ТЕМА 3.1 ОСНОВНЫЕ ПОЛОЖЕНИЯ И ПРИНЦИПЫ ЯЗЫКА CSS2

Введение в каскадные таблицы стилей

Для начала попробуем понять, что такое каскадные таблицы стилей и для чего они нужны. Таблица стилей представляет собой написанный на языке CSS2 (или CSS1) набор правил, применяемых к HTML-элементам документа, к которому эта таблица подключена. Правила, эти задают визуальное представление содержимого HTML-элементов, или, иначе говоря, то, как они будут выглядеть на экране монитора (цвет, рамки, размер, выравнивание, шрифт, отступы, видимость и т.п.). Благодаря CSS эти параметры не надо задавать для каждого элемента через его атрибуты. Таким образом, каскадные таблицы стилей позволяют отделять содержимое HTML-документа от описания его внешнего вида. К тому же этим достигается значительное удобство, компактность описания визуальных свойств HTML-элементов и оперативность их изменения.

Рассмотрим это на примере. Допустим, что в документе присутствует несколько заголовков третьего уровня H3 и разработчику требуется вывести их зеленым цветом и курсивом, тогда как весь остальной текст должен быть черного цвета. Используя стандартные средства HTML, автор документа должен будет для каждого такого заголовка установить зеленый цвет и выделение курсивом. А если он потом задумает изменить цвет заголовка, то ему придется снова перебрать все элементы H3. Причем какой-то из них по невнимательности он может и пропустить.

Применяя каскадные таблицы стилей, описание визуальных свойств заголовков достаточно произвести только один раз. При этом указанные для них правила будут применяться ко всем элементам H3 в данном документе, благодаря чему достаточно один раз, в задании правила, поменять цвет, чтобы все заголовки в документе поменяли свою окраску.

Сравнение:

```
<HTML>
<HEAD>
<TITLE>Документ,  написанный
     с использованием CSS</TITLE>
<STYLE type="text/css">
H3 {color:green;
          font-style:italic}
<STYLE>
</HEAD>
<BODY>
<H3>Заголовок 3-го уровня</H3>
……текст документа…….
……текст документа……
<H3>Заголовок 3-го уровня</H3>
……текст документа.......
……текст документа…….
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>Документ, написанный
      без  использования  CSS</TITLE>
</HEAD>
<BODY>
<FONT color=green>
<H3><1>Заголовок 3-го уровня <1x/H3>
</FONT>
…….текст документа……..
…….текст документа……..
<PONT color=green>
<H3><1>Заголовок 3-го уровня<1x/H3>
</FONT>
……текст документа…….
……текст документа…….
</BODY>
</HTML>
```

Заметьте, что в примере с помощью возможностей CSS оптимизирован только один HTML-элемент, а как упростился документ.

Умелое использование возможностей CSS2 (обособленно или в сочетании с DHTML) позволяет реализовать оригинальные визуальные эффекты, описание которых произведено ниже.

Браузер, как и в случае с HTML-элементами, игнорирует непонятные ему правила, потому никаких конфликтных ситуаций из-за использования каскадных таблиц стилей возникать не может.

Каскадные таблицы стилей могут располагаться либо в заголовке HTML-документа (в содержимом элемента STYLE), либо во внешнем файле с расширением .css. В этом случае подключение таблицы стилей осуществляется элементом заголовка Link. Использование внешних таблиц стилей особенно актуально для многостраничных сайтов, все страницы которых должны быть выдержаны в одном дизайнерском решении. Поэтому, вместо того, чтобы в коде каждой страницы писать одинаковые стилевые настройки, достаточно написать их один раз, поместить во внешний css-файл и подключить ко всем страницам сайта.

Таблицы стилей называются каскадными, потому что при подключении к одному HTML-документу нескольких стилевых таблиц, они, в соответствии со своим приоритетом, выстраиваются в каскад. По этому каскаду и прогоняется документ. При этом правила с более высоким приоритетом переопределяют идентичные правила с более низким приоритетом. Подробнее вопрос каскадирования и наследования будет рассмотрен в разделе «Каскады и наследование». Стоит упомянуть о том, что таблицы стилей могут быть заданы и написаны под конкретное устройство, например экран или принтер. Это значит, что правила таблицы стилей, написанной для принтера будут применяться к документу только при его печати.

Способы подключения каскадных таблиц стилей к HTML- документам

Для того чтобы браузер применял правила какой-либо таблицы стилей к HTML-документу, необходимо связать таблицу стилей и документ друг с другом. Существует четыре метода, которыми это можно сделать:

- ♠ Внедрение — задание таблицы стилей непосредственно в заголовке самого HTML-документа в качестве содержимого элемента STYLE.
- ♠ Присоединение — таблица стилей находится во внешнем файле и присоединяется к HTML-документу через элемент Link. При этом CSS-файл с внешней таблицей стиля всегда сопровождает HTML-файл документа.
- ♠ Импортирование — текст таблицы стилей, находящейся во внешнем файле на сервере, импортируется с помощью Css-свойства @import внутрь текста HTML-файла.
- ♠ Поэлементное задание стиля — для всех HTML-элементов определен атрибут STYLE и через него, используя синтаксис CSS2, можно задавать (или переопределять) стиль для каждого элемента индивидуально.

