в процентах:

По высоте содержащего блока.

Вычисляемое значение:

Для исходных значений **auto** и в процентах такое же, как объявляемое значение, а иначе — абсолютная длина, если только данное свойство не применяется к элементу, и тогда оно равно **auto**.

Описание:

Определяет высоту области содержимого элемента или блока его границы в зависимости от значения свойства box-sizing. Указывать отрицательные значения абсолютной длины и в процентах не разрешается.

Примеры:

```
img.icon {height: 50px;}
h1 {height: 1.75em;}
```

left

Значения:

<length> | <percentage> | **auto**

Исходное значение:

auto

Применяется:

К размещаемым элементам, т.е. к таким элементам, для которых задано значение свойства position, отличающееся от **static**.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для относительно размещаемых элементов вычисляемое значение свойства **left** всегда равно значению свойства **right**; для статически размещаемых элементов (т.е. таких элементов, для которых задано значение **static** свойства position) вычисляемое значение равно **auto**; для значений длины оно равно соответствующей абсолютной длине; для значений в процентах — объявляемому значению, а иначе — **auto**.

Описание:

Определяет смещение внешнего края левого отступа размещаемого элемента относительно левого края содержащего его блока, а для относительно размещаемых элементов — расстояние, на которое элемент смещается вправо от своего исходного положения.

Примеры:

```
div#footer {position: fixed; left: 0;}
*.hanger {position: relative; left: -25px;}
```

letter-spacing

Значения:

```
[ normal | <length> | <percentage> ]{1,3}
```

Исходное значение:

normal

Применяется:

Ко всем элементам.

Наследуется:

Да.

Задание в процентах:

По ширине глифа пробела в Уникоде (**U+0020**) для начертания шрифта элемента.

Вычисляемое значение:

Для значений длины вычисляемое значение равно соответствующей абсолютной длине, а иначе — **normal**.

Описание:

Определяет величину пробела, вставляемого между блоками символов текста. Как правило, глифы символов оказываются более узкими, чем их блоки, а благодаря указываемым значениям длины можно изменять обычное расстояние между буквами. Таким образом, значение **normal** зачастую оказывается равнозначным нулевому значению (**0**). Разрешается указывать отрицательные значения абсолютной длины и в процентах. В таком случае буквы располагаются ближе одна к другой.

Три возможных значения данного свойства соответствуют минимальному, максимальному и оптимальному расстояниям между буквами. Если указаны два значения данного свойства, то первое из них соответствует минимальному и оптимальному расстояниям между буквами, а второе — максимальному расстоянию. Если же указано единственное значение данного свойства, то оно применяется для обозначения всех трех упомянутых выше расстояний между буквами. А если текст выравнивается, то пользовательский агент может при необходимости превысить максимальное расстояние между буквами, хотя оно и не должно быть меньше минимального расстояния. И наконец, если текст *не* выравнивается, то всегда применяется оптимальное расстояние между буквами.

Примеры:

```
p.spacious {letter-spacing: 6px;}
em {letter-spacing: 0.2em;}
p.cramped {letter-spacing: -0.5em;}
```

Примечание:

В спецификации CSS2.1 свойство `letter-spacing` принимает только одно значение: абсолютную длину или **`normal`**.

line-height

Значения:

`<length> | <percentage> | <number> | normal | none`

Исходное значение:

`normal`

Применяется:

Ко всем элементам, тем не менее читайте в приведенном ниже описании пояснения, касающиеся заменяемых и блочных элементов.

Наследуется:

Да.

Задание в процентах:

По размеру шрифта элемента.

Вычисляемое значение:

Для значений длины и значений в процентах вычисляемое значение равно соответствующей абсолютной длине, а иначе — такое же, как и объявляемое значение.

Описание:

Это свойство оказывает влияние на разметку построчных блоков. Если это свойство применяется к блочному элементу, оно определяет минимальное, а не максимальное расстояние между базовыми линиями текста в данном элементе. А если это свойство применяется к внутристрочному элементу, оно служит для определения интерлиньяжа в тексте данного элемента.

Разность вычисляемых значений свойств line-height и font-size, называемая в CSS *интерлиньяжем*, делится надвое и прибавляется к верхнему и нижнему краям каждого фрагмента содержимого в текстовой строке. Самым коротким блоком, в который можно заключить все подобные фрагменты содержимого, является построчный блок.

Исходное числовое значение определяет масштабный коэффициент, который наследуется вместо вычисляемого значения. Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
p {line-height: 1.5em;}
h2 {line-height: 200%;}
ul {line-height: 1.2;}
pre {line-height: 0.75em;}
```

Примечание:

Ключевое слово **none** было внедрено в спецификации CSS3, а его поддержка зависит от конкретного пользовательского агента.

list-style

Значения:

```
<list-style-type> || <list-style-image> || <list-style-
position>
```

Исходное значение:

См. ниже описание отдельных свойств.

Применяется:

К элементам со значением **list-item** свойства display.

Наследуется:

Да.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая тип маркера списка в виде знака или изображения, а также его приблизительное расположение. Данное свойство применяется к любому элементу со значением **list-item** свойства display и к наследующим от него элементам с аналогичным значением, и поэтому действие этого свойства распространяется только на элементы разметки li обычных HTML- и XHTML-документов.

Примеры:

```
ul {list-style: square url(bullet3.gif) outer;}
   /* значения наследуются элементами разметки li */
ol {list-style: upper-roman;}
```

list-style-image

Значения:

<uri> | **none**

Исходное значение:

none

Применяется:

К элементам со значением **list-item** свойства display.

Наследуется:

Да.

Вычисляемое значение:

Абсолютный URI для значений <uri>, а иначе — **none**.

Описание:

Определяет изображение, используемое в качестве маркера элемента маркированного или немаркированного списка.

Расположением изображения относительно содержимого элемента списка можно приблизительно управлять с помощью рассматриваемого далее свойства `list-style-position`.

Примеры:

```
ul {list-style-image: url(bullet3.gif);}
ul li {list-style-image: url(http://example.org/pix/
checkmark.png);}
```

list-style-position

Значения:

inside | **outside**

Исходное значение:

outside

Применяется:

К элементам со значением **list-item** свойства `display`.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет расположение маркера списка относительно содержимого элемента списка. Внешние маркеры размещаются на некотором расстоянии от края границы элемента списка, хотя это расстояние в CSS не определено. А внутренние маркеры интерпретируются как внутристрочные элементы, вставляемые в начале содержимого элемента списка.

Примеры:

```
li {list-style-position: outside;}
ol li {list-style-position: inside;}
```

list-style-type

Значения в CSS2.1:

disc | circle | square | decimal | decimal-leading-zero | upperalpha | lower-alpha | upper-latin | lower-latin | upper-roman | lower-roman | lower-greek | georgian | armenian | none

Значения в CSS3:

<glyph> | <algorithmic> | <numeric> | <alphabetic> | <symbolic> | <non-repeating> | **normal | none**

Расширения:

<glyph>

box | check | circle | diamond | disc | hyphen | square

<algorithmic>

armenian | cjk-ideographic | ethiopic-numeric | georgian | hebrew | japanese-formal | japanese-informal | lower-armenian | lowerroman | simp-chinese-formal | simp-chinese-informal | syriac | tamil | trad-chinese-formal | trad-chinese-informal | upperarmenian | upper-roman

<numeric>

arabic-indic | binary | bengali | cambodian | decimal | decimal-leading-zero | devanagari | gujarati | gurmukhi | kannada | khmer | lao | lower-hexadecimal | malayalam | mongolian | myanmar | octal | oriya | persian | telugu | tibetan | thai | upper-hexadecimal | urdu

<alphabetic>

afar | amharic | amharic-abegede | cjk-earthly-branch | cjkheavenly-stem | ethiopic | ethiopic-abegede | ethiopic-abegedeam-et | ethiopic-abegede-gez | ethiopic-abegede-ti-er | ethiopicabegede-ti-et |

ethiopic-halehame-aa-er | **ethiopic-halehame-aaet** |
ethiopic-halehame-am-et | **ethiopic-halehame-gez** |
ethiopichalehame-om-et | **ethiopic-halehame-sid-et** |
ethiopic-halehameso-et | **ethiopic-halehame-ti-er** |
ethiopic-halehame-ti-et | **ethiopic-halehame-tig** |
hangul | **hangul-consonant** | **hiragana** | **hiragana-iroha** |
katakana | **katakana-iroha** | **lower-alpha** | **lower-greek** |
lower-norwegian | **lower-latin** | **oromo** | **sidama** | **somali** |
tigre | **tigrinya-er** | **tigrinya-er-abegede** | **tigrinya-et** |
tigrinya-et-abegede | **upper-alpha** | **upper-greek** |
upper-norwegian | **upper-latin**

<symbolic>

asterisks | **footnotes**

<non-repeating>

circled-decimal | **circled-lower-latin** | **circled-upper-latin** |
dotted-decimal | **double-circled-decimal** | **filled-circleddecimal** |
parenthesised-decimal | **parenthesised-lower-latin**

Исходное значение:

disc

Применяется:

К элементам со значением **list-item** свойства display.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет тип маркерной системы, применяемой в представлении списка. В спецификации CSS3 предоставляется значительно расширенный ряд типов списков, хотя новые типы списков поддерживаются по-разному. Поэтому особого внимания требуют те типы списков, которые выходят за рамки спецификации CSS2.1.

Поведение, когда список, маркируемый в алфавитном порядке, исчерпывает буквы алфавита, не определено. Так, если

значение **upper-latin** данного свойства достигнет буквы **Z**, то в спецификации ничего сказано, каким должен быть следующий маркер списка. Это может быть маркер **AA** или **ZA**. И это может произойти независимо от применяемого алфавита. Следовательно, нет никакой гарантии, что разные пользовательские агенты будут действовать в подобных случаях согласованно.

Примеры:

