ГЛАВА 3. CASCADING STYLE SHEETS (CSS)

§ 3.1. Каскадные таблицы стилей

Знакомство с HTML лишь первый этап в процессе обучения основам интернет-технологий. Сейчас мы уже научились создавать различные элементы web-страниц. Теперь пришло время познакомиться с возможностями их оформления.

Стиль web-страницы или CSS (Cascading Style Sheets, каскадные таблицы стилей) - набор параметров форматирования, который применяется к элементам документа, чтобы изменить их внешний вид. Возможность работы со стилями издавна включают в развитые издательские системы и текстовые редакторы, тем самым, позволяя одним нажатием кнопки придать тексту заданный, заранее установленный вид. Дизайнерам сайта тоже это доступно, когда цвет, размеры текста и другие параметры хранятся в определенном месте и легко применяются к любому тегу. Еще одним преимуществом стилей является то, что они предлагают намного больше возможностей для форматирования, чем обычный HTML. Таким образом, если HTML задает «скелет» нашего человечка (страницы), то CSS позволяет задать цвет глаз, кожи, вид и форму одежды, и т.д.

Элементами страницы являются блоки текста, графика и встроенные в страницу приложения, размер и границы каждого из них в рамках HTML-разметки задаются с разной степенью точности. Размер графики и приложений можно задать с точностью до пикселя, а вот размеры текстовых блоков - нельзя, они вычисляются браузером автоматически на основе относительного размера шрифта. Каскадные таблицы стилей призваны разрешить это противоречие.

Кроме размера элементов, таблицы стилей позволяют определить цвет и начертание текстового фрагмента, изменять эти параметры внутри текстового блока, выполнять выравнивание текстового блока относительно других блоков и компонентов страницы.

CSS позволяет полностью изменить свойства элемента страницы, заданные по умолчанию. Например, тег ... выводит текст

курсивом. С помощью каскадных таблиц стилей можно задать, например, чтобы он был прямой и подчеркнутый.

```
<i style="text-decoration:underline;font-
style:normal;"> текст должен быть выведен курсивом
</i>
```

Практическая значимость CSS для Web-разработки заключается в возможности применения общего стиля оформления для всех страниц сайта.

Весь процесс разработки сводится к следующим этапам:

1. Определение всех видов страниц: стартовая, страницы навигации, информативные страницы и т.п. в зависимости от информационной и пользовательской направленности сайта.
2. Разработка для каждого типа страниц своего «скелета» - набора элементов web-страницы.
3. Разработка системы навигации сайта и ее представления на страницах.
4. Разработка стиля оформления для каждого типа страниц с помощью таблицы описания стилей оформления.
5. Разработка множества страниц указанного типа по подготовленным шаблонам оформления.

До появления мощного инструмента CSS оформление интернетстраниц осуществлялось непосредственно внутри содержимого документа – это было долго и очень неудобно. Однако с появлением CSS стало возможным принципиальное разделение содержания и представления документа. За счѐт этого нововведения стало возможно лѐгкое применение единого стиля оформления для массы схожих документов, а также быстрое изменение этого оформления.

Помимо уже указанных плюсов применения таблиц стилей выделяют также следующие:

- Возможность вариации дизайна страницы для разных устройств просмотра. Например, на экране дизайн будет рассчитан на большую ширину, во время печати меню не

будет выводиться, а на телефоне, планшете и т.п. представляться в упрощенном виде.

- Уменьшение общего объема страниц за счет хранения единого файла с оформлением. Браузер загружает только структуру документа и данные, хранимые на странице, а представление этих данных загружается браузером только один раз и кэшируется.
- Быстрота и простота корректировки дизайна всех страниц. Стили, как правило, хранятся в одном или нескольких специальных файлах, ссылка на которые указывается во всех документах сайта – а значит достаточно обновить оформление только в них, а не просматривать все страницы подлежащие корректировке. Несомненно, наиболее ярко проявляется при работе с большим сайтом.
- Простота единообразного оформления. Сайт это не просто набор связанных между собой документов, но и единое расположение основных блоков, и их вид. Никаких разночтений, неточно подобранных цветов, размеров и положений. Страницы всегда будут похожи друг на друга как близнецы, различаясь лишь контентом.
- Дополнительные возможности оформления. Многие из возможностей, предоставляемыми таблицами стилей невозможно или крайне затруднительно реализовать с помощью традиционной HTML-разметки. Например, растягивание фонового изображения под размер окна.

Принципы форматирования таблиц стилей

CSS представляет собой свой собственный язык, который в большинстве своем не совпадает с HTML. Именно с этим связано большинство проблем на начальных этапах знакомства с этой технологией.

Основным понятием описания стиля выступает селектор — это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают теги, классы и идентификаторы: имя тега (без уголков), значение его параметра id, перед которым указывается знак #, или же имя так называемого класса.

Общий способ записи имеет следующий вид: вначале пишется имя селектора, например, IMG, это означает, что все стилевые параметры будут применяться к тегу <img>, затем идут фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия. Стилевые свойства разделяются между собой точкой с запятой, в конце этот символ можно опустить.

```
селектор: {имя_параметра1:значение; имя_параметра2:
значение;…имя_параметраN:значение }
```

Заметим, CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от желания разработчика и удобства чтения описания[12,13,14].

Итак, в качестве селектора может выступать имя тега. Данный вариант используется для определения общего стиля данного элемента для всех страниц. Это дает возможность скомпоновать страницы из логических элементов, а стиль отображения этих элементов описать во внешнем файле.

Следующее форматирование изменит все заголовки первого уровня в документе.

```
h1 {color: red; font-size:14pt}
```

Очень удобно использовать таблицы стилей для форматирования различных элементов страницы одинаковым набором значений параметров – в этом случае достаточно их перечислить через запятую перед общим набором параметров.

```
H1,p {color: red; font-size:14pt}
```

Кроме того, как известно, одни элементы могут быть вложены в другие. Иногда случается необходимость выделить такое вхождение. В подобных случаях используют так называемые контейнерные селекторы, которые задают исключения из общих правил форматирования, определенных с помощью простых селекторов или по умолчанию. В данном случае селектор состоит из имени тегаконтейнера и имени вложенного тега, разделенных пробелом.