Внедрение, как уже упоминалось выше, осуществляется HTML-элементом заголовка STYLE. Именно в его содержимом и задается каскадная таблица стилей. При этом атрибут type элемента STYLE должен быть установлен в значении "text/css".

Например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN ">
<HTML>
    <HEAD>
    <TITLE>Пример внедренной таблицы стиля </TITLE>
    <STYLE type ="text/css">
       ..... список CSS-правил.....
       ..... список CSS-правил.....
       ..... список CSS-правил.....
   </STYLE>
   </HEAD>
   <BODY>
       .....текст документа.....
       .....текст документа.....
```

.....текст документа.....

</BODY>

</HTML>

Присоединение внешних таблиц стилей к HTML-документу осуществляется элементом Link и имеет следующий вид:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN ">
<HTML>
    <HEAD>
        <TITLE>Пример подключения таблицы стиля </TITLE>
        <Link rel ="stylesheet" type ="text/css" href = "file_of_style.css">
    </HEAD>
    <BODY>
        ……текст документа……
        ……текст документа……
        ……текст документа……
    </BODY>
</HTML>
```

Здесь *file_of_style.css* — имя файла, в котором хранится подключаемая таблица стилей. Файл, по сути является текстовым и содержит перечень правил CSS2. Пример текста файла с внешней таблицей стиля:

```
Body   {font-family:sans-serif
    font-size:14  pt}
P   {background — color:   yellow}
```

В содержимое элемента STYLE можно импортировать таблицу стилей, хранящуюся во внешнем файле на сервере. Это осуществимо благодаря особому свойству @import каскадных таблиц стилей. При этом задание элемента STYLE будет иметь следующий вид:

```
<STYLE  type  =  "text/css  ">
        @import URL(www.servername/file_of_style)
</STYLE>
```

Значением свойства ©import является URL-адрес файла, хранящего в себе импортируемую таблицу стилей. Элемент STYLE вместе с описанием свойства ©import может содержать в себе и другие правила. Надо только, чтобы они следовали после задания свойства ©import.

Например:

```
< STYLE type = "text/css"
    @import URL(www.servername/file_of_style)
    Body {font-family:sans-serif
        font-size:14 pt}
P {background — color: yellow}
</STYLE>
```

Индивидуальное задание стиля для элемента применятся редко, и используется для переопределения уже установленных стилевых правил для данного элемента, так как оно обладает наибольшим приоритетом по сравнению с другими.

Типы данных в синтаксис CSS2

Синтаксические правила, присутствующие в CSS2

В основе всех версий CSS (CSS1, CSS2 и в будущем CSS3) лежит общий для всех набор синтаксических правил. Именно благодаря этому обстоятельству обеспечивается преемственность различных версий CSS. В том случае, если браузер поддерживает ранние версии CSS и не поддерживает новые возможности, то последние будут просто им проигнорированы. Однако правила, которые браузер в состоянии применить, он применит.

Каскадная таблица стилей, написанная на языке CSS любой версии, представляет собой список правил, которые в свою очередь подразделяются на обычные правила и ©правила (читается как Эй-ти- правило). В тексте таблицы стилей с обоих сторон правила в строке может находиться пустое пространство любых размеров.

Начинаются ©правила с ключевого символа @, непосредственно за которым следует указание имени правила. Затем ©правило включает в себя все, что находится до первого символа точки с запятой или другого правила. Следует помнить, что все правила ©import, которые находятся внутри блока определений другого ©правила, будут проигнорированы браузером.

Пример недопустимого задания:

@media print {@import URL("style_file_for_printer.css")}

К тому же, если таблица стилей содержит еще другие правила, кроме правил ©import, то эти правила должны располагаться ниже по тексту правил ©import.

Пример неправильного задания:

```
    P  {background-color:green}
@ import  URL("  file_of_style.css  ")
```

Пример правильного задания:

```
@ import  URL("  file_of_style.css  ")
    P  {background-color:green}
```

Все обычные правила каскадных таблиц стилей состоят из двух частей: селектора и блока определений и имеют следующий вид:

селектор {блок определений}.

Селектором служит название HTML-элемента, или комбинация названий HTML-элементов, для которых и задается правило форматирования. Блок определений начинается с левой фигурной скобки "{" и заканчивается правой фигурной скобкой "}". Все, что находится перед левой фигурной скобкой, считается селектором. Пример:

H1,H2 {color: blue; font family: Times New Roman}

Между фигурными скобками блока определений располагаются объявления. Они имеют следующий общий вид:

Название свойства: значение свойства

Причём между «названием», двоеточием и «значением» может находиться любое количество пробелов.

Несколько объявлений для одного селектора могут быть объединены в группы, отделяясь, друг от друга точкой или запятой.