```
ul {list-style-type: square;}
ol {list-style-type: lower-roman;}
```

Примечание:

На момент написания данной книги широкой поддержкой пользовались только следующие значения данного свойства, определенные в спецификации CSS2.1: **disc**, **circle**, **square**, **decimal**, **upper-alpha**, **lower-alpha**, **upperlatin**, **upper-roman** и **lower-roman**.

margin

Значения:

[<length> | <percentage> | **auto**]{**1**,**4**}

Исходное значение:

Не определено.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая ширину общего отступа для элемента или задающая ширину отступа по отдельным его сторонам. Соседние по вертикали отступы блочных элементов сводятся, тогда как отступы сверху и снизу у внутристрочных элементов, по существу, отсутствуют. А отступы слева и справа внутристрочных элементов не сводятся, как, впрочем, и отступы свободно перемещаемых элементов. Разрешается указывать отрицательные значения отступов, но делать это следует осторожно, поскольку отрицательные значения становятся причиной наложения одних элементов на другие или появления более широких элементов, чем родительские.

Примеры:

```
h1 {margin: 2ex;}
p {margin: auto;}
img {margin: 10px;}
```

margin-bottom

Значения:

<length> | <percentage> | **auto**

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет ширину нижнего отступа для элемента. Разрешается указывать отрицательные значения, хотя делать это следует осторожно (см. выше описание свойства `margin`).

Примеры:

```
ul {margin-bottom: 0.5in;}
h1 {margin-bottom: 2%;}
```

margin-left

Значения:

<length> | <percentage> | **auto**

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет ширину левого отступа для элемента. Разрешается указывать отрицательные значения, хотя делать это следует осторожно (см. выше описание свойства `margin`).

Примеры:

```
p {margin-left: 5%;}
pre {margin-left: 3em;}
```

margin-right

Значения:

`<length> | <percentage> | auto`

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет ширину правого отступа для элемента. Разрешается указывать отрицательные значения, хотя делать это следует осторожно (см. выше описание свойства `margin`).

Примеры:

```
img {margin-right: 30px;}
ol {margin-right: 5em;}
```

margin-top

Значения:

<length> | <percentage> | **auto**

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет ширину верхнего отступа для элемента. Разрешается указывать отрицательные значения, хотя делать это следует осторожно (см. выше описание свойства `margin`).

Примеры:

```
ul {margin-top: 0.5in;}
h3 {margin-top: 1.5em;}
```

max-height

Значения:

`<length> | <percentage> | none`

Исходное значение:

none

Применяется:

Ко всем элементам, кроме внутристрочных незаменяемых и табличных элементов.

Наследуется:

Нет.

Задание в процентах:

По высоте содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое; для значений длины оно равно соответствующей абсолютной длине, а иначе — **none**.

Описание:

Определяет максимальное ограничение, накладываемое на высоту элемента, хотя конкретная его высота зависит от значения свойства `box-sizing`. Следовательно, высота элемента может быть меньше объявленного значения, но не больше его. Отрицательные значения в данном свойстве не допускаются.

Пример:

```
div#footer {max-height: 3em;}
```

max-width

Значения:

`<length> | <percentage> | none`

Исходное значение:

none

Применяется:

Ко всем элементам, кроме внутристрочных незаменяемых и табличных элементов.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое; для значений длины оно равно соответствующей абсолютной длине, а иначе — **none**.

Описание:

Определяет максимальное ограничение, накладываемое на ширину элемента, хотя конкретная его ширина зависит от значения свойства `box-sizing`. Следовательно, ширина элемента может быть меньше объявленного значения, но не больше его. Отрицательные значения в данном свойстве не допускаются.

Пример:

```
#sidebar img {width: 50px; max-width: 100%;}
```

min-height

Значения:

`<length> | <percentage>`

Исходное значение:

0

Применяется:

Ко всем элементам, кроме внутристрочных незаменяемых и табличных элементов.

Наследуется:

Нет.

Задание в процентах:

По высоте содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет минимальное ограничение, накладываемое на высоту элемента, хотя конкретная его высота зависит от значения свойства `box-sizing`. Следовательно, высота элемента может быть больше объявленного значения, но не меньше его. Отрицательные значения в данном свойстве не допускаются.

Пример:

```
div#footer {min-height: 1em;}
```

min-width

Значения:

```
<length> | <percentage>
```

Исходное значение:

0

Применяется:

Ко всем элементам, кроме внутристрочных незаменяемых и табличных элементов.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое; для значений длины оно равно соответствующей абсолютной длине; а иначе — **none**.

Описание:

Определяет минимальное ограничение, накладываемое на ширину элемента, хотя конкретная его ширина зависит от значения свойства `box-sizing`. Следовательно, ширина элемента может быть больше объявленного значения, но не меньше его. Отрицательные значения в данном свойстве не допускаются.

Пример:

```
div.aside {float: right; width: 13em; max-width: 33%;}
```

opacity

Значения:

`<number>`

Исходное значение:

1

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение (или ограниченное значение, если объявляемое значение должно быть ограничено).

Описание:

Определяет степень непрозрачности элемента по числу в пределах от **0** до **1** включительно. Любые значения вне этих пределов ограничиваются до ближайшего предела (**0** или **1**). Это свойство оказывает влияние на каждую видимую часть элемента. Если же требуется сделать полупрозрачным содержимое элемента, но не задний план, или наоборот, то лучше воспользоваться функцией `rgba()`, чтобы задать цвета с альфа-каналом, определяющим степень непрозрачности цвета.

Если для элемента задано нулевое (**0**) значение свойства `opacity`, он, по существу, становится невидимым и может вообще не реагировать на события от мыши или другие события, наступающие в объектной модели документа (DOM). В связи с особенностями воспроизведения полупрозрачных элементов каждый элемент со значением свойства `opacity` меньше **1,0** создает свой контекст наложения, даже если его размещение не предполагается. По аналогичным причинам в абсолютно размещаемом элементе со значением свойства `opacity` меньше **1,0** и значением **auto** свойства `z-index` принудительно устанавливается нулевое (**0**) значение свойства `z-index`.

Примеры:

```
h2 {opacity: 0.8;}
.hideme {opacity: 0;}
```

outline

Значения:

```
<'outline-color'> || <'outline-style'> || <'outline-width'>
```

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая общий контур элемента. Чаще всего контуры служат для обозначения элементов формы или гиперссылок, находящихся в фокусе ввода, т.е. готовых принять вводимые пользователем данные. Контуры могут иметь неправильную форму, и независимо от своей толщины они не изменяют и не оказывают иного воздействия на размещение элементов.

Примеры:

```
*[href]:focus {outline: 2px dashed invert;}
form:focus {outline: outset cyan 0.25em;}
```

outline-color

Значения:

<color> | **invert**

Исходное значение:

invert

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет цвет видимых участков общего контура элемента. Не следует, однако, забывать, что значение свойства `outline-style` должно отличаться от **`none`**, чтобы появилась любая видимая граница. Пользовательским агентам разрешается игнорировать значение **`invert`** данного свойства на тех платформах, на которых инверсия цвета не поддерживается. В таком случае цвет контура элемента по умолчанию принимает значение свойства **`color`**.

Примеры:

```
*[href]:focus {outline-color: invert;}
form:focus {outline-color: cyan;}
```

outline-offset

Значения:

`<length>`

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютное значение длины.

Описание:

Определяет смещение относительно внешнего края границы и внутреннего края контура. Это свойство может принимать только одно значение длины. Оно применяется в равной степени ко всем сторонам контура. Допускается указывать отрицательные значения данного свойства. В этом случае контур сокращается вовнутрь к центру элемента. Следует, однако, иметь в виду, что свойство `outline-offset` нельзя устанавливать посредством сокращенной формы свойства `outline`.

Примеры:

```
*[href]:focus {outline-offset: 0.33em;}
form:focus {outline-offset: -1px;}
```

outline-style

Значения:

none | **dotted** | **dashed** | **solid** | **double** | **groove** | **ridge** | **inset** | **outset**

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет стиль общей границы элемента. Задаваемый в этом свойстве стиль должен отличаться от **none**, чтобы появился какой-нибудь контур.

Примеры:

```
*[href]:focus {outline-style: dashed;}
form:focus {outline-style: outset;}
```

outline-width

Значения:

thin | **medium** | **thick** | <length>

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютная длина. Нулевое значение (**0**), если задан стиль границы **none** или **hidden**.

Описание:

Определяет толщину общего контура элемента. Указанная толщина окажет воздействие на заданный контур только в том случае, если значение свойства `outline-style` отличается от **none**. Если же оно принимает значение **none**, то толщина контура, по существу, устанавливается равной нулю (**0**). Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
*[href]:focus {outline-width: 2px;}
form:focus {outline-width: 0.25em;}
```

overflow

Значения:

[**visible** | **hidden** | **scroll** | **auto** | **no-display** | **no-content**]{1,2}

Исходное значение:

Не определено для сокращенных форм свойств (**visible** в спецификации CSS2.1).