Например, если в абзаце встретился тег полужирного форматирования, текст, содержащийся внутри этих двух элементов, будет выделен 16 размером шрифта.

```
<style>
p b {font-size:16pt}
</style>
```

Совсем другая ситуация возникает, если в пределах одного тега необходимо задать не один тип форматирования. Тогда на помощь приходят селекторы – классы.

Для того чтобы отнести элемент web-страницы к определенному классу, нужно воспользоваться параметром class этого элемента. При этом селектор формируется как имя тега, за которым через разделительную точку следует имя класса, наподобие работы с объектно-ориентированными языками.

```
<style>
  p.red {color:red }
p.green {color:green }
</style> ...
<p class=red>
Этот абзац напечатан красным цветом </h1>
<p class=green>
Этот абзац выделен зеленым цветом </h1>
<p>
Этот абзац обычный- черный </h1>
```

В данном примере класс абзаца red имеет одно описание стиля, а класс абзаца green совершенно другое. При этом каждый из этих классов нельзя применить к другим элементам страницы, например, заголовку или списку. Если же имя элемента не задано, то класс можно применить к любому сопоставимому элементу (раскрасить красным картинку как абзац нам вряд ли удастся). В примере в одном классе сосуществуют параграф и гипертекстовая ссылка.

```
    <style>
.all {color:white; background-color: green;}
</style> ...
<p class=all>
Этот параграф отобразиться белым цветом на зеленом
фоне </p>
...
<p>
Эта <a class=all>гипертекстовая ссылка</a> будет
напечатана белым цветом на зеленом фоне.
</p>
```

Причем начальную точку в имени класса в данном случае можно опускать, она приводится только из соображений единообразия в описании.

Объектная модель документа (Document Object Model) позволяет обратиться к любому элементу страницы по его уникальному имени – так называемому «идентификатору». В качестве объектов, которых можно именовать и обращаться к ним по имени выступает сам документ, картинки, параграфы, приложения и т.п. Данная возможность используется, например, для задания этому элементу определенных свойств или при программировании страниц на стороне клиента, но подробнее о последнем будет изложено в четвертой главе. Применение в качестве селектора имени идентификатора целесообразно лишь в случае наличия уникального элемента, которому необходимо задать описание, в остальных случаях стоит использовать имена тегов и классы.

Пусть в нашем HTML-документе есть уникальный абзац с заданным именем (<p id=my_p>), описание его свойств в таблице стилей может выглядеть следующим образом:

```
  #my_p {color:green; font-size:14pt}
```

Правила применения стилей

Существует ряд дополнительных правил, которые необходимо знать при описании стиля.

Правило первое – о форме записи: для селектора допускается добавлять каждое стилевое свойство и его значение по отдельности**.**

```
p { background: olive; } p{
color: white; }
p{ border: 1px solid black; }
```

Следует заметить, что такая форма записи не удобна - приходится повторять несколько раз один и тот же селектор, да и легко запутаться в их количестве. Целесообразнее писать все свойства для каждого селектора вместе – так будет нагляднее и уменьшит объем файла таблицы стиля. Указанный набор записей в таком случае получит следующий вид.

```
p {
  background: olive;
color: white;
  border: 1px solid black;
}
```

Правило второе - чем ниже, тем главнее: если для селектора задается сначала свойство с одним значением, а затем то же свойство, но уже с другим значением, то применяться будет то значение, которое в коде описано ниже.

Например, при данном описании таблицы стилей абзац будет напечатан красным цветом шрифта:

```
p { color: green; }
```

```
p { color: red; }
```

Конечно, таких казусов стоит вообще избегать, и отслеживать при подключении нескольких стилевых таблиц с одинаковыми селекторами.

Правило третье – корректность записи значений: для каждого свойства может быть только соответствующее его функции значение.

Например, для font-size, который устанавливает размер текста, в качестве значений недопустимо использовать слова.

Правило четвертое – использование комментариев: комментарии применять в отладочных или учебных целях, а при выкладывании сайта в сеть убирать.

Комментарии записывают с помощью конструкции /* ... */, их можно добавлять в любое место CSS-документа, а также писать текст комментария в несколько строк, вложенные комментарии недопустимы. Комментарии позволяют сделать пояснения по поводу использования того или иного стилевого свойства, выделить разделы или записать свои заметки. Они позволяют легко вспомнить логику и структуру селекторов, повысить разборчивость кода, но вместе с тем, увеличивают объем документов, что отрицательно сказывается на времени их загрузки.

```
table {
  width: 200px; /* Ширина таблицы */
margin: 10px; /* Поля  вокруг элемента */
  float: left; /* Обтекание по правому краю */}
```

Эти четыре простых правила позволяют избежать множества ошибок при задании таблиц стилей.

Встраивание таблицы стилей в HTML-документ

Для применения таблицы стилей к гипертекстовому документу необходимо ее связать с ним или встроить в него. Сделать это возможно одним из 4 способов.

Самый простой из способов - вставка стиля непосредственно в элемент. Так называемое «переопределение стиля». Именно этим способом нам удалось заставить тег для вывода текста курсивом

выводить его прямым и подчеркнутым. Осуществляется такое переопределение с помощью параметра STYLE у данного элемента web-страницы. В данном случае заголовок первого уровня будет выведен нежирным курсивом в 10 пунктов.

```
<h1 style="font-weight:normal; font-style:italic;
font-size:10pt;">Заголовок первого уровня </h1>
```

Параметр style можно применить внутри любого элемента интернет-страницы. Например, с помощью style возможно определить ширину и выравнивание элемента hr (горизонтальной линии):

```
<hr style="width:50px;">
```

Следует заметить, что данный вид встраивания рекомендуется применять на сайте лишь в крайних случаях или вообще отказаться от их использования. Во-первых, это увеличивает объем web-страниц, а вовторых, усложняет редактирование.

Второй способ предполагает вставку стиля непосредственно в HTML документ путем добавления контейнерный тег STYLE – «внедрение каскадных таблиц стилей в ткань HTML-документа». В данном случае элемент STYLE позволяет определить стиль отображения для стандартных элементов HTML-документов, произвольных классов (селектор class) и HTML-объектов (селектор id).