Например:

Следующий набор правил

P{font-size:14 pt}

P{font-family: Arial}

P{font-color: yellow}

P{background-color: blue}

Небольшая тонкость заключается в отличии обработки CSS правил разными браузерами. Браузер Internet Explorer, если какое либо свойство в правиле задано неправильно, игнорирует только это свойство, а все остальные свойства этого правила учитывает. Браузер Netscape Communicator 4.74 в этой ситуации игнорирует всё правило. Netscape Communicator 6 этого недостатка лишён и ошибки обрабатывает аналогично Internet Explorer.

Объявление может не определять никакого свойства. При этом оно называется пустым. Например,

IMG {}.

Где: IMG — селектор;

{} — блок определений.

Каскадные таблицы стилей CSS2 не учитывают регистр. Это значит, что названия правил, селекторов и их значения можно указывать как прописными, так и строчными буквами. Чувствительными к регистру могут оказаться объекты, присутствующие в таблице стилей, но не являющиеся объектами языка CSS. Например, по-прежнему учитывается регистр атрибутов id и class у HTML-элементов.

Примечание. *Если таблицы стилей будут применяться к XML-файлам, то необходимо помнить о том, что имена элементов в XML (в отличие от HTML) также являются чувствительными к регистру.*

При написании текста таблиц стилей допустимо использование комментариев. Они должны начинаться символом "/*" и заканчиваться символом "*/" Например:

```
B {font-color: blue; /*комментарии */
  font-size : 14pt /* комментарии */}
```

В CSS допустимо также использование комментариев "<!--" и "-->", установленных в HTML. При корретной поддержке CSS-таблиц стилей браузером группы символов "<!-- " " -->" будут проигнорированы. При этом они никаких комментариев создавать не могут.

Использовать SGML-комментарии имеет смысл только тогда, когда требуется скрыть содержимое таблицы стилей от браузеров, работающих с HTML 3.2 и более ранних версий, где эти таблицы не поддерживаются. Иначе текст таблицы будет отображаться в окне браузера как содержимое документа. Чтобы этого избежать, как раз и рекомендуется текст css-таблицы заключать в SGML-комментарии. Например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN ">
<HTML>
<HEAD>
<TITLE>
Пример заключения таблицы стиля в SGML-комментарии
</TITLE>
<STYLE type ="text/css">
<!--
IMG {float:left}
   BODY (background-color: lightgrey;
```

```
                                font-color: black;
                                font-color: 12pt} -->