Применяется:

К незаменяемым элементам со значением **block** или **inline-block** свойства `display`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Это сокращенная форма свойства, определяющая происходящее с содержимым при переполнении области содержимого элемента. Так, если установлено значение **scroll** данного свойства, пользовательские агенты должны предоставить механизм прокрутки независимо от того, требуется ли он. В частности, полосы прокрутки будут появляться даже в том случае, если все содержимое может поместиться в блоке элемента. Если же указываются два значения данного свойства, то первое из них определяет значение свойства `overflow-x`, а второе — значение свойства `overflow-y`. В противном случае оба эти значения определяются одним и тем же значением данного свойства.

Примеры:

```
#masthead {overflow: hidden;}
object {overflow: visible scroll;}
```

Примечание:

В спецификации CSS2.1 свойство `overflow` определено как автономная, а не сокращенная форма свойства. Поддержка значений `no-display` и `no-content` данного свойства в современных браузерах отсутствует.

overflow-x

Значения:

visible | **hidden** | **scroll** | **auto** | **no-display** | **no-content**

Исходное значение:

visible

Применяется:

К незаменяемым элементам со значением **block** или **inline-block** свойства `display`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет поведение при переполнении по горизонтальной оси (X) элемента, т.е. от левого до правого края элемента.

Примеры:

```
#masthead {overflow-x: hidden;}
object {overflow-x: visible;}
```

Примечание:

На момент написания данной книги поддержка значений **no-display** и **no-content** данного свойства в основных браузерах отсутствовала.

overflow-y

Значения:

visible | **hidden** | **scroll** | **auto** | **no-display** | **no-content**

Исходное значение:

visible

Применяется:

К незаменяемым элементам со значением **block** или **inline-block** свойства display.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет поведение при переполнении по вертикальной оси (Y) элемента, т.е. от верхнего до нижнего края элемента.

Примеры:

```
#masthead {overflow-y: hidden;}
object {overflow-y: scroll;}
```

Примечание:

На момент написания данной книги поддержка значений **no-display** и **no-content** данного свойства в основных браузерах отсутствовала.

padding

Значения:

```
[ <length> | <percentage> ]{1,4}
```

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая ширину общего внутреннего поля элемента или задающая ширину внутреннего поля по каждой из сторон элемента в отдельности. Внутреннее поле, задаваемое для внутристрочных незаменяемых элементов, не оказывает никакого влияния на вычисление высоты строки. Следовательно, такой элемент с внутренним полем и задним планом может явно простираться на другие строки, а возможно, и перекрывать другое содержимое. Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
img {padding: 10px;}
h1 {padding: 2ex 0.33em;}
pre {padding: 0.75em 0.5em 1em 0.5em;}
```

padding-bottom

Значения:

```
<length> | <percentage>
```

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет ширину нижнего внутреннего поля элемента. Нижнее внутреннее поле, задаваемое для внутристрочных незаменяемых элементов, не оказывает никакого влияния на вычисление высоты строки. Следовательно, такой элемент с нижним внутренним полем и задним планом может явно простираться на другие строки, а возможно, и перекрывать другое содержимое. Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
ul {padding-bottom: 0.5in;}
h1 {padding-bottom: 2%;}
```

padding-left

Значения:

```
<length> | <percentage>
```

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет ширину левого внутреннего поля элемента. Левое внутреннее поле, задаваемое для внутристрочного элемента, появится только на левом краю первого строкового блока, сформированного элементом. Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
p {padding-left: 5%;}
pre {padding-left: 3em;}
```

padding-right

Значения:

```
<length> | <percentage>
```

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет ширину правого внутреннего поля элемента. Правое внутреннее поле, задаваемое для внутристрочного элемента, появится только на правом краю первого строкового блока, сформированного элементом. Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
img {padding-right: 30px;}
ol {padding-right: 5em;}
```

padding-top

Значения:

```
<length> | <percentage>
```

Исходное значение:

0

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет ширину верхнего внутреннего поля элемента. Верхнее внутреннее поле, задаваемое для внутристрочных незаменяемых элементов, не оказывает никакого влияния на вычисление высоты строки. Следовательно, такой элемент с верхним внутренним полем и задним планом может явно простираться на другие строки, а возможно, и перекрывать другое содержимое. Отрицательные значения в данном свойстве не допускаются.

Примеры:

```
ul {padding-top: 0.5in;}
h3 {padding-top: 1.5em;}
```

perspective

Значения:

none | <number>

Исходное значение:

none

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет величину видимой трехмерной перспективы преобразуемых потомков, а не самого элемента. Указываемые числовые значения данного свойства определяют глубину сокращения перспективы в пикселях. Чем меньше эти числовые значения, тем сильнее искажается перспектива. Отрицательные значения данного свойства интерпретируются таким же образом, как и значение **none**.

Примеры:

```
body {perspective: 250;} /* средняя перспектива' */
#wrapper {perspective: 10;} /* крайняя перспектива' */
```

Примечание:

На момент написания данной книги это свойство поддерживалось в префиксной форме только в WebKit.

perspective-origin

Значения:

```
[ [ <percentage> | <length> | left | center | right ]
[ <percentage> | <length> | top | center | bottom ]? ] |
[ [ left | center | right ] || [ top | center | bottom ] ]
```

Исходное значение:

50% 50%

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Задание в процентах:

По размеру блока элемента.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет исходную точку видимой трехмерной перспективы в элементе. По существу, данное свойство определяет в элементе точку, которая появляется непосредственно перед наблюдателем.

Примеры:

```
body {perspective-origin: bottom right;}
#wrapper div {perspective-origin: 0 50%;}
```

Примечание:

На момент написания данной книги это свойство поддерживалось в префиксной форме только в WebKit.

position

Значения:

static | **relative** | **absolute** | **fixed**

Исходное значение:

static

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет схему расположения, применяемую при разметке элемента. Расположить можно любой элемент, хотя элемент, размещаемый при значении **absolute** или **fixed** данного свойства, сформирует структурный блок независимо от разновидности этого элемента. Элемент с относительным расположением смещается относительно своего исходного расположения в обычном потоке.

Примеры:

```
#footer {position: fixed; bottom: 0;}
*.offset {position: relative; top: 0.5em;}
```

Примечание:

Абсолютное расположение поддерживается в браузере Internet Explorer только начиная с версии 7.

quotes

Значения:

[<string> <string>]+ | **none**

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам в CSS2, а в CSS3 — ко всем элементам, псевдоэлементам `::before`, `::after`, `::alternate`, `::marker`, `::line-marker`, областям отступов и нижних колонтитулов в блоках `@footnote`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет образец цитирования, применяемый для обычных и вложенных кавычек. Открывающие и закрывающие кавычки вставляются по значениям **openquote** и **close-quote** свойства content соответственно. Следует, однако, иметь в виду, что некоторые псевдоэлементы, к которым применяются кавычки, в том числе ::alternate, ::marker и ::line-marker, являются относительно новыми в CSS3 и могут поддерживаться не во всех браузерах.

Пример:

```
q {quotes: '\201C' '\201D' '\2018' '\2019';}
```

resize

Значения:

none | **both** | **horizontal** | **vertical**

Исходное значение:

none

Применяется:

Ко всем элементам, значение свойства overflow которых не равно visible.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок (или возможность) изменения размеров элемента пользователем. Конкретный внешний вид и действие механизма изменения размеров элемента оставляется на

усмотрение пользовательского агента и, вероятнее всего, зависит от направления письма.

Примеры:

```
textarea {resize: vertical;}
iframe {resize: both;}
```

right

Значения:

<length> | <percentage> | **auto**

Исходное значение:

auto

Применяется:

К размещаемым элементам, т.е. к таким элементам, для которых задано значение свойства position, отличающееся от **static**.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для относительно размещаемых элементов читайте приведенное ниже примечание; для статически размещаемых элементов (т.е. таких элементов, для которых задано значение **static** свойства position) вычисляемое значение равно **auto**; для значений длины оно равно соответствующей абсолютной длине; для значений в процентах — объявляемому значению, а иначе — **auto**.

Описание:

Определяет смещение внешнего края правого отступа размещаемого элемента относительно правого края содержащего его блока.

Примеры:

```
div#footer {position: fixed; right: 0;}
*.overlapper {position: relative; right: -25px;}
```

Примечание:

Для относительно размещаемых элементов вычисляемое значение свойства **left** всегда равно значению свойства **right**.

ruby-align

Значения:

auto | **start** | **left** | **center** | **end** | **right** | **distribute-letter** | **distribute-space** | **line-edge**

Исходное значение:

auto

Применяется:

Ко всем элементам и формируемому их содержимому.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет выравнивание идеографического текста относительно содержимого идеографической основы.

Примеры:

```
ruby {ruby-align: start;}
rubytext {ruby-align: distribute-space;}
```

Примечание:

Идеографическим называется такой текст, который располагается над основным текстом, что характерно для письма восточно-азиатских языков. На момент написания данной книги это свойство поддерживалось только в браузере Internet Explorer.

ruby-overhang

Значения:

auto | **start** | **end** | **none**

Исходное значение:

none

Применяется:

К родительским элементам тех элементов, значение свойства `display` которых равно **`ruby-text`**.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, с какой стороны базового текста идеографическому тексту разрешается выступать над его идеографической основой, если он оказывается шире базового текста.