Указанный тег добавляется в заголовок web-страницы, правила задания стилей стандартные, о которых будет рассказано чуть позже.

```
<head> <style>
p {color:darkred;text-align:justify;font-size:8pt;}
</style>
</head> <body>
...
<p>
Оформление этого параграфа задано с помощью тега
стиля</p> ...
</body>
```

В данном случае все добавляемые на эту страницу параграфы будут отображаться стилем из элемента STYLE, если только стиль не будет переопределен каким-либо способом.

Следующим способом присоединения к web-странице таблицы с оформлением ее элементов является ее связывание с таблицей стилей, заданной в отдельном файле. Ссылка на файл с описаниями оформления элементов осуществляется при помощи тега LINK, который размещается в теге HEAD. Файл с описанием стиля представляет собой текстовый файл с характеристиками форматирования, заданными в соответствии с правилами идентичными написанию в теге и параметре тегов STYLE.

Параметрами тега LINK при работе с таблицами стилей являются REL, значением которого всегда является таблица стилей ("stylesheet"), подтип таблицы стилей, задаваемый параметром TYPE –текстовая таблица стилей ("text/css"). Параметр HREF задает адрес внешнего файла, содержащего описания стилей. Это может быть ссылка на файл с любым именем, а не только на файл с расширением *.css.

```
<link type="text/css" rel="stylesheet"
href="http://www.narfu.ru/css1.css">
```

Использование данного способа встраивания таблицы стилей в web-страницы является наиболее универсальным и удобным методом добавления стиля на сайт - стили хранятся в одном файле, а в webдокументах указывается только ссылка на него.

Последним, четвертым способом добавления к интернет-странице стиля оформления ее элементов является импорт описания стилей.

«Импортировать», подключать стиль можно либо внутрь элемента STYLE, либо внутрь внешнего файла, который представляет собой описание стиля. Оператор импорта стиля должен предшествовать всем прочим описаниям стилей:

```
<style>
@import:url(http://www.narfu.ru/style.css) a
{color:cyan;text-decoration:underline;}
```

```
</style>
```

Этот способ можно применять совместно с другими, за исключением первого. Для подключения стиля достаточно после ключевого слова @import указать путь к стилевому файлу одним из двух приведенных способов — с помощью url или без него.

```
@import url("имя файла") типы носителей;
@import "имя файла" типы носителей;
```

Например,

```
@import "/style/palm.css";
BODY {  font-family: Arial, Verdana, sans-
serif;  background: white;
 color: black;
}
```

Все описанные методы использования CSS могут применяться как самостоятельно, так и в сочетании друг с другом. В этом случае необходимо помнить об их иерархии: сначала всегда применяется внутренний стиль, затем стиль на странице и в последнюю очередь связанный стиль.

Типы стилей. Каскадирование

Аббревиатура CSS означает каскадные таблицы стилей (Cascading Style Sheets), определяя основной закон данных таблиц – каскад. Под каскадом в данном случае понимается «одновременное применение разных стилевых правил к элементам документа — с помощью подключения нескольких стилевых файлов, наследования свойств и других методов»[12].

Для того чтобы определить, какое в итоге правило будет применено к элементу, введены определенные приоритеты стилей.

Самым маленьким приоритетом обладает стиль браузера. Им называют оформление, которое по умолчанию применяется к элементам

web-страницы программой просмотра. Именно он и предстает взору, если загрузить в браузер «скелет» страницы. Например, заголовок страницы, формируемый тегом <H1>, в большинстве браузеров выводится шрифтом с засечками размером 24 пункта.

Стиль пользователя может включить посетитель сайта через настройки браузера, традиционно через команду «Оформление» или «Параметры стиля». Этот стиль имеет больший приоритет и переопределяет исходное оформление документа.

Стиль автора добавляет к документу его разработчик. Именно об авторском стиле, возможных способах подключения и параметрах ниже и будет рассказано. Он переопределяет стили пользователя и браузера.

Все три типа стилей могут спокойно существовать друг с другом, если они не пытаются изменить вид одного элемента. В случае возникновения противоречия в силу вступает приоритет стилей.

Помимо указанных стилей возможно влиять на оформление документа с помощью ключевого слово !important, добавляя его к стилю автора и пользователя.

Слово !important позволяет повысить приоритет стиля или его важность. в случае несовпадения стиля автора страницы и пользователя к конкретному элементу страницы.

При использовании пользовательской таблицы стилей или одновременном применении разного стиля автора и пользователя к одному и тому же элементу, браузер руководствуется принципами, приведенными в таблице 33.

Таблица 1. Наличие слова !important в стилях

Авторский стиль	**Пользовательский стиль**	**Результат**
присутствует	-	применяется стиль автора
-	присутствует	применяется стиль пользователя
-	-	применяется стиль автора
присутствует	присутствует	применяется стиль пользователя

Для добавления важности стилевому свойству вначале пишется желаемое стилевое свойство, затем через двоеточие его значение и в конце после пробела указывается ключевое слово !important.

```
Свойство: значение !important
```

Повышение важности требуется не только для регулирования приоритета между авторской и пользовательской таблицей стилей, но и для повышения специфичности определенного селектора.

Наследование

Разговор о каскадных таблицах стилей будет не полным, без упоминания системы наследования. «Наследованием называется перенос правил форматирования для элементов, находящихся внутри других. Такие элементы являются дочерними, и они наследуют некоторые стилевые свойства своих родителей, внутри которых располагаются. Наследование позволяет задавать значения некоторых свойств единожды, определяя их для родителей верхнего уровня»[12].

Например, таблица состоит из тега-контейнера <table>, внутри которого добавляются помещаются строки <tr>, а внутри которых располагаются ячейки <td>. Если в стилях TABLE задать цвет текста, то он автоматически устанавливается для всех содержимого ячеек.

При этом следует понимать, что не все стилевые свойства наследуются, их перечень можно найти в различных справочниках по CSS. Так, border задает рамку вокруг таблицы в целом, но никак не вокруг ячеек.

В большинстве случае задавая цвет и размер для тега BODY, мы путем наследования задаем их и для всех элементов страницы.

```
BODY {font-family: Arial, sans-serif; color: green;}
```

Лишь для явного изменения, каких-либо свойств элемента необходимо переопределить его стиль.