</STYLE>
</HEAD>
<BODY>
 ..... текст документа……..
 ..... текст документа.....
 ..... текст документа.....
</BODY>
</HTML>
```

Обработка синтаксических ошибок

При синтаксической проверке браузером CSS-таблицы стилей могут возникать следующие ошибки и соответствующие им действия браузера:

- Если указано имя несуществующего свойства (или сделана ошибка при написании существующего), то браузером будет проигнорировано объявление, его содержащее. Например, правило

B {font-color: blue;

*thick:15; /*thick - несуществующее свойство*/*

font-size:14 pt}

будет воспринято браузером как:

B (font-color: blue;

font-size : 14 pt}

- Если для определенного в CSS2 свойства указано недопустимое значение, то также будет проигнорировано содержащее его объявление. Например, фрагмент таблицы стилей:

P {font-color:black}

P{background-color:"lightgrey"} / некорректное задание, т.к. ключевое слово заключено в кавычки */*

P{font-size:A3x} / неверное задание размера шрифта*/*

после синтаксической проверки примет следующий вид:

P{font-color: black}

P{ }

P{}

- Если неверно задано имя ©правила, то игнорируется все, что относится к этому правилу. Например, фрагмент CSS-таблицы:

@superstyle {

P { font-color: black;

background-color: lightgrey}

H3{font-color: yellow}

}

BODY Ifont.-familv:Arial 1

Будет воспринято браузером просто как:

BODY {font-style:Arial}

так как @правило @superstyle в CSS2 не определено.

Допустимые значения величин, используемых в CSS2

В этом разделе будут описаны значения, которые могут принимать те или иные свойства, а также их размерности и допустимые интервалы. При первом ознакомлении с CSS2 этот раздел можно пропустить и возвращаться к нему по мере надобности. Но для более продуктивного ознакомления с возможностями CSS2 рекомендуется этот раздел прочесть предварительно.

Числа

Здесь ничего нового. В CSS2 могут использоваться целые и вещественные числа, представленные в десятичной форме исчисления. Допустимые интервалы значений индивидуальны для каждого свойства.

Единицы измерения длины

В CSS2 допустимо использование двух типов задание длины: относительное задание и абсолютное. Относительное задание длины подразумевает ее задание относительно чего-то. К относительным единицам относятся:

em……..lem равен используемому значению свойства font-size элемента, к которому данный em-размер применяется;

ex..... lex равен размеру высоты строчной буквых для используемого шрифта;

px..... указывает размер в пикселах и, соответственно, зависит от разрешения экрана.

Использование абсолютных единиц измерения предпочтительно только в тех случаях, когда известны геометрические размеры устройства вывода.

cm..... сантиметр;

mm..... миллиметр;

in .....дюйм (1 дюйм равен 2,54 см);

pt .....точка (point). В CSS2 lpt = 1/72 дюйма;

pc…….пика. В CSS2 1пика = 12 точкам.

Задание URL в контексте CSS2

Указание URL-адреса в CSS2 осуществляется в следующем формате:

url (" www.nam6server.com")

Например, указать адрес фонового изображения можно следующим способом:

Body {background: url (www.myserver.da.ru/myimage.gif)}

При этом допустимо использование как абсолютных, так и относительных URL-адресов.

Указание цвета

Для задания цвета отображаемого содержимого HTML-элемента могут применяться следующие возможности:

- указание цвета с помощью ключевых слов. Пример:

H3{color: blue}

- указание цвета с помощью шестнадцатеричного задания его RGB-кода. Например:

H3 {color: # FFFFFFFF}

- указание цвета с помощью десятеричного задания его RGB-кода. При этом используются целые числа в диапазоне от 0 до 255. Например:

H3 (color: rgb (0,255,0)} / зеленым цветом*/*

- указание цвета с помощью процентного задания насыщенности каждого из цветов в RGB-коде. При этом используются вещественные числа в интервале от 0% до 100%. Например:

H3 {color: rgb (0%,100%,0%) }

Числовые значения, выходящие за допустимые пределы, приводятся к ближайшему предельному значению. Например:

H3 {color: rgb (0,300,0) }

будет уменьшено до

H3 (color: rgb (0,255,0)}

H1 {color: rgb (-10%,50%,20%)} до H1 {color: rgb (0%,50%,20%)}

Углы

Углы в каскадных таблицах стилей, написанных на языке CSS2, могут быть заданы в следующих единицах измерения:

deg.....градусы;

rad .....радианы;

grad.....грады.

Значения углов могут быть отрицательными и задаваться как целыми, так и вещественными числами.

Время

Задание времени (не путать с датой) может осуществляться либо в секундах (S), либо в миллисекундах (ms).

Понятие о селекторе.

Работа с селекторами.

Все обычные правила CSS состоят из селекторов и соответствующих им блоков определений. В этом разделе будет произведено детальное рассмотрение используемых в CSS селекторов. Простейший селектор представляет собой название HTML-элемента. Такой селектор называется "простым селектором" (рис. 5.1).

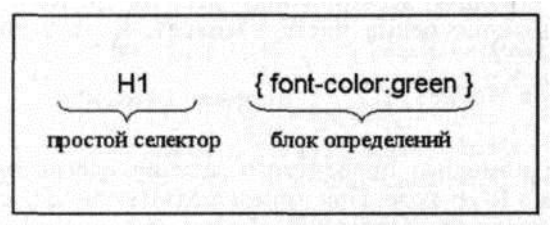

Рис. 5.1. *Пример простого селектора*

Простые селекторы употребляются наиболее часто. Однако язык CSS2 позволяет использовать более широкие возможности для создания таблиц стилей. Например, можно описать свойства HTML-элемента, который является дочерним по отношению к другому HTML-элементу. Также можно использовать селектор, правила которого будут применяться только при определенных действиях пользователя. Существуют также специальные селекторы, которые позволяют обращаться к различным группам HTML-элементов, например, ко всем элементам с определенным значением атрибута id.

Универсальный селектор

Универсальный селектор применяет установки, указанные в его блоке определений, ко всем HTML-элементам. Обозначается универсальный селектор символом * - звездочка. Но в том случае, если этот селектор используется в сочетании с другим (или другими) селекторами, то символ * может быть опущен. Например, записи:

*. myclass и .myclass —эквивалентны;

*# id_name и # id_name — также эквивалентны.

Пример задания:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN ">
<HTML>
<HEAD>
<TITLE>
```

Пример заключения таблицы стиля в SGML-комментарии

```
</TITLE>
<STYLE type ="text/css">
* (color=olive}  /* правило с использованием универсального селектора*/
H1 {color=red}  /* правило с использованием простого селектора*/
   *.myclass {font-color=blue}  /*использование универсального селектора в сочетании с селектором классов*/
</STYLE>
<BODY>
…. .....текст документа.....
.....текст документа.....
.....текст документа.....
</BODY>
</HTML>
```

Селектор классов

Используя селектор классов, разработчик может обращаться к группе разнородных HTML-элементов, принадлежащих к одному классу (имеющих одноименное значение атрибута class). Селектор классов показан на рис. 5.2. Сначала, через запятую, указываются названия HTML-элементов, а затем, через точку, следует имя класса, к которому эти элементы должны принадлежать, чтобы к ним было применено данное правило.