Пример:

```
rubytext {ruby-overhang: none;}
```

Примечание:

Идеографическим называется такой текст, который располагается над основным текстом, что характерно для письма восточноазиатских языков. На момент написания данной книги это свойство поддерживалось только в браузере Internet Explorer.

ruby-position

Значения:

before | **after** | **right**

Исходное значение:

before

Применяется:

К родительским элементам тех элементов, значение свойства `display` которых равно **ruby-text**.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет положение идеографического текста относительно его базового текста.

Пример:

```
rubytext {ruby-position: before;}
```

Примечание:

Идеографическим называется такой текст, который располагается над основным текстом, что характерно для письма восточноазиатских языков. На момент написания данной книги это свойство поддерживалось только в браузере Internet Explorer.

ruby-span

Значения:

attr(x) | none

Исходное значение:

none

Применяется:

К элементам со значением **ruby-text** свойства display.

Наследуется:

Нет.

Вычисляемое значение:

Число.

Описание:

Определяет количество элементов базового текста, которые может охватывать идеографический текст. Значение данного свойство должно быть числовым и вычисляться как таковое. А значения **0** и **none** равнозначны значению **1** и обозначают отсутствие такого охвата.

Пример:

```
rubytext {ruby-span: attr(span);}
```

Примечание:

Идеографическим называется такой текст, который располагается над основным текстом, что характерно для письма восточно-азиатских языков. На момент написания данной книги это свойство поддерживалось только в браузере Internet Explorer.

table-layout

Значения:

auto | fixed

Исходное значение:

auto

Применяется:

К элементам со значением **table** или **inline-table** свойства display.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли размечать табличный элемент по алгоритму автоматической или фиксированной разметки. Преимущество алгоритма автоматической разметки заключается в том, что он чаще всего применяется в веб-разработке, и за более чем десятилетнее существование браузеров разработчики к нему уже привыкли, хотя теоретически алгоритм фиксированной разметки действует быстрее и более предсказуемо.

Примеры:

```
table.data {table-display: fixed;}
table.directory {table-display: auto;}
```

text-align

Значения в CSS2:

left | **center** | **right** | **justify** | <string>

Значения в CSS2.1:

left | **center** | **right** | **justify**

Значения в CSS3:

[**start** | **end** | **left** | **center** | **right** | **justify** | **match-parent**] || <string>

Исходное значение:

Зависит от конкретного пользовательского агента, а зачастую — от направления письма (в CSS2.1); **start** (в CSS3).

Применяется:

К блочным элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет выравнивание текста по горизонтали в пределах блочного элемента путем определения точки, по которой должны выравниваться построчные блоки. Поддержка значения **justify** данного свойства дает пользовательским агентам возможность выравнивать программным способом расстояние между словами, а не буквами в содержимом текстовых строк, хотя достигаемый результат может различаться для пользовательских агентов.

Примеры:

```
p {text-align: justify;}
h4 {text-align: center;}
```

Примечание:

В спецификацию CSS2 включено значение <string>, исключенное из спецификации CSS2.1 из-за отсутствия должной поддержки, хотя оно снова возвращено в спецификацию CSS3. На момент написания данной книги ему все еще недоставало поддержки в браузерах.

text-decoration

Значения:

none | [**underline** || **overline** || **line-through** || **blink**]

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет эффекты художественного оформления текста, например подчеркивание. Подобными эффектами охватываются и порожденные элементы, у которых отсутствует собственное художественное оформление. Допускается указывать значения данного свойства в разных сочетаниях. Но всякий раз, когда два объявления свойства `text-decoration` применяются к одному и тому же элементу, объявляемые значения *не* сочетаются вместе. Рассмотрим в качестве примера следующие объявления:

```
h1 {text-decoration: overline;}
h1, h2 {text-decoration: underline;}
```

Благодаря объявленным здесь стилям содержимое элементов `h1` не будет надчеркнуто, поскольку значение **overline** полностью замещается значением **underline**. Если же необходимо надчекнуть и подчеркнуть содержимое элементов `h1`, то после объявления `h1, h2` следует ввести еще одно объявление `h1` со значениями **overline underline** свойства `text-decoration` или расширить селектор в объявлении `h1, h2`,

чтобы конкретизировать его назначение. Поддержка значения **blink** данного свойства для пользовательских агентов необязательна.

Примеры:

```
u {text-decoration: underline;}
.old {text-decoration: line-through;}
u.old {text-decoration: line-through underline;}
```

text-indent

Значения в CSS2:

```
<length> | <percentage>
```

Значения в CSS3:

```
[ <length> | <percentage> ] && [ hanging || each-line ]?
```

Исходное значение:

0

Применяется:

К блочным элементам.

Наследуется:

Да.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для значений в процентах вычисляемое значение оказывается таким же, как и объявляемое, а для значений длины оно равно соответствующей абсолютной длине.

Описание:

Определяет отступ первой строки содержимого в блочном элементе. Чаще всего служит для создания эффекта табуляции.

Разрешается указывать отрицательные значения данного свойства, которые вызывают эффекты выступа (или обратного отступа), т.е. смещения текста влево. В спецификации CSS3 значение **each-line** данного свойства вызывает отступ любой новой, а не только первой строки. А значение **hanging** обращает заданный образец отступа, что дает возможность создавать эффект выступа, не прибегая к отрицательным значениям длины.

Примеры:

```
p {text-indent: 5em;}
h2 {text-indent: -25px;}
```

text-overflow

Значения:

clip | **ellipsis**

Исходное значение:

clip

Применяется:

К блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет поведение при переполнении блока родительского элемента содержимым внутристрочного элемента в тех случаях, когда для родительского элемента не задано значение **visible** свойства `overflow`. Устанавливаемое по умолчанию значение данного свойства соответствует исторически

сложившемуся поведению, когда содержимое просто отсекается по краям блока родительского элемента. А значение **ellipsis** данного свойства означает, что содержимое должно быть усечено, но в конце элемента вводится многоточие (. . .). В языках с правосторонним письмом, например в английском, многоточие размещается в правом нижнем углу элемента.

Примеры:

```
pre {text-overflow: clip;}
article {text-overflow: ellipsis;}
```

text-shadow

Значения:

none | [<length>{**2**,**4**} <color>?,] * <length>{**2**,**4**} <color>?

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Один или более наборов цветов плюс три абсолютные длины.

Описание:

Определяет одну или более теней, отбрасываемых текстовым содержимым элемента. Тени всегда воспроизводятся позади текста в элементе, но впереди его переднего плана, границ и контура. Кроме того, тени воспроизводятся от первой сверху и до последней снизу.

Четыре значения длины теней могут быть объявлены в следующем порядке: смещение по горизонтали, смещение

по вертикали, расстояние размытия и расстояние залегания тени. Если указаны положительные значения смещения, тени отбрасываются вниз и вправо. А если указаны отрицательные значения смещения, тени отбрасываются назад и влево. При положительных значениях залегания тени расширяются, а при отрицательных значениях — сокращаются. Значения размытия теней не могут быть отрицательными.

Примеры:

```
h1 {text-shadow: 0.5em 0.33em 4px 2px gray;}
h2 {text-shadow: 0 -3px 0.5em blue;}
```

text-transform

Значения:

uppercase | **lowercase** | **capitalize** | **none**

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет образец для изменения регистра букв в элементе независимо от регистра букв текста в исходном документе. Какие именно буквы следует сделать заглавными по значению **capitalize** данного свойства, явно не определено, поскольку это зависит от способности пользовательских агентов распознавать отдельные слова в тексте.

Примеры:

```
h1 {text-transform: uppercase;}
.title {text-transform: capitalize;}
```

top

Значения:

<length> | <percentage> | **auto**

Исходное значение:

auto

Применяется:

К размещаемым элементам, т.е. к таким элементам, для которых задано значение свойства position, отличающееся от **static**.

Наследуется:

Нет.

Задание в процентах:

По высоте содержащего блока.

Вычисляемое значение:

Для относительно размещаемых элементов читайте приведенное ниже примечание; для статически размещаемых элементов (т.е. таких элементов, для которых задано значение **static** свойства position) вычисляемое значение равно **auto**; для значений длины оно равно соответствующей абсолютной длине; для значений в процентах — объявляемому значению, а иначе — **auto**.

Описание:

Определяет смещение внешнего края верхнего отступа размещаемого элемента относительно верхнего края содержащего его блока.

Примечание:

Если для относительно размещаемых элементов задано значение **auto** обоих свойств, bottom и top, то их вычисляемые значения равны нулю (0). Если же только одно из этих свойств принимает значение **auto**, то его вычисляемое значение становится отрицательным значением другого свойства. А если ни одно из этих свойств не принимает значение **auto**, то вычисляемое значение свойства bottom становится отрицательным значением свойства top.

Примеры:

```
#masthead {position: fixed; top: 0;}
sub {position: relative; top: 0.5em;
vertical-align: baseline;}
```

transform

Значения:

< **none** | <transform-function> [<transform-function>]*

Расширения:

<transform-function>

См. приведенное ниже описание.