Единицы измерения CSS

Для задания размеров различных элементов страницы в CSS используются два вида единиц измерения: абсолютные и относительные. Относительные единицы определяют размер элемента относительно значения другого размера, абсолютные единицы не зависят от устройства вывода.

Наиболее используемыми относительными единицами являются пиксели (px) и проценты (%), они зависят от разрешения монитора, его размеров и других системных настроек. Кроме того, выделяют высоту шрифта элемента (em), и высоту символа x (ex). Традиционно значение ex равно удвоенному em.

Самой распространенной единицей является пункт, который используется для указания размера шрифта. Именно эта единица фигурирует в текстовых редакторах и издательских системах. Все используемые в интернет-приложениях абсолютные единицы приведены в таблице 34.

Таблица 2. Абсолютные единицы измерения

Единица	Описание
in	Дюйм (1 дюйм равен 2,54 см)
cm	Сантиметр
mm	Миллиметр
pt	Пункт (1 пункт равен 1/72 дюйма)
pc	Пика (1 пика равна 12 пунктам)

Заметим, что использование абсолютных единиц измерения в большинстве случаев должно быть ограничено. Иначе заданный для экрана монитора небольшой размер шрифта может оказаться нечитаемым для людей с ослабленным зрением – его настройки невозможно изменить в браузере, или же при просмотре на небольших, «наладонных» устройствах. Хотя, последнее легко решается при подключении разных таблиц стилей для разных устройств.

Типы носителей информации

Широкое развитие различных платформ и устройств позволяющих работать с интернет-приложениями заставляет разработчиков создавать специальные версии сайтов, как неизбежное и необходимое звено их развития. С учетом этого в CSS введено понятие типа носителя, когда стиль применяется только для определенного устройства. В табл. 35 перечислены основные типы носителей информации.

Таблица 3. Типы носителей и их описание.

Тип	Описание
all	Все типы. Это значение используется по умолчанию.
aural	Речевые синтезаторы, а также программы для воспроизведения текста вслух. Сюда, например, можно отнести речевые браузеры
braille	Устройства, основанные на системе Брайля, которые предназначены для слепых людей.
handheld	«Ручные» компьютеры и аналогичные им аппараты: планшеты, смартфоны, телефоны и т.п.
print	Печатающие устройства вроде принтера
projection	Проектор
screen	Экран монитора
tv	Телевизор

В CSS для указания типа носителей применяются команды @import и @media, с помощью которых можно определить стиль для элементов в зависимости от того, выводится документ на экран или на принтер.

При импортировании стиля через команду @import тип носителя указывается после адреса файла. При этом допускается задавать сразу несколько типов, упоминая их через запятую.

Например,

```
<html>
 <head>
    <style type="text/css">
```

```
    @import "main.css" screen; /* Стиль для вывода
на монитор */
    @import "smart.css" print, handheld; /* Стиль
для печати и смартфона */
  </style>
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

В данном примере подключаются основной стиль для экрана монитора (main.css) и специальный для печати и отображения на смартфоне (smart.css).

Команда @media позволяет указать тип носителя для глобальных или связанных стилей. После ключевого слова @media идет один или несколько типов носителя, перечисленных в таблице, разделяемых между собой запятой. После чего следуют обязательные фигурные скобки, внутри которых идет обычное описание стилевых правил.

```
@media тип носителя 1 {Описание стиля для типа
носителя 1}
@media тип носителя 2 {Описание стиля для типа
носителя 2}
```

Ниже показано, как задать разный стиль для печати и отображения на мониторе.

```
<head>
  <style type="text/css">
   @media screen {/* Стиль для отображения в
браузере */     BODY {
     font-family: Arial, Verdana, sans-serif; /*
Рубленый шрифт */
     color: #000080; /* Цвет текста */
    }
H1 {
```

```
     background: #faf0e6; /* Цвет фона */
border: 2px dashed maroon; /* Рамка вокруг
заголовка */
    }
    P {     margin-top: 0.5em; /* Отступ сверху */
}
   }
   @media print { /* Стиль для печати */
BODY {
     font-family: Times, 'Times New Roman', serif;
/* Шрифт с засечками */
    }
    H1, H2, P { color: black; /* Черный цвет текста
*/
    }
   }
  </style>
 </head>
 <body>
  <h1>Татьяна Снежина</h1>
  <h2> Из повести "Дождь" 1990 год, Москва</h2>
<p> Звезды....Звезды-Это счастливые мгновения,
которые, отслужив свое на Земле, возвысились в небо
и остались там, чтобы вечно напоминать о себе по
ночам, когда люди освобождаются ото всего, могут
поднять головы и устремить свои взоры туда, где
светит их ушедшее когда то счастье. Но иногда
звезды покидают небо и вновь спускаются на Землю,
чтобы озарить счастьем своего избранника. Наверно,
поэтому мы так часто с тоской смотрим в глубине
ночи на звезды, боясь пропустить свою счастливую
звезду. И даже если не находим ее, то все равно
звездный свет, подобно целебному бальзаму,
смазывает наши душевные раны, сглаживает рубцы на
сердце и очищает нас от земной суеты.</p>
 </body>
</html>
```

В данном примере вводится два стиля — один для изменения вида элементов при их обычном отображении в браузере, а второй — при

выводе страницы на печать. При этом облик документа для разных носителей может сильно различаться между собой.

Команда @media применяется в основном для формирования одного стилевого файла, который разбит на блоки по типу устройств. Иногда же имеет смысл создать несколько разных CSS-файлов — один для печати, другой для отображения в браузере — и подключать их к документу по мере необходимости. В подобном случае следует воспользоваться тегом <link> с параметром media, значением которого выступают все те же типы, перечисленные в таблице 35.