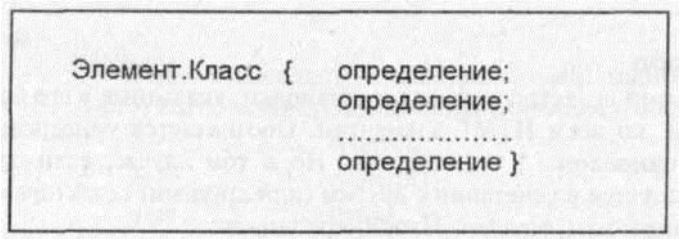

Рис. 5.2. *Схема использования селектора классов*

Например, чтобы правило применялось ко всем элементам, принадлежащим к классу superclass, используется следующая запись:

.superclass {определение}

Или другой пример:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN ">
```

<HTML>

<HEAD>

<TITLE>

Пример заключения таблицы стиля в SGML-комментарии

</TITLE>

<STYLE type ="text/css">

H3{color : green}

H3.titlepage (color: red}

H3.indexpage {color: blue}

</STYLE>

</HEAD>

<BODY>

...текст документа .....

...текст документа .....

...текст документа .....

</BODY> </HTML>

При таком задании все заголовки H3, принадлежащие к классу titlepage (class = titlepage), будут отображаться красным цветом. Принадлежащие к классу indexpage (class = indexpage) — синим цветом. Все остальные H3-заголовки будут иметь зеленый цвет. Указание класса относится только к одному, слева расположенному селектору. Это значит, что правило

PRE, P.chapter1 {color: red)

будет применено ко всем элементам PRE и к элементам P класса chapter, а не к элементам PRE и P класса chapter.

Селектор ID-имен

Селектор ID-имен применяется аналогично селектору классов. Отличие заключается в том, что селектор ID-имен фильтрует HTML-элементы не по классу (атрибуту class), а по id-именам (атрибуту id). Синтаксис селектора ID-имен имеет следующий вид: сначала пишется имя HTML-элемента, а затем, через символ «#», прописывается его id-имя.

Элемент#id_имя { определение; определение;
_____ определение }

РИС. 5.3. *Схема использования селектора ID-имен*

Например, правило

P#famous {font-family: Heretz}

будет применено только к тем HTML-элементам P, у которых для атрибута id указано значение "famous". Чтобы обратиться ко всем HTML-элементам, имеющим атрибут id = "famous", достаточно следующей записи:

 # famous{определение;

определение;

……

определение}

В одном правиле может содержаться несколько селекторов. Допустим, разработчику требуется применить одно и то же правило к нескольким разным HTML элементам, не объединённым в классы и имеющим разные id имена. Можно, конечно, написать отдельное правило для каждого такого элемента, но более эффективным и правильным будет перечислить, через запятую селекторы HTML элементов в списке селекторов одного правила. Например:

H1, P, Q {color: lightgrey;

Font-family: Arial}

.my-class, B {color: black}

Селекторы контекстного окружения

0Возможности CSS2 позволяют учитывать расположение HTML-элементов в иерархическом дереве документа при задании их визуальных характеристик, а также учитывать их контекстное окружение.

Как уже говорилось HTML-элемент, заданный в содержимом другого HTML-элемента, является его потомком. Исходя из этого правила, каждый HTML-документ имеет свое иерархическое дерево. Например, для следующего файла, иерархическое дерево будет иметь вид, приведенный на рис. 5.4:

<HTML>

<HEAD>

<BASE href="www.hrt.hd.net ">

<TITLE>Название документа </TITLE>

<META http-equiv=content-type content= "text/html;charset=iso-8859-l"

>

</HEAD> <BODY>

.....текст документа

<P>.....текст абзаца </P>

<UL>

<LI>элемент списка

<Ы>элемент списка

<Ы>элемент списка

</UL>

.....текст документа

</BODY>

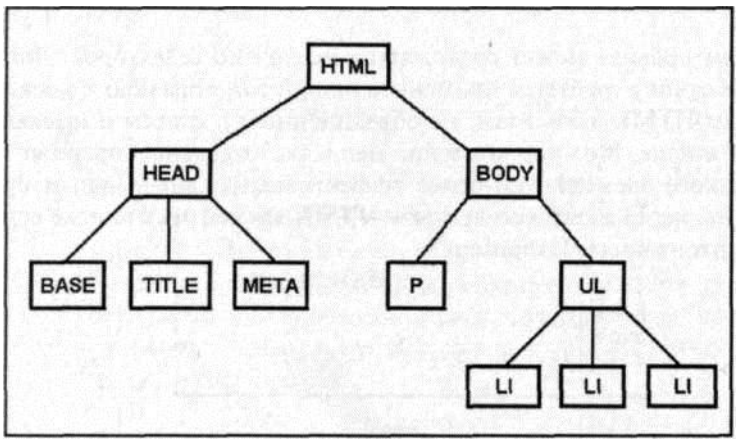

Рис. 5.4. *Иерархическое дерево HTML-документа*

В CSS возможно использование правил, которые будут применяться только к HTML-элементам, являющихся потомками других определенных элементов. При этом к таким же HTML-элементам, не являющимся потомками указанных элементов, данное правило применяться не будет. При задании такого правила через пробел указывается селектор родительского элемента и селектор элемента-потомка. Например, разработчику нужно установить синий цвет для цитат, только если они содержатся в тексте элемента PRE, то есть, если элемент CITE входит в состав элемента PRE:

<PRE>

отформатированный текст <CITE>цитата

</CITE>

отформатированный текст

</PRE>.

Для этого достаточно использовать следующее правило:

PRE CITE (color: blue}

В рамках одного CSS-правила допустим многократный переход родитель-потомок. Например, правило

DIV P EM {color: yellow;

font-family: Italic}

будет применено только к тем элементам EM, которые являются потомками элемента P, который, в свою очередь, должен являться потомком элемента DIV. Чтобы обратиться ко всем элементам EM, являющимися потомками второго уровня элемента DIV, используется запись DIV * EM. При этом символ "*" должен быть с обеих сторон выделен пробелами. Таким образом, правило, использующее иерархию HTML-элементов документа, имеет общий вид, приведенный на рис. 5.5.

Рис. 5.5. *Правило, использующее иерархию HTML-элементов документа*

Язык CSS позволяет обращаться к определенным HTML-элементам, учитывая их соседство с другими элементами. При этом какому-либо элементу можно задавать характеристики, которые будут к нему применены только в том случае, если он следует за другим указанным элементом. Например, чтобы обратиться только к тем неупорядоченным спискам, которые следуют после заголовка H3, используется следующая запись:

H3+UL{объявление;

объявление;

объявление}

Элементы H3 и UL называются сестринскими элементами. Символ "+" с обеих сторон выделяется пробелами. Таким образом, правило, учитывающее предыдущие HTML-элементы для описываемого HTML-элемента, имеет общий вид, представленный на рис. 5.6.

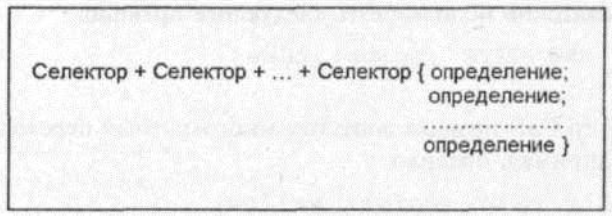

Рис. 5.6. *Правило, учитывающее соседство HTML-элементов*

Использование псевдоэлементов и псевдоклассов.

В рамках спецификации CSS2 определено некоторое количество псевдоэлементов и псевдоклассов, которые представляют собой элементы и классы, не входящие в иерархическое дерево HTML-документа и не образующие в нем структурных единиц. Иными словами, они являются специфичными элементами и классами языка CSS2 и используются им только в своих целях для задания специальных визуальных эффектов. Например, псевдоэлемент first-letter представляет собой первую букву фрагмента текста, first-liNe — первую строку и т.п. На данный момент использование ни одного псевдоэлемента не поддерживается браузерами Internet Explorer и Netscape Communicator. Поэтому заниматься их рассмотрением мы не будем. Информацию о них можно прочесть на сайте консорциума World Wild Web — www.w3c.com.

Из всех псевдоклассов сейчас поддерживаются псевдоклассы ссылок (элементов A). Все контекстные ссылки, задаваемые элементом A, в понимании CSS2 принадлежат одному из четырех псевдоклассов:

- классу visited посещенных ссылок;
- классу link не посещенных ссылок;
- классу active активных ссылок;
- классу hover ссылок, над видимым содержимым которых находится курсор мыши.

Например, чтобы задать красный цвет для ссылок, уже просмотренных пользователем, используется следующее CSS-правило:

A: visited {color: red}

Полное цветовое описание возможных состояний элемента A может иметь следующий вид:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN ">

<HTML>

<HEAD>

<TITLE>

Пример задания оформления ссылок средствами CSS

</TITLE>

<STYLE type ="text/css">

A : visited {color: lightgrey}

A : link {color: blue}

A : active {color: yellow}

A : hover {color: red}

</STYLE>

</HEAD>

<BODY>

.....текст документа.....

.....текст документа....

.....текст документа....

</BODY>

</HTML>

В данном случае непосредственно после открытия документа все его ссылки будут отображаться синим цветом. При наведении на ссылку курсора цвет ее изменится на красный. Сразу после щелчка по выбранной ссылке и до того момента, пока не произойдет загрузка целевого документа, цвет ссылки будет желтым. Затем, при откате из открывшегося документа по клавише "Назад" исходная ссылка будет восприниматься браузером как посещенная и отображаться серым цветом.

Правила каскадирования

Обработка HTML-файла, содержащего каскадную таблицу стилей, имеет следующие этапы:

1 этап. Синтаксический анализ документа и обработка синтаксических ошибок.

2 этап. Построение иерархического дерева HTML-элементов, образующих данный документ.

3 этап. Применение таблицы стилей в соответствии с правилом установки

каскадирования и иерархическим деревом документа.

Иерархическое дерево документа используется для передачи установок родительского элемента элементам-потомкам, то есть благодаря ему реализуется процесс наследования.

Все каскадные таблицы стилей, используемые для форматирования документа, выстраиваются в каскад в соответствии со своим приоритетом. По этому каскаду и "прогоняется" документ. Сначала применяются таблицы стилей с меньшим приоритетом, затем — с большим приоритетом. При этом для одноименных HTML-элементов правила таблиц с большим приоритетом перекрывают правила таблиц с меньшим приоритетом.

Если для одного HTML-элемента указано несколько одинаковых правил в рамках одной таблицы стилей, то приоритет имеет правило заданное последним. Все таблицы стилей в качестве своего источника могут иметь разработчика, пользователя или браузер. Самым низким приоритетом обладают стилевые установки, используемые браузером по умолчанию. Таблицы стилей, заданные разработчиком, имеют приоритет над таблицами, применяемыми к документу пользовательскими таблицами стилей.

По умолчанию к браузеру никаких пользовательских таблиц не подключено, но это всегда можно сделать. Для этого написанная Вами таблица стилей помещается в .css-файл. Затем в браузере в разделе "Сервис | Свойства обозревателя | Оформление"

выставляем флажок "Оформлять, используя стиль пользователя" и далее указываем название файла с таблицей стиля (рис. 5.7).

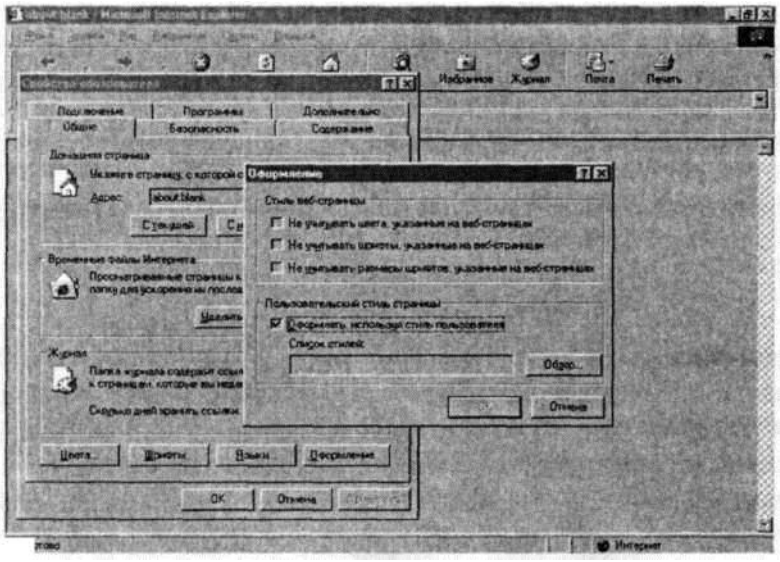

Рис. 5.7. *Окно подключения пользовательских таблиц стилей в браузере Internet Explorer*