Исходное значение:

none

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет одно или более преобразований элемента. Такие преобразования могут происходить как в двухмерном, так и в имитируемом трехмерном пространстве в зависимости от того, каким образом они объявлены.

Допустимые значения для расширения ***<transform-function>*** слишком сложны и пространны. А полный их перечень и краткое описание можно найти по адресу `http://w3.org/TR/css3-3d-transforms/#transform-functions`.

Примеры:

```
table th {transform: rotate(45deg);}
li {transform: scale3d(1.2,1.7,0.85);}
```

transform-origin

Значения:

```
[ [ [ <percentage> | <length> | left | center | right ]
[ <percentage> | <length> | top | center | bottom ]? ]
<length>? ] | [ [ [ left | center | right ] ||
[ top | center | bottom ] ] <length>? ]
```

Исходное значение:

50% 50% 0

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Задание в процентах:

По размеру блока элемента.

Вычисляемое значение:

Абсолютная длина для значения `<length>`, а иначе — в процентах.

Описание:

Определяет исходную точку для преобразований элемента как в двухмерном, так и в имитируемом трехмерном пространстве. Значения <length>, отмеченные как необязательные, определяют исходную точку преобразований в имитируемом трехмерном пространстве. Если они не указаны, то данная точка относится к двухмерному пространству.

Примеры:

```
table th {transform-origin: bottom left;}
li {transform-origin: 10% 10px 10em;}
```

Примечание:

На момент написания данной книги существовали два разных рабочих варианта для преобразований в двух- и трехмерном пространстве, и каждый из них предлагал свой синтаксис для определения свойства `transform-origin`. Здесь же предпринята попытка согласовать оба варианта, не прибегая к двум разным, но почти одинаковым синтаксисам определения данного свойства.

transform-style

Значения:

[flat | preserve-3d

Исходное значение:

flat

Применяется:

К блочным и внутристрочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, должен ли элемент, преобразуемый в имитируемом трехмерном пространстве, иметь потомков, воспроизводимых в той же самой двухмерной плоскости, что и родительский элемент, или же следует попытаться создать трехмерный эффект, когда потомки с положительным или отрицательным значением свойства `z-index` воспроизводятся впереди или позади плоскости своего родительского элемента при его вращении. Элементы со значением **`hidden`** свойства `overflow` не могут сохранять трехмерные эффекты и поэтому интерпретируются так, как будто значение их свойства `transform-style` равно **`flat`**.

Пример:

```
li {transform-style: preserve-3d;}
```

transition

Значения:

```
[<'transition-property'> || <'transition-duration'> ||
<'transitiontiming-function'> || <'transition-delay'>
[, [<'transition-property'> || <'transition-duration'> ||
<'transition-timing-function'> || <'transition-delay'>]]*
```

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам, а также к псевдоэлементам `::before` и `::after`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Это сокращенная форма свойства, определяющая особенности переходов одного или более элементов из одного состояния в другое. И хотя на момент написания данной книги синтаксис объявления значений этого свойства еще не был окончательно определен, в спецификации поясняется, что если объявлены два значения `<time>`, то первое из них обозначает продолжительность перехода, а второе — задержку. Если же объявлено лишь одно значение `<time>`, то оно обозначает только продолжительность перехода.

Примеры:

```
a[href]:hover {transition: color 1s 0.25s ease-in-out;}
h1 {transition: linear all 10s;}
```

transition-delay

Значения:

`<time> [, <time>]*`

Исходное значение:

0

Применяется:

Ко всем элементам, а также к псевдоэлементам `::before` и `::after`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет задержку от теоретически возможного начала перехода и до его фактического начала. Так, если начало

перехода определяется при наведении курсора на элемент, но с задержкой на **0,5** с, то сам переход фактически начнется через полсекунды после наведения курсора на элемент в первый раз. Разрешается указывать и отрицательные значения данного свойства, но они приводят лишь к тому, что переход переносится к тому моменту, которого он достиг бы, если бы переход начинался в определенное время, отодвинутое в прошлое. Иными словами, процесс отчасти начнется в течение самого перехода и дойдет до своего логического завершения.

Примеры:

```
a[href]:hover {transition-delay: 0.25;}
h1 {transition-delay: 0;}
```

transition-duration

Значения:

```
<time> [, <time>]*
```

Исходное значение:

0

Применяется:

Ко всем элементам, а также к псевдоэлементам `::before` и `::after`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет продолжительность перехода от начала и до конца. Устанавливаемое по умолчанию нулевое значение (**0**) означает, что переход происходит мгновенно и без всякой

анимации. Отрицательные значения данного свойства интерпретируются как нулевые (**0**).

Примеры:

```
a[href]:hover {transition-duration: 1s;}
h1 {transition-duration: 10s;}
```

transition-property

Значения:

none | **all** | [<IDENT>] [',' <IDENT>]*

Исходное значение:

all

Применяется:

Ко всем элементам, а также к псевдоэлементам `::before` и `::after`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет одно или более свойств, которые должны переходить из одного состояния в другое. Например, для свойства `color` это означает переход цвета переднего плана от начального оттенка к конечному оттенку этого цвета. Если объявлена сокращенная форма свойства, то действие параметров перехода этого свойства распространяется и на все остальные свойства, представленные в сокращенной форме.

Ключевое слово **all** означает, что переходу подлежат *все* свойства. А ключевое слово **none** препятствует переходу любых свойств, по существу, отменяя их переход из одного состояния в другое.

Примеры:

```
a[href]:hover {transition-property: color;}
h1 {transition-property: all;}
```

transition-timing-function

Значения:

<transition-timing> [, <transition-timing>]*

Расширения:

<transition-timing>

ease | **linear** | **ease-in** | **ease-out** | **ease-in-out** | **cubicbezier**(<number>, <number>, <number>, <number>)

Исходное значение:

ease

Применяется:

Ко всем элементам, а также к псевдоэлементам `::before` и `::after`.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок вычисления промежуточных состояний при переходе. Ключевые слова (**ease**, **linear** т.д.), указываемые в качестве значений данного свойства, служат сокращенной формой обозначения конкретных значений функции `cubic-bezier()`, определяемых в спецификации. Следовательно, все значения данного свойства, по существу, являются значениями функции `cubic-bezier()`.

Примеры:

```
a[href]:hover {transition-timing-function: ease-in-out;}
h1 {transition-timing-function: linear;}
```

unicode-bidi

Значения:

normal | **embed** | **bidi-override**

Исходное значение:

normal

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Позволяет формировать уровни вложения в алгоритме двунаправленного форматирования символов в Уникоде. Пользовательским агентам, не поддерживающим двунаправленный текст, разрешается игнорировать данное свойство.

Пример:

```
span.name {direction: rtl; unicode-bidi: embed;}
```

vertical-align

Значения:

baseline | **sub** | **super** | **top** | **text-top** | **middle** | **bottom** | **textbottom** | <percentage> | <length>

Исходное значение:

baseline

Применяется:

К внутристрочным элементам и ячейкам таблицы.

Наследуется:

Нет.

Задание в процентах:

По значению <line-height> элемента.

Вычисляемое значение:

Для значений в процентах и значений длины вычисляемое значение равно соответствующей абсолютной длине, а иначе оно оказывается таким же, как и объявляемое значение.

Описание:

Определяет вертикальное выравнивание базовой линии внутристрочного элемента относительно базовой линии той строки, в которой находится этот элемент. Разрешается указывать отрицательные значения длины и значения в процентах, причем они опускают элемент, а не поднимают его.

В ячейках таблицы данное свойство задает выравнивание содержимого ячейки в пределах ее блока. Если данное свойство применяется к ячейкам таблицы, то допускаются только следующие его значения: **baseline**, **top**, **middle** и **bottom**.

Примеры:

```
sup {vertical-align: super;}
.fnote {vertical-align: 50%;}
```

visibility

Значения:

visible | **hidden** | **collapse**

Исходное значение:

visible

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли воспроизводить блок элемента, сформированный самим элементом. Это означает, что элемент может занять то место, которое он обычно занимает, а все остальное останется совершенно невидимым. Значение **collapse** данного свойства применяется в таблицах для удаления столбцов и строк из разметки таблицы.

Примеры:

```
ul.submenu {visibility: hidden;}
tr.hide {visibility: collapse;}
```

white-space

Значения:

normal | nowrap | pre | pre-wrap | pre-line

Исходное значение:

normal

Применяется:

Ко всем элементам (в CSS2.1); к блочным элементам (в CSS2).

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок обработки пробелов в элементе при его разметке. Значение **normal** данного свойства означает традиционную для браузеров обработку текста, при которой любая последовательность пробелов сводится к одному пробелу. Префикс **pre** в некоторых значениях данного свойства означает, что пробелы будут обрабатываться таким же образом, как и в элементе **pre** разметки HTML-документов, когда пробелы и разрывы строк полностью сохраняются. Значение **nowrap** данного свойств препятствует разбиению элемента на строки, как и аналогичный атрибут элементов разметки `td` и `th` в HTML4. А значения **pre-wrap** и **pre-line** были внедрены в спецификации CSS2.1. Первое из них обозначает сохранение пробелов пользовательским агентом наряду с автоматическим переносом текстовых строк, а второе — учет знаков новой строки в тесте наряду со сведением всех лишних пробелов в один, как при значении **normal**.