```
  <link media="print, handheld" rel="stylesheet"
href="print.css" type="text/css">   <link
media="screen" rel="stylesheet" href="main.css"
type="text/css">   В данном примере используются
две таблицы связанных стилей, одна для отображения
в браузере, а вторая — для печати документа и его
просмотре на смартфоне. Хотя на страницу
загружаются одновременно два разных стиля,
применяются они только для определенных
устройств.
```

☑ Вопросы для самоконтроля

1. Что такое каскадная таблица стилей?
2. Какие способы добавления таблиц стилей на страницу существуют? Какой из них является предпочтительным?
3. Сформулируйте основные правила применения стилей.
4. Какие стили для web-страницы существуют, и каким образом определяется, какой из них будет применен?
5. Как можно повлиять на конечный стиль спорного элемента?
6. Раскройте суть технологии наследования.
7. Какие единицы измерения величин применимы для таблиц стилей? Какие из них являются предпочтительными?
8. Что означает запись «@import "1.css" print»?

§ 3.2. Стилевые свойства текстовых элементов

Управление цветом текста в CSS

Каскадные таблицы стилей в первую очередь описывают свойства текста. Это касается как текстовых блоков, так и строковых элементов содержания страницы. Здесь возможно управлять отображением цвета самого текста (color), цвета фона (background-color), на котором отображается текст, а также определять цвет границы текстового блока (border-color). Рассмотрим подробнее первое из свойств.

Для задания цветов используются его представление в шестнадцатеричном коде. Для идентификации системы счисления, перед шестнадцатеричным числом ставят символ решетки #, например #668899. Каждый из трех цветов — красный, зеленый и синий — может принимать значения от 00 до FF. Таким образом, обозначение цвета разбивается на три составляющие #rrggbb, где первые два символа отмечают красную компоненту цвета, два средних — зеленую, а два последних — синюю. Допускается использовать сокращенную форму вида #rgb, где каждый символ следует удваивать. Так, запись #689 следует расценивать как #668899.

Браузеры поддерживают некоторые цвета по их названию. В таблице ниже приведены названия, шестнадцатеричный код и описание[12].

Таблица 4. Задание цветов в CSS.

Имя	**Цвет**	**Код**	**Описание**
white		#ffffff или #fff	Белый
silver		#c0c0c0	Серый
gray		#808080	Темно-серый
black		#000000 или #000	Черный
maroon		#800000	Темно-красный
red		#ff0000 или #f00	Красный
orange		#ffa500	Оранжевый
yellow		#ffff00 или #ff0	Желтый
olive		#808000	Оливковый
lime		#00ff00 или #0f0	Светло-зеленый
green		#008000	Зеленый
aqua		#00ffff или #0ff	Голубой
blue		#0000ff или #00f	Синий
navy		#000080	Темно-синий
teal		#008080	Сине-зеленый
fuchsia		#ff00ff или #f0f	Розовый
purple		#800080	Фиолетовый

Те же самые цвета можно задавать и в десятичном представлении, в так называемой «раскладке RGB». Каждая из составляющих задается числом от 0 до 255, хотя возможно задание и в процентном соотношении. После указания цветовой схемы rgb в скобках, через запятую

указываются составляющие цвета, например rgb(0, 100, 100) или rgb(100%, 20%, 20%).

В HTML-документе для задания цвета текста использовался тег FONT, в CSS за это отвечает свойство color, его можно применять для любых текстовых элементов страницы.

Например, как в примерах выше:

```
p {color:green;}
```

При определении цвета текста для блочных элементов весь текст этого элемента будет отображаться заданным цветом. Если же применить строчный элемент страницы внутри блочного, то возможно и частичное изменение цвета. Так, например, внутри блочного параграфа может быть расположен строчный тег курсива:

```
p {color:green;}  i
{color:#00ff66;}
```

В качестве блочного элемента страницы может выступать и сама страница.

```
body {color:green;}
```

Стили шрифта

Шрифтам в компьютерной графике всегда уделялось много внимания, интернет в этом вопросе не является исключением. Все богатство и разнообразие шрифтов, существующих в природе, для русского языка ограничено тремя группами, так называемыми гарнитурами:

- serif - пропорциональный шрифт с засечками (например, Times)
- sans-serif - пропорциональный шрифт без засечек
  (например, Arial или Verdana)

- monospace – моноширинный, тот, что использовался на печатной машинке (например, Courier).

Дизайнер web-страниц может применить следующие параметры шрифта, влияющие на начертание букв текста:

- font-family - семейство начертаний шрифта (гарнитура);
- font-size - размер шрифта (кегль);
- font-style - прямое начертание или курсив;
- font-weight - насыщенность шрифта, (наличие «жирности» в его написании;
- font-variant - вариант начертания (обычный или мелкими буквами – «капителью»).

font-family обычно задается в виде списка нескольких похожих шрифтов отделяемых запятыми в порядке предпочтения на тот случай, если на компьютере пользователя нет нужного шрифта. Если название шрифта состоит из нескольких слов, его заключают в кавычки[8,12]. В конце списка обычно указывают имя гарнитуры, к которой относится шрифт, на случай, если ни один из перечисленных шрифтов не был найден. Порой, именно только гарнитуру и указывают, предлагая браузеру самому подобрать шрифт из своей коллекции.

```
p {font-family:‖Times New Roman‖, serif}  p
{font-family: sans-serif}
```

При оформлении статей на сайтах возникает необходимость что-то выделить «крупным» шрифтом, а некоторый текст наоборот не должен акцентировать на себе внимание. В этом случае на помощь приходит свойство font-size, так называемый «кегль».

Размер шрифта можно задавать в абсолютных единицах длины, т.е. в типографских пунктах (pt, 0,35мм) или пикселях(px), а также в относительных с помощью служебных слов. Размер шрифта определяется через размер «контейнера» под букву, который больше самой буквы.

```
<p style="font-size:24pt;">Кегль параграфа
установлен в 24 пункта</p>
<p style="font-size:12px;">Кегль параграфа
установлен в 12 пикселей</p>
<p style="font-size:large;">Размер кегля
относительный</p>
```

Соответствие между размерами, принятыми в CSS и HTML приведено в таблице.

Таблица 5. Размер шрифта в CSS и HTML

CSS	xxsmall	xsmall	small	medium	large	x-large	xx-large	
HTML	1		2	3	4	5	6	7

Возможно также и процентное соотношение в задании шрифта, тогда он будет вычисляться относительно охватывающего его элемента

```
<p style="font-size:200%;">Кегль параграфа
установлен в 2 раза больше размера букв
охватывающего параграф элемента</p>
```

Практически у каждого шрифта есть несколько написаний: курсив, полужирный, полужирный курсив. Каждое из этих начертаний определяется в CSS параметрами стиля: font-style и font-weight.

Свойство стиля font-style позволяет задать курсив (italic), по умолчанию прямое начертание (normal). Если хочется усилить насыщенность текста, свойству font-weight, указывают одно из значений:

- normal,
- bold (жирный),
- bolder (жирнее),
- lighter (светлее)
- одно из девяти целых чисел от 100 до 900. (400 – норма).

Если хочется жирный курсив, то необходимо указать оба этих параметра.