Установки таблицы стилей, подключенной к браузеру таким образом, перекрывают установки, используемые браузером по умолчанию. Например, по умолчанию в браузере Internet Explorer принят серый цвет фона, а можно сделать его красным. И тогда все HTML-документы, для которых не задано фоновое оформление (цвет фона или фоновое изображение), будут иметь в окне этого браузера фон красного цвета.

Пользовательские таблицы, подключенные к браузеру, перекрываются любыми авторскими таблицами стиля, содержащихся в коде самого HTML-документа, внедренными в него с помощью правила @import или подключенных к нему элементов LINK.

Однако можно сделать так, чтобы пользовательские правила перекрывали авторские. Для этого нужно у пользовательских правил выставить правило-флаг !impotent. Например:

BODY {background: laightgrey ! important;

font-color:gray ! important}

Правило-флаг *!impotant* может выставить и автор HTML-документа., но правило *!impotant* пользователя имеет приоритет над правилом !impotant автора. При задании и подключении нескольких таблиц стилей каждая последующая заданная (или подключенная) таблица имеет приоритет над предыдущей. Говоря точнее, каждый последующий элемент STYLE или LINK имеет приоритет над предыдущим.

В этом отношении между таблицами, задаваемыми элементами STYLE и LINK, различий не делается. Т.е., если задание подключенной таблицы (элемент LINK) следует после внедренной таблицы (элемент STYLE), то подключенная имеет приоритет над внедренной. Верно также и обратное. В содержимое одного элемента STYLE с помощью правила ©import может быть импортировано несколько таблиц стилей, после чего могут следовать другие правила, задаваемые непосредственно в этом элементе STYLE. Внутри одного элемента STYLE все импортированные установки имеют меньший приоритет по отношению к установкам, заданным в нем непосредственно. Надо помнить, что все правила @import задаются в таблице стилей до определения остальных правил. Если в элемент STYLE импортировано несколько CSS-таблиц, то каждая последующая импортируемая таблица обладает большим приоритетом над предыдущей.

Кроме всего вышеперечисленного, приоритет какого-либо правила по сравнению с другими правилами, применяемыми к одному и тому же элементу, определяется, исходя из специфичности правила.

Например, к элементу

<H3 class="vas" id="ret"x/H3>

могут быть применены следующие правила:

H3 {color:gray}

H3.vas {color:red}

H3#ret(color:green}

H3, H2, H1 (color:blue)

Правило H3 {color:gray} относится только к заголовкам третьего уровня. Правило H3.vas {color:red) применяется только к элементам H3, принадлежащим к классу "vas", то есть является более специфичным по сравнению с предыдущим правилом.

Самым специфичным является правило H3#ret{color:green}, которое применяется к конкретному элементу, элементу H3 с id-именем ret. Все эти правила могут быть применены к элементу <H3class="vas"id="ret"></ H3>. Но если все они заданы вместе, то к этому элементу будет применено самое специфичное правило H3#ret{color:green}, независимо от того, в каком порядке эти правила заданы.

К HTML-элементам применяются правила, обладающие наибольшей специфичностью, независимо от порядка задания. Специфичность правила определяется по следующей схеме:

- подсчитывается число атрибутов ID в данном правиле (число a);
- подсчитывается число других атрибутов и псевдоклассов в данном правиле (число b);
- подсчитывается число названий HTML-элементов в данном правиле (число c);

Псевдоэлементы и псевдоклассы считаются как обычные элементы и классы.

Далее, из чисел a, b и c составляется число abc, которое и показывает специфичность правила. Например:

** {.....*	*}*	*a = 0 ,*	*b=0, c=0*	*->*	*специфичность = O*
H3 {	*)*	*a-0,*	*b-0, c=1*	*->*	*специфичность = 1*
H3, H2 {.....	*)*	*a = 0,*	*b=0, c=2*	*->*	*специфичность = 2*
H3, P + UL {.....	*}*	*a = 0,*	*b=O, c=3*	*->*	*специфичность = 3*
H3. var {.....	*}*	*a-0,*	*b=1, c=1*	*->*	*специфичность = 11*
H3.var, PRE {.....	*}*	*a = 0,*	*b=1, c=2*	*->*	*специфичность = 12*
H3.var, .yty {.....	*}*	*a = 0,*	*b=2 c=l*	*->*	*специфичность = 21*
H3#ret {	*}*	*a = 1,*	*b=0, c=1*	*->*	*специфичность = 101*
*P#kil, UL LI, *.tgt. {*	*}*	*a=l,*	*b=l, c=3*	*->*	*специфичность = 113*

Возможности CSS2 по созданию аппаратно-зависимых таблиц стилей.

Возможности языка CSS2 позволяют разработчикам создавать аппаратно-зависимые таблицы стилей. Или, говоря другими словами, таблицы стилей для определенных устройств вывода: монитора, принтера, телевизора, синтезатора речи и т.п. В силу технических особенностей этих устройств один и тот же документ может иметь у них разное визуальное представление (внешний вид). Например, общеизвестным считается тот факт, что для отображения текста на экране монитора требуется больший размер шрифта, чем для его печати. Поэтому для достижения наилучшего качества

представления документа рекомендуется создавать стилевые таблицы под каждое устройство вывода, для которого он может предназначаться. Указать целевое устройство можно с помощью правила @media, например:

```
@media print {BODY {background : lightgrey;
 font-family: Sans serif}
B (color: blue}
}
```

Внешнюю таблицу стилей можно определить для определенного целевого устройства с помощью атрибута media элемента Link, например

```
<Link rel = "stylesheet" type = "text/css" media = "print"
href = "mystylefile.css">
```

Правило @media

Правило @media задает список типов устройств, которым предназначается таблица стилей, расположенная далее и заключенная в фигурные скобки, например:

```
@media tv (
    BODY { font-size: 10 pt;
    font-color: black}
    }
```

Если таблица предназначается нескольким устройствам разного типа, то их перечисление осуществляется через запятую, например:

```
@media print, screen {
    BODY {font-color: san serif;
    font-color: 12 pt ;
    font-color: black}
}
```

Типы устройств, распознаваемые CSS2.

Каскадные таблицы стилей, написанные на языке CSS2, могут быть конкретно предназначены следующим типам устройств:

all — этот тип предназначен для обозначения устройств всех типов. Его задание говорит о том, что таблица стилей предназначена для всех возможных устройств вывода. Данное значение используется по умолчанию.

print — обозначает печатные устройства и говорит о том, что таблица стилей будет

применена к документу при его печати.

screen — обозначает цветной экран монитора.

tv — обозначает устройства типа телевизора, для которых характерны следующие характеристики:

- цветное изображение;
- низкое разрешение;
- передача звука.

aural — обозначает синтезаторы речи. Projection — обозначает устройства, предназначенные для проведения презентаций

(проекторов) и для устройств печати плакатов.

braille, embossed — обозначают устройства, используемые для прочтения и печати документов, построенных на азбуке Брайля (для слепых пользователей). Предназначены для документов, требующих тактильного представления.

Пример использования

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN ">
<HTML>
<HEAD>
<TITLE>пример аппаратно-зависимого документа</TITLE>
<STYLE type ="text/css">
@media screen {
правила, применяемые к документу, при выводе его на экран монитора
правила, применяемые к документу, при выводе его на экран монитора
правила, применяемые к документу, при выводе его на экран монитора
}
@media print {
правила, применяемые к документу, при его печати
правила, применяемые к документу, при его печати
правила, применяемые к документу, при его печати
}
@media aural {
правила, применяемые к документу, при его звуковом воспроизведении
правила, применяемые к документу, при его звуковом воспроизведении
правила, применяемые к документу, при его звуковом воспроизведении
                                                    }
правила, применяемые к документу, в не зависимости от устройства вывода
правила, применяемые к документу, в не зависимости от устройства вывода
правила, применяемые к документу, в не зависимости от устройства вывода
</STYLE>
</HEAD>
……текст документа……
……текст документа……
……текст документа……
</BODY>
</HTML>
```