Примеры:

```
td {white-space: nowrap;}
tt {white-space: pre;}
```

width

Значения:

<length> | <percentage> | **auto**

Исходное значение:

auto

Применяется:

К блочным и заменяемым элементам.

Наследуется:

Нет.

Задание в процентах:

По ширине содержащего блока.

Вычисляемое значение:

Для исходных значений **auto** и в процентах такое же, как объявляемое значение, а иначе — абсолютная длина, если только данное свойство не применяется к элементу, и тогда оно равно **auto**.

Описание:

Определяет ширину области содержимого элемента, к которой снаружи добавляются внутренние поля, границы и отступы. Данное свойство игнорируется при разметке внутристрочных незаменяемых элементов. Указывать отрицательные значения абсолютной длины и в процентах не разрешается.

Примеры:

```
table {width: 80%;}
#sidebar {width: 20%;}
.figure img {width: 200px;}
```

word-spacing

Значения:

```
[ normal | <length> | <percentage> ]{1,3}
```

Исходное значение:

normal

Применяется:

Ко всем элементам.

Наследуется:

Да.

Задание в процентах:

По ширине глифа пробела в Уникоде (**U+0020**) для начертания шрифта элемента.

Вычисляемое значение:

Для значений длины вычисляемое значение равно соответствующей абсолютной длине, а иначе — **normal**.

Описание:

Определяет величину пробела, вставляемого между словами. Следует, однако, иметь в виду, что в спецификации понятие *слово* не определено. Поэтому на практике пользовательские агенты будут, как правило, применять данное свойство к сворачиваемым пробелам между строками непробельных символов. Разрешается указывать отрицательные значения абсолютной длины и в процентах. В таком случае слова располагаются ближе одно к другому.

Примеры:

```
p.spacious {letter-spacing: 6px;}
em {letter-spacing: 0.2em;}
p.cramped {letter-spacing: -0.5em;}
```

Примечание:

В спецификации CSS2.1 свойство `word-spacing` принимает единственное значение: `<length>` или **normal**.

word-wrap

Значения:

normal | **break-word**

Исходное значение:

normal

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок автоматического переноса текста в тех случаях, когда он не переносится обычным образом. Примером тому служит перенос очень длинной строки чисел без пробелов подобно первой тысяче цифр после десятичной точки в числе π. Значение **`break-word`** данного свойства разрешает пользовательским агентам разбивать слово на произвольные части, если им не удается найти обычные точки разбиения в том, что они распознают как *слово* (т.е. текстовую строку).

Примеры:

```
td {word-wrap: break-word;}
p {word-wrap: normal;}
```

z-index

Значения:

`<integer>` | **`auto`**

Исходное значение:

`auto`

Применяется:

К размещаемым элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет расположение размещаемого элемента по оси z, которая проводится перпендикулярно области отображения. При положительных значениях данного свойства элемент размещается ближе к наблюдателю, а при отрицательных значениях — дальше от него.

Пример:

```
#masthead {position: relative; z-index: 10000;}
```

Свойства печатных носителей информации

Ниже перечислены все свойства печатных носителей информации с кратким пояснением их назначения.

break-after

Значения:

auto | **always** | **avoid** | **left** | **right** | **page** | **column** | **avoid-page** | **avoid-column**

Исходное значение:

auto

Применяется:

К блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли размещать разрыв столбца или страницы после элемента. И хотя при значении **always** дан-

ного свойства теоретически возможно принудительное размещение разрывов, гарантировать запрет их появления нельзя. Самое лучшее, что можно сделать, — указать значение **avoid**, чтобы пользовательский агент не вставлял разрыв столбца или страницы после элемента. Ключевые слова **avoid-column** и **avoid-page** препятствуют вставке разрыва столбца или страницы соответственно после элемента. Ключевое слово **left** служит для вставки достаточного количества разрывов после элемента, чтобы следующая страница располагалась слева. Аналогично ключевое слово **right** служит для размещения следующей страницы справа. А ключевые слова **page** и **always** служат для вставки разрыва страницы после элемента, тогда как ключевые слова **column** и **always** — для вставки разрыва столбца.

Примеры:

```
h3 {break-after: avoid;}
div.col {break-after: column;}
```

break-before

Значения:

auto | **always** | **avoid** | **left** | **right** | **page** | **column** | **avoid-page** | **avoid-column**

Исходное значение:

auto

Применяется:

К блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли размещать разрыв столбца или страницы перед элементом. И хотя при значении **always** данного свойства теоретически возможно принудительное размещение разрывов, гарантировать запрет их появления нельзя. Самое лучшее, что можно сделать, — указать значение **avoid**, чтобы пользовательский агент не вставлял разрыв столбца или страницы перед элементом. Ключевые слова **avoid-column** и **avoid-page** препятствуют вставке разрыва столбца или страницы соответственно перед элементом. Ключевое слово **left** служит для вставки достаточного количества разрывов перед элементом, чтобы страница располагалась слева. Аналогично ключевое слово **right** служит для размещения страницы справа. А ключевые слова **page** и **always** служат для вставки разрыва страницы перед элементом, тогда как ключевые слова **column** и **always** — для вставки разрыва столбца.

Примеры:

```
h2 {break-before: always;}
h3 {break-before: avoid;}
```

break-inside

Значения:

auto | **avoid** | **avoid-page** | **avoid-column**

Исходное значение:

auto

Применяется:

К блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли исключить появление разрыва столбца или страницы в самом элементе. Следует, однако, иметь в виду, что избежать появления таких разрывов нельзя. Так, если сделать объявление

```
body {break-inside: avoid-page;}
```

в длинном документе не удастся избежать вставки разрывов страниц пользовательским агентом.

Примеры:

```
table {break-inside: avoid;}
ul {break-inside: avoid-column;}
```

image-orientation

Значения:

auto | <angle>

Исходное значение:

auto

Применяется:

К элементам изображений.

Наследуется:

Нет данных.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет угол поворота изображений по часовой стрелке, когда они воспроизводятся на печатных носителях информации. Это дает возможность поворачивать изображения, поступающие из мобильных устройств, где они не были

повернуты после съемки "боком". Пользовательские агенты должны поддерживать следующие вычисляемые в градусах значения углов поворота: **0deg**, **90deg**, **180deg** и **270deg**, а остальные значения могут игнорироваться. Следует, однако, иметь в виду, что данное свойство *не* требуется для вращения изображений при переходе от книжной ориентации к альбомной и обратно, поскольку такое вращение должно выполняться пользовательским агентом автоматически.

Пример:

```
img.oldphone {image-orientation: 90deg;}
```

marks

Значения:

[**crop** || **cross**] | **none**

Исходное значение:

none

Применяется:

К содержимому страницы.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли добавить крестообразные приводочные метки или метки обрезки при воспроизведении страницы.

Пример:

```
@page {marks: cross crop;}
```

orphans

Значения:

<integer>

Исходное значение:

2

Применяется:

К блочным элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет минимальное количество текстовых строк в элементе, которые можно оставить внизу страницы. Это может оказать влияние на расположение разрывов страниц в элементе.

Примеры:

```
p {orphans: 4;}
ul {orphans: 2;}
```

page

Значения:

auto | <identifier>

Исходное значение:

auto

Применяется:

К блочным элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет тип страницы, которая *должна* быть использована при воспроизведении элемента. Слово *должна* намеренно выделено не только здесь, но и в спецификации, чтобы обратить особое внимание на данное обстоятельство.

Так, если значение данного свойства для текущего элемента отличается от его значения для предыдущего элемента, то перед этим элементом вставляется хотя бы один разрыв страницы, а новая страница начинается с типа страницы, объявленного в свойстве `page`. (Если же в других стилях требуется разделять страницы на левые и правые, когда начинается новая страница, то может быть вставлено несколько разрывов страниц.)

Примеры:

```
@page wide {size: landscape;}
table.summary {page: wide;}
```

page-break-after

Значения:

auto | **always** | **avoid** | **left** | **right**

Исходное значение:

auto

Применяется:

К блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли вставить один или более разрывов страниц после элемента. И хотя при значении **always** данного свойства теоретически возможно принудительное размещение разрывов, гарантировать запрет их появления нельзя. Ключевое слово **avoid** служит для того, чтобы пользовательский агент не вставлял, если это возможно, разрыв страницы после элемента. Ключевое слово **left** служит для вставки достаточного количества разрывов после элемента, чтобы страница располагалась слева. Аналогично ключевое слово **right** служит для размещения страницы справа. Свойство `page-break-after`, по существу, заменяется свойством `break-after`, хотя поддержка первого в браузерах может оказаться более прочной, чем поддержка последнего.

Примеры:

```
section {page-break-after: always;}
h1 {page-break-after: avoid;}
```

page-break-before

Значения:

auto | **always** | **avoid** | **left** | **right**

Исходное значение:

auto

Применяется:

К блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли вставить один или более разрывов страниц перед элементом. И хотя при значении **always** данного свойства теоретически возможно принудительное размещение разрывов, гарантировать запрет их появления нельзя. Ключевое слово **avoid** служит для того, чтобы пользовательский агент не вставлял, если это возможно, разрыв страницы перед элементом. Ключевое слово **left** служит для вставки достаточного количества разрывов перед элементом, чтобы страница располагалась слева. Аналогично ключевое слово **right** служит для размещения страницы справа. Свойство `page-break-before`, по существу, заменяется свойством `break-before`, хотя поддержка первого в браузерах может оказаться более прочной, чем поддержка последнего.