```
<p style="color:green; font-style:italic;
fontweight:bold;">Жирный курсив</p>
```

В ситуации, когда вдруг возникает необходимость отобразить, например, все заголовки прописными буквами на помощь приходит свойство font-variant. Ну не перепечатывать же в каждой странице заголовки – ведь завтра же концепция сайта может вновь поменяться!

Итак, для замены строчных букв прописными свойству font-variant необходимо задать значение small-caps (значение по умолчанию – normal). Стоит заметить, что он делает их несколько меньшими по размеру, чем прописные буквы текущего шрифта. Это называют «капителью».

В заключении отметим, что можно задавать каждое из свойств шрифта по отдельности или же указывать все эти свойства в одном параметре font через пробел:

```
H1{font: bold 24pt small-caps sans-serif;}
```

Форматирование текстовых фрагментов

Во второй главе данного пособия для задания выравнивания текста по левому или правому краю, по ширине или же центру использовался параметр тегов align. Для переноса свойств по выравниванию блоков текста в таблицу стилей используется свойство text-align с абсолютно аналогичными значениями:

```
<p style="text-align:center; color:green;">Этот
параграф выровнен по центру </p>
```

Еще раз обращаем внимание на написание значение параметра, выполняющего выравнивание по центру «center», а не «centre».

Итак, теперь мы можем работать с web-страницами почти как любыми электронными документами: определить шрифт, размер шрифта, выравнивание по ширине. Обычно при оформлении документов просят еще полуторный межстрочный интервал - но и здесь web-страница

ничем не будет отличаться - за задание этого свойства отвечает line-height.

line-height позволяет определить расстояние между базовыми линиями строк: например, если в две строки напечатать символы «ж» и провести линии через точки пересечения «черточек» при написании букв или по верхней границе – это и будет указанное расстояние (Рисунок 1). Также как и размер шрифта может задаваться аналогичными абсолютными и относительными значениями.

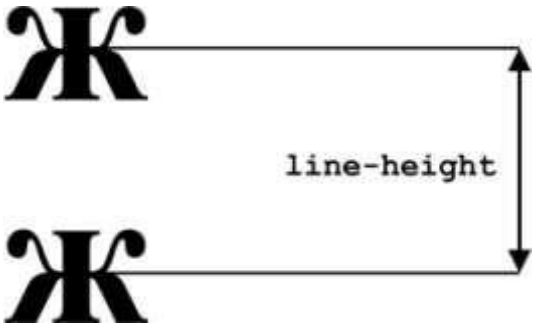

Рисунок 1. Расстояние между базовыми линиями строк.

```
<p style="line-height:12pt;font-size:12pt; "> Этот
параграф мы набрали кеглем 12 pt. Интервал
одинарный</p>
<p style="line-height:24pt;font-size:12pt; "> Этот
параграф мы набрали кеглем 12 pt. Интервал
двойной</p>
```

Благодаря данному свойству можно добавлять на страницы «встроенные» в строки изображения. Например, картинка имеет размеры 20x20 и выровнена по верхнему краю строки, и раз ее размер больше размера кегля межстрочное расстояние увеличено браузером автоматически.

```
<p style="font-size:20px;">
В эту строку мы встраиваем картинку - <img
src=‖inline.gif‖ border=0 width=20 height=20
align=top> в центре строки. </p>
```

Таким образом, можно точно позиционировать текст и графику в строке.

Помимо межстрочного расстояния можно задать и межбуквенное расстояние в одной строке – получая то разреженный, то уплотненный шрифт. Изначально расстояние между буквами автоматически регулируется размером шрифта и, очевидно, что чем больше размер шрифта, тем больше расстояние между буквами. Оно представляет собой расстояние между «контейнерами», содержащими буквы, может быть как фиксированным (как у моноширинных шрифтов наподобие печатающей машинки), или изменяться в зависимости от конкретной буквы.

Для изменения стандартного межбуквенного расстояния применяют свойство letter-spacing:

```
<p style="font-
family:monospace;letterspacing:10pt;"> Межбуквенное
расстояние 10pt/p>
```

Выше уже говорилось о том, что иногда возникает необходимость автоматического изменения регистра при написании того или иного текста. И если возможностей свойства font-variant недостаточно на помощь приходит параметр text-transform, задание значений которого позволяет получить различные результаты:

- uppercase – все прописные,
- lowercase – все строчные,
- capitalize – каждое слово с прописной буквы, none – снимает унаследованные установки.

```
H1{font: bold 24pt; text-transform: uppercase; }
```

Еще один вид преобразования шрифта - это подчеркивание, перечеркивание или надчеркивание слов, главное чтобы данные значения присутствовали в файле со шрифтом. Выполняется такое преобразование путем применения параметра text-decoration:

- line-through - зачеркивание
- underline – линия под текстом

- overline – линия над текстом
- none – нет линии

```
<p style="text-decoration:line-through;">
Перечеркнем это предложение.</p>
```

Как можно догадаться, наиболее применяемым значением данного свойства является none, используемое для гиперссылок – тогда они перестают подчеркиваться.

С помощью свойств каскадной таблицы стилей возможно решить еще одну проблему HTML-разметки – отступ первой строки абзаца. Да, можно конечно использовать неразрывные пробелы, но это очень неудобно, особенно при работе с большими объемами текста. Для задания «красной» строки достаточно воспользоваться свойством textindent. Причем, для данного параметра можно задавать как положительные, так и отрицательные значения.