Примеры:

```
section {page-break-before: always;}
h2 {page-break-before: avoid;}
```

page-break-inside

Значения:

auto | **avoid**

Исходное значение:

auto

Применяется:

К блочным элементам.

Наследуется:

Нет.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет, следует ли исключить появление разрыва страницы в самом элементе. Необходимо, однако, иметь в виду, что избежать появления таких разрывов нельзя. Так, если сделать объявление

```
{page-break-inside: avoid;}
```

в длинном документе не удастся избежать вставки разрывов страниц пользовательским агентом. Свойство `page-break-inside`, по существу, заменяется свойством `break-inside`, хотя поддержка первого в браузерах может оказаться более прочной, чем поддержка последнего.

Пример:

```
table {page-break-inside: avoid;}
```

page-policy

Значения:

`start | first | last`

Исходное значение:

`start`

Применяется:

К блокам `@counter` и `@string`.

Наследуется:

Нет данных.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет порядок указания значений счетчиков или строковых значений относительно страничного элемента. Например, сначала может возникнуть потребность определить счетчик CSS для подсчета количества разделов, а затем включить в заголовок каждой страницы номер первого обнаруживаемого на ней раздела. Это можно сделать в объявлении

```
@counter secnum {page-policy: first;}
```

а также объявить таблицу стилей CSS, которая необходима для создания образца счетчика. Если же потребуется последний экземпляр счетчика на странице, то в приведенном выше объявлении следует указать `page-policy: last`.

Значение **`start`** данного свойства означает, что используется прежнее значение, установленное до обработки текущей страницы. Так, если вернуться к предыдущему примеру, то будет использован номер счетчика, взятый из предыдущей страницы, а не первый экземпляр счетчика на текущей странице.

Примеры:

```
@counter chapter {page-policy: first;}
@string section-title {page-policy: start;}
```

size

Значения:

auto | <length>{1,2} | [<page-size> || [**portrait** | **landscape**]]

Расширения:

<page-size>

A5 | **A4** | **A3** | **B5** | **B4** | **letter** | **legal** | **ledger**

Исходное значение:

auto

Применяется:

В контексте страницы.

Наследуется:

Нет данных.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет размеры и ориентацию страничного блока. Ключевые слова **auto**, **portrait** и **landscape** приводят к тому, что страничный блок заполняет пространство, доступное для воспроизведения на странице. Содержимое страничных блоков со значением **portrait** данного свойства печатается таким образом, чтобы более длинные стороны страничного блока располагались слева и справа. А содержимое страничных блоков со значением **landscape** данного свойства печатается таким образом, чтобы более длинные стороны страничного блока располагались вверху и внизу.

Если размеры страничного блока указываются с помощью значений длины или какого-нибудь ключевого слова из расширения <***page-size***>, обозначающего формат страницы (например, **A4**), а сам страничный блок не вписывается в размеры страницы для его воспроизведения, то размеры страничного блока и его содержимого могут быть уменьшены, чтобы подогнать его по странице. Если же указано лишь одно значение длины, то оно означает оба размера, тем самым определяя квадратный страничный блок. Значения длины, указанные в единицах измерения **em** или **ex**, рассчитываются относительно вычисляемого размера шрифта, выбираемого из контекста страницы.

Пример:

```
body {page-size: landscape;}
```

widows

Значения:

<integer>

Исходное значение:

2

Применяется:

К блочным элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет минимальное количество текстовых строк в элементе, которые можно оставить вверху страницы. Это может оказать влияние на расположение разрывов страниц в элементе.

Примеры:

```
p {widows: 4;}
ul {widows: 2;}
```

Свойства акустических носителей информации

Ниже перечислены все свойства акустических носителей информации с кратким пояснением их назначения.

cue

Значения:

<'cue-before'> || <'cue-after'>

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

Применяется к наследуемому значению свойства `voice-volume`.

Вычисляемое значение:

См. ниже описание отдельных свойств (`cue-before` и пр.).

Описание:

Это сокращенная форма свойства, определяющая звуковые сигналы, которые следуют до и после озвучивания содержимого элемента. Звуковой сигнал можно рассматривать как акустический аналог пиктограммы.

Примеры:

```
table.layout {
    cue: url(shattered-glass.ogg) url(sad-trombone.wav);}
pre {cue: url(raygun.mp3);}
```

cue-after

Значения:

none | <uri> [<number> | <percentage> | **silent** | **x-soft** | **soft** | **medium** | **loud** | **x-loud**]

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

Применяется к наследуемому значению свойства voice-volume.

Вычисляемое значение:

Абсолютный URI для значений <uri>, а иначе — **none**.

Описание:

Определяет звуковой сигнал, который следует после озвучивания содержимого элемента.

Примеры:

```
table.layout {cue-after: url(sad-trombone.wav);}
pre {cue-after: url(raygun.mp3);}
```

cue-before

Значения:

none | <uri> [<number> | <percentage> | **silent** | **x-soft** | **soft** | **medium** | **loud** | **x-loud**]

Исходное значение:

none

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Задание в процентах:

Применяется к наследуемому значению свойства voice-volume.

Вычисляемое значение:

Абсолютный URI для значений <uri>, а иначе — **none**.

Описание:

Определяет звуковой сигнал, который предшествует озвучиванию содержимого элемента.

Примеры:

```
table.layout {cue-before: url(shattered-glass.ogg);}
pre {cue-before: url(raygun.mp3);}
```

pause

Значения:

```
<'pause-before'> || <'pause-after'>
```

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

См. ниже описание отдельных свойств.

Описание:

Это сокращенная форма свойства, определяющая паузы, которые следуют до и после озвучивания содержимого элемента. *Пауза* — это промежуток, в течение которого содержимое вообще не озвучивается, хотя фоновые звуки могут быть все еще слышны. Подробнее о размещении пауз читайте ниже, в описании свойств pause-before и pause-after.

Примеры:

```
h1 {pause: 1s 500ms;}
ul {pause: 250ms;}
```

pause-after

Значения:

none | **x-weak** | **weak** | **medium** | **strong** | **x-strong** | <time>

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютное значение времени.

Описание:

Определяет продолжительность паузы, которая следует после озвучивания содержимого элемента. Пауза — это промежуток, в течение которого содержимое вообще не озвучивается, хотя фоновые звуки могут быть все еще слышны. Эта пауза воспроизводится после любого звукового сигнала, который следует после озвучиваемого элемента. (См. также описание свойства `cue-after` и других соответствующих свойств.)

Примеры:

```
h1 {pause-after: 500ms;}
ul {pause-after: 250ms;}
```

pause-before

Значения:

none | **x-weak** | **weak** | **medium** | **strong** | **x-strong** | <time>

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Абсолютное значение времени.

Описание:

Определяет продолжительность паузы, которая предшествует озвучиванию содержимого элемента. Пауза — это промежуток, в течение которого содержимое вообще не озвучивается, хотя фоновые звуки могут быть все еще слышны. Эта пауза воспроизводится до любого звукового сигнала, который предшествует озвучиваемому элементу. (См. также описание свойства `cue-before` и других соответствующих свойств.)

Примеры:

```
h1 {pause-before: 1s;}
ul {pause-before: 250ms;}
```

phonemes

Значения:

<string>

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Не определено, но, вероятнее всего, такое же, как и объявляемое значение.

Описание:

Определяет фонетическое произношение содержимого элемента. При указании значения `<string>` используется Международный фонетический алфавит, обозначаемый экранируемыми кодовыми точками в Уникоде.

Пример:

```
#tomato {phonemes: "t\0252 m\0251 to\028a ";}
```

rest

Значения:

```
<'rest-before'> || <'rest-after'>
```

Исходное значение:

Не определено для сокращенных форм свойств.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Не определено, но, вероятнее всего, соответствует паре абсолютных значений времени.

Описание:

Это сокращенная форма свойства, определяющая перерывы, следующие до и после озвучивания содержимого элемента. *Перерыв* — это промежуток, в течение которого содержимое вообще не озвучивается, хотя фоновые звуки могут быть все еще слышны. Подробнее о размещении перерывов читайте ниже, в описании свойств `rest-before` и `rest-after`.

Примеры:

```
th {rest: 0.5s;}
strong {rest: 333ms 250ms;}
```

rest-after

Значения:

none | **x-weak** | **weak** | **medium** | **strong** | **x-strong** | <time>

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Не определено, но, вероятнее всего, соответствует абсолютному значению времени.

Описание:

Определяет продолжительность перерыва, который следует после озвучивания содержимого элемента. Перерыв — это промежуток, в течение которого содержимое вообще не озвучивается, хотя фоновые звуки могут быть все еще слышны. Этот перерыв воспроизводится после озвучивания содержимого

элемента, но перед любым звуковым сигналом, который следует после озвучиваемого элемента. (См. также описание свойства cue-after и других соответствующих свойств.)

Примеры:

```
th {rest-after: 0.5s;}
strong {rest-after: 250ms;}
```

rest-before

Значения:

none | **x-weak** | **weak** | **medium** | **strong** | **x-strong** | <time>

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Не определено, но, вероятнее всего, соответствует абсолютному значению времени.