```
<p style="text-indent:20pt;">Этот параграф с горизонтальным отступом в двадцать типографских пунктов от левого края параграфа. </p>
<p style="text-indent:-10pt;">А в этом параграфе - отрицательный горизонтальный отступ в первой строке параграфа. </p>
```

Форматирование списков

Задание маркеров или цифр для всех списков на сайте также можно перенести в таблицу стилей, лишь единожды задав соответствующие свойства для тегов <ul> и <ol>.

Например, для того чтобы совсем избавиться о маркера при отображении списка достаточно определить свойство display как none.

```
<ul style="display:none;">
<li>Первый элемент списка
</ul>
```

Но чаще всего, разработчики сайта планируют все же видеть рядом с элементом списка точку или квадратик. Свойство list-style-type позволяет задать следующие типы маркеров:

- square – квадрат,
- disk – диск,
- circle – круг

При задании этого же параметра для <ol> возможно получить:

- lower-roman - строчные римские буквы
- lower-roman - строчные римские буквы
- upper-alpha- заглавные латинские буквы
- lower-alpha- строчные латинские буквы

При задании list-style-type равным none, также можно избавиться от маркеров.

Если же стандартные формы не устраивают дизайнера, то при задании list-style-image URL-адреса с графическим файлом, содержащим необходимый маркер можно получить и «галочки» и «палочки».

Например,

```
<ul style="list-style-image: url(_barrow.gif‘);">
<li>Элемент списка
</ul>
```

☑ Вопросы для самоконтроля

1. Какие способы задания цвета элементов существуют?
2. Назовите группы шрифтов.
3. Какие свойства отвечают за межстрочное расстояние? За межбуквенное?
4. Как задать маркированный список с маркерами-картинками?

☑ Лабораторная работа № 14

Используя полученные знания, примените к созданному в лабораторной №11 проекту

☑ *Задание. Вариант 1.*

- единый стиль оформления заголовков с разреженным межбуквенным расстоянием и отступом от основного текста,
- оформите оглавление в виде маркированного списка в виде картинок, оформите каждый параграф с отступом красной строки,
- выделите цветом и размером все встреченные в параграфах жирные начертания
- создайте два класса ссылок: для оглавления и межстраничные.
- Предусмотрите отдельную упрощенную таблицу стилей для печати.

☑ *Задание. Вариант 2.*

- стиль оформления заголовков с полуторным межстрочным расстоянием, прописными буквами, шрифтом без засечек и 24 пункта,
- меню в виде нумерованного римскими цифрами списка,
- все абзацы с отступом красной строки и выровнены по ширине,
- выделите цветом и размером все встреченные в параграфах курсивные начертания,
- создайте два класса ссылок: для оглавления и межстраничные.

Оформление страниц задать в виде подключаемых каскадных таблиц стилей.

§ 3.3 Стилевые свойства блочных элементов. Слои

Блочные элементы (блоки текста), к которым относят абзац <p>, текстовые блоки и , таблицы <table> позволяют оперировать с текстом в терминах прямоугольников, которые этот текст занимает. При этом блок текста становится элементом дизайна страницы с теми же свойствами, что и картинка, таблица или прямоугольная область приложения.

Блок текста обладает свойствами: ширины и высоты, границы, отступов, фоновой заливки или фонового изображения, произвольного размещения и управления обтеканием.

Слои

Одним из ярких представителей блочных элементов является слой. Слой — это элемент web-страницы, созданный с помощью тега <DIV>, к которому применяется стилевое оформление. Слои давно уже стали привычной технологией для пользователей профессиональных графических и издательских пакетов. Если бы Adobe Photoshop не работал бы со слоями, он стоял бы на одной ступеньке со «старым уродливым карликом Paint» из состава Windows.

Первоначально слои на web-страницах были введены компанией Netscape в виде тега <LAYER>. Этот тег позволял прятать/показывать текущее содержимое, устанавливать положение относительно окна браузера, накладывать один слой поверх других и загружать данные в содержимое слоя из файла. Что характерно, все эти параметры легко менялись с помощью JavaScript и это расширяло возможности по созданию действительно динамического контента на странице. Несмотря на столь впечатляющий набор возможностей, тег <LAYER> не был включен в спецификацию HTML. Однако необходимость в указанных возможностях, которые бы применялись на сайтах, уже назрела, и в конце 1996 года синтаксис для работы со слоями был разработан и одобрен в рабочем проекте консорциума «CSS Positioning (CSS-P)». Основная нагрузка ложилась на стили, с их помощью можно управлять видом любого элемента, в том числе менять значения динамически через JavaScript[12].

Итак, сейчас слой представляют собой прямоугольный блок, который может содержать различные объекты - текст, графику, таблицы. Допускается применение нескольких слоев в пределах одного документа, которые могут взаимно перекрываться, с помощью чего можно добиваться различных интересных эффектов. Кроме того, содержимое слоя в любой момент можно отредактировать или вообще удалить, не трогая элементы других слоев. Это гораздо удобнее, чем заново создавать картинку (что приходилось бы делать при невозможности использования слоев).

Благодаря этому тегу HTML-код распадается на ряд четких наглядных блоков, за счет чего верстка слоями называется также блочной версткой. Код при этом получается более компактным, чем при табличной верстке, к тому же поисковые системы его лучше индексируют.

Применение слоев позволяет также создать на странице некоторое подобие трехмерного пространства, хотя на самом деле это лишь плоские слои с заданным порядком перекрытия, как в колоде карт. При помощи таблиц стилей CSS возможно задание различных свойств слоям и позиционировать их. Это открывает перед web-мастером массу возможностей, позволяет создавать страницы более похожими на типографские и, в большинстве случаев, отказаться от нежелательного использования таблиц для позиционирования содержимого.

Итак, для добавления слоя на web-страницу необходимо использовать тег . Внутри любого слоя может содержаться множество элементов. Все они записываются между открывающим и закрывающим тегом слоя .

```
<div >
Содержимое слоя
</div>
```

Как и ко всем остальным элементам, к слою применим ряд свойств, задаваемых с помощью таблицы стилей. Так как слой является универсальным блочным элементом, рассмотрение стилевых свойств будем проводить на примере работы со слоями.

Задание размеров

Изначально слой принимается размеры самого широкого из элементов, которых он содержит. В случае если необходимо жестко определить высоту или ширину слоя, используются соответственно свойства height и width. Эти значения можно задавать в пикселях, числовыми значениями, а также в процентах. Если указанное значение меньше, чем размеры картинок, таблиц и прочих неразрывных элементов, слой определится по самому широкому элементу. Значение по умолчанию auto делает слой «резиновым».