Описание:

Определяет продолжительность перерыва, который предшествует озвучиванию содержимого элемента. Перерыв — это промежуток, в течение которого содержимое вообще не озвучивается, хотя фоновые звуки могут быть все еще слышны. Этот перерыв воспроизводится перед озвучиванием содержимого элемента, но после любого звукового сигнала, который предшествует озвучиваемому элементу. (См. также описание свойства cue-before и других соответствующих свойств.)

Примеры:

```
th {rest-before: 0.5s;}
strong {rest-before: 333ms;}
```

speak

Значения:

normal | **spell-out** | **digits** | **literal-punctuation** | **no-punctuation**

Исходное значение:

normal

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Не определено.

Описание:

Определяет порядок озвучивания содержимого элемента. Значение **spell-out** данного свойства, как правило, служит для озвучивания таких сокращений, как, например, W3C или CSS. Указание значения **digits** данного свойства означает, что числа озвучиваются отдельными цифрами. Например, число **13** произносится как "one three" (один три). А указание значения **literal-punctuation** данного свойства приводит к тому, что знаки препинания произносятся буквально так, как слова "period" (точка) и "semicolon" (точка с запятой). И наконец значение **no-punctuation** означает, что озвучивание знаков препинания пропускается, а на их месте никаких пауз не воспроизводится.

Примеры:

```
abbr {speak: spell-out;}
*.tel {speak: digits;}
```

speakability

Значения:

auto | **none** | **normal**

Исходное значение:

auto

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Не определено.

Описание:

Определяет, будет ли озвучено содержимое элемента. Если указано значение **normal** данного свойства, элемент озвучивается независимо от значения свойства display. Если же указано значение **none** данного свойства, то элемент и все сопутствующие ему звуковые сигналы, паузы и перерывы пропускаются, т.е. для их озвучивания не выделяется времени. Но это значение может быть переопределено в порожденных элементах, что приведет к их озвучиванию. И наконец значение **auto** данного свойства приводится к значению **none**, если задано значение **none** свойства display. В противном случае оно приводится к значению **normal**.

Примеры:

```
abbr {speak: spell-out;}
*.tel {speak: digits;}
```

voice-balance

Значения:

<number> | **left** | **center** | **right** | **leftwards** | **rightwards**

Исходное значение:

center

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Не определено, но, вероятнее всего, соответствует абсолютному числовому значению.

Описание:

Определяет стереобалансировку озвучивающего голоса. Это дает возможность перемещать звучание голоса из одного стереоканала в другой, а при заданном числовом значении <number> — смешивать звучание голоса в обоих стереоканалах. Например, при значении **-50** голос звучит так, как будто он исходит из положения слева от центра. Числовые значения <number> указываются в пределах от **-100** до **100** включительно. Ключевое слово **left** равнозначно числовому значению **-100**, а ключевое слово **right** — числовому значению **100**. Если указано ключевое слово **leftwards**, то из наследуемого значения свойства voice-balance вычитается числовое значение **20**. А если указано ключевое слово **rightwards**, то к наследуемому значению свойства voice-balance прибавляется числовое значение **20**. Данное свойство применяется к звуковым сигналам (см. также описание свойства cue и других соответствующих свойств.)

Примеры:

```
.beck {voice-balance: right;}
.moore {voice-balance: left;}
```

voice-duration

Значения:

`<time>`

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Нет.

Вычисляемое значение:

Не определено.

Описание:

Определяет продолжительность времени, необходимого для озвучивания содержимого элемента. В конечном итоге замещает значение свойства `voice-rate`. Допускаются только положительные значения `<time>` данного свойства.

Примеры:

```
.tel {voice-duration: 3s;}
big {voice-duration: 10s;}
```

voice-family

Значения:

`<voice> [, <voice> ]*`

Расширения:

<voice>

<specific-voice> | [<age>? <generic-voice> <non-negative-number>?]

<age>

child | young | old

<generic-voice>

male | female | neutral

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Такое же, как и объявляемое значение.

Описание:

Определяет одно или более семейств голосов, которые можно использовать для озвучивания содержимого элемента. Сравнимо со свойством `font-family` в том отношении, что может быть использовано для предоставления списка семейств голосов, включая типичные альтернативы.

Примеры:

```
body {voice-family:
      "Karla", "Jenny", young female, female, neutral;}
small {voice-family: male child, child;}
```

voice-pitch

Значения:

<number> | <percentage> | **x-low** | **low** | **medium** | **high** | **x-high**

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Да.

Задание в процентах:

По наследуемому значению.

Вычисляемое значение:

Не определено, но, вероятнее всего, соответствует абсолютному числовому значению.

Описание:

Определяет среднюю частоту основного тона голоса, озвучивающего содержимое элемента. Средняя частота основного тона озвучивающего голоса будет зависеть, главным образом, от выбранного семейства голосов. Значения `<number>` данного свойства указываются в герцах, определяя частоту основного тона.

Примеры:

```
big {voice-pitch: 100;}
small {voice-pitch: high;}
```

voice-pitch-range

Значения:

`<number>` | `<percentage>` | **x-low** | **low** | **medium** | **high** | **x-high**

Исходное значение:

Зависит от конкретного пользовательского агента.

Применяется:

Ко всем элементам.

Наследуется:

Да.

Задание в процентах:

По наследуемому значению.

Вычисляемое значение:

Не определено, но, вероятнее всего, соответствует абсолютному числовому значению.

Описание:

Определяет отклонение средней частоты основного тона голоса, озвучивающего содержимое элемента. Чем больше это отклонение, тем более оживленно звучит голос. Значения `<number>` данного свойства указываются в герцах, определяя диапазон частот основного тона.

Примеры:

```
em {voice-pitch-range: high;}
code {voice-pitch-range: 50;}
```

voice-rate

Значения:

`<percentage>` | **x-low** | **low** | **medium** | **fast** | **x-fast**

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Да.

Задание в процентах:

По исходному значению, устанавливаемому по умолчанию.

Вычисляемое значение:

Не определено, но в предыдущем воплощении в виде свойства `speech-rate` соответствовало абсолютному числовому значению.

Описание:

Определяет среднюю частоту произношения слов при озвучивании содержимого элемента.

Примеры:

```
h1 {voice-rate: 33%;}
.legalese {voice-rate: x-fast;}
```

voice-stress

Значения:

strong | **moderate** | **reduced** | **none**

Исходное значение:

moderate

Применяется:

Ко всем элементам.

Наследуется:

Да.

Вычисляемое значение:

Не определено, но, вероятнее всего, соответствует объявляемому значению.

Описание:

Оказывает влияние на высоту тона в интонации озвучивающего голоса, которая обычно повышается по знакам ударения, принятым в конкретном языке.

Примеры:

```
strong {voice-stress: strong;}
footer {voice-stress: reduced;}
```

voice-volume

Значения:

<number> | <percentage> | **silent** | **x-soft** | **soft** | **medium** | **loud** | **x-loud**

Исходное значение:

medium

Применяется:

Ко всем элементам.

Наследуется:

Да.

Задание в процентах:

По наследуемому значению.

Вычисляемое значение:

Не определено, но в предыдущем воплощении в виде свойства volume соответствовало абсолютному числовому значению.

Описание:

Определяет средний уровень громкости для формы звуковых колебаний при озвучивании содержимого элемента. Следовательно, форма звуковых колебаний с крупными всплесками и впадинами может значительно подниматься выше или опускаться ниже среднего уровня громкости, устанавливаемого с помощью данного свойства. Значения <number> данного свойства указываются в пределах от **0** до **100** включительно. Следует также иметь в виду, что нулевое значение (**0**)

равнозначно ключевому слову **silent**, а значение **100** — ключевому слову **x-loud**.

Примеры:

```
big {voice-volume: x-loud;}
footer {voice-volume: 15;}
```

Предметный указатель

П

Р

С

Т

Э

Я

O'REILLY®

CSS. Карманный справочник

Работая с каскадными таблицами стилей (CSS), читатель может быстро найти нужный ответ в этом удобном кратком справочнике, в котором предоставляются все основные сведения, необходимые для оперативной реализации CSS. Этот карманный справочник идеально подходит для веб-разработчиков от промежуточного до продвинутого уровня квалификации, а его четвертое издание дополнено согласно спецификации CSS3. Помимо полного перечня в алфавитном порядке селекторов и свойств CSS3, вы найдете в этом издании краткое введение в основные понятия CSS.

В основу этого справочника положен материал книги *CSS. Каскадные таблицы стилей. Подробное руководство* того же автора. Он служит удобной памяткой по спецификациям CSS при решении текущих задач веб-разработки.

Эта книга поможет вам:

- быстро найти и адаптировать нужные элементы стилевого оформления;
- узнать, каким образом средства CSS3 дополняют и расширяют ваши практические навыки применения CSS;
- обнаружить новые типы значений и новые CSS-селекторы;
- реализовать падающие тени, несколько задних планов, скругленные углы и изображения границ элементов разметки веб-страниц;
- получить новые сведения о преобразованиях и переходах.

Эрик А. Мейер считается признанным во всем мире специалистом, автором и пропагандистом HTML, CSS и прочих веб-стандартов. Он является основателем компании Complex Spiral Consulting и соучредителем конференции An Event Apart, а также автором многочисленных книг по CSS и координатором работ по созданию официального тестового набора CSS Test Suite от консорциума W3C.

http://www.williamspublishing.com

www.oreilly.com

ISBN 978-5-8459-2081-2

9 785845 920812 16040