```
div {height: 100px; width: 200px;}
```

Управление видимостью и прозрачностью

Как уже было упомянуто выше, слои можно использовать для всплывающих подсказок и выпадающих меню. Основной особенностью данных объектов web-страницы является возможность в некоторые моменты времени «появляться» и «прятаться», то есть делать слои видимыми и невидимыми. За данное свойство слоев отвечает параметр visibility, принимающий следующие значения:

- visible - элемент виден на странице.
- hidden – элемент не виден на странице, спрятан.
- inherit – видимость определяется свойством родителя.

Еще одно свойство по управлению видимостью display, визуально отличающееся от visibility тем, что сдвигает предыдущий и последующий слои вместе, принимает следующие возможные значения:

- inline - содержимое начинается с того места, где окончился предыдущий элемент.
- list-item - элемент выводится как блочный и добавляется маркер списка.
- none - Временно удаляет элемент из документа. Занимаемое им место не резервируется и web-страница формируется так, словно элемента и не было.

- block - происходит перенос строк в начале и в конце содержимого

Уже оговаривалось, что если заданные размеры слоя не вмещают содержащиеся в нем элементы, он автоматически растягивается под самый «громоздкий» из них. Свойство overflow позволяет изменить данное поведение.

- hidden - отображается только область внутри слоя, остальное будет скрыто.
- auto - Полосы прокрутки добавляются только при необходимости
- scroll - Всегда добавляются полосы прокрутки.

Позиционирование в пространстве

Для любого слоя возможно задать его положение в пространстве web-страницы. Положение по горизонтали определяется свойством left, отвечающим за расположение верхнего левого угла слоя относительно выбранной точки отсчета. Аналогично положение слоя по вертикали определяет свойство top, также задаваемое для верхнего левого угла слоя. Напоминаем, что ордината в компьютерной технике в отличие от Декартовой системы координат увеличивается при перемещении вниз.

Обе координаты можно задавать в пикселях, процентах, числах или устанавливая значение auto для определения автоматически относительно других элементов.

```
div {left: 50px;top: auto;}
```

Помимо размещения в плоскости web-страницы элементы могут накладываться друг на друга в определенном порядке, образуя тем самым третье измерение, перпендикулярное экрану. Каждый элемент может находиться как ниже, так и выше других элементов на странице, определенный свойством z-index.

Слой с самым большим значением свойства z-index будет «накладываться» поверх всех остальных. В качестве значения этого свойства допустимо использовать целое число, в том числе и отрицательное. При равном значении z-index или при указании значения `auto`, на переднем плане окажется тот элемент, который в коде HTML описан ниже[12].

Например,

```
   <style type="text/css">
    #layer1, #layer2, #layer3, #layer4 {
 position: relative; /* Относительное
позиционирование */
    }
    #layer1, #layer3 {
 font-size: 50px;  color:
 #000080;
    }
    #layer2, #layer4 {
 top: -55px; left: 5px;
 color: #ffa500;     font-
 size:70px;
    }
    #layer1 { z-index: 2; }
    #layer2 { z-index: 1; }
    #layer3 { z-index: 3; }
    #layer4 { z-index: 4; }
   </style>

 <body>
   <p>Слой 1 наверху</p>
   <div id="layer1">ИМИКТ</div>
   <div id="layer2">ИМИКТ</div>
   <p>Слой 4 наверху</p>
   <div id="layer3">САФУ</div>
   <div id="layer4">САФУ</div>
  </body>
```

В браузере обретет вид (Рисунок 2):

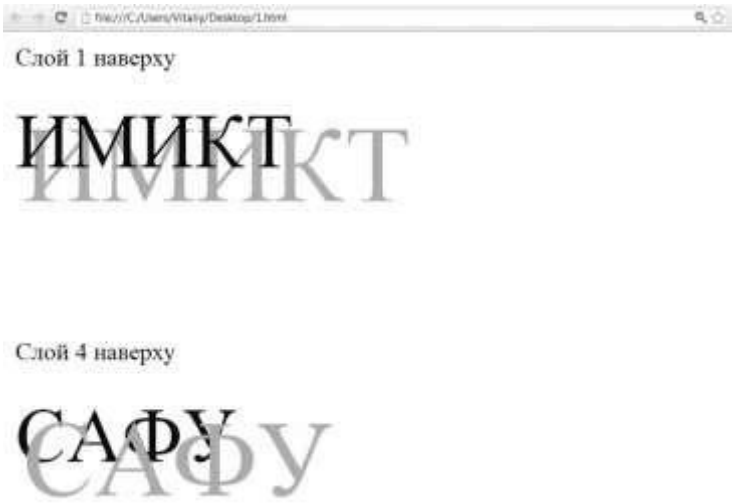

Рисунок 2. Позиционирование слоев.

В примере выше указан незнакомое еще свойство Position, которое определяет способ позиционирования элементов, в данном случае слоев. Описывая правила задания координат слоев, было указано, что они рассчитываются относительно точки отчета, в качестве которой может выступать левый верхний угол страницы, или другие окружающие слой элементы.

Значение position:absolute («абсолютное позиционирование») определяет координаты элементов относительно левого верхнего угла страницы; position:relative - относительно предыдущего (в написании кода страницы) элемента («относительное позиционирование»). position:static же означает, что позиционирование слоя будет происходить относительно фона.

Задание свойств left и top слоя относительно левого верхнего угла окна браузера возможно только при абсолютном позиционировании. Слой, заданный с абсолютным позиционированием, может располагаться под основным текстом или, наоборот, поверх него благодаря свойству z-index, что позволяет выводить на странице различные подсказки, всплывающие окна, рекламу или плавающие меню[4]. Если точно известны координаты каждого слоя относительно угла браузера, применение теории абсолютного позиционирования не составляет большого труда. Самым ярким примером данной теории является создание верхнего меню страницы, которое всегда «прибито» к верхнему

краю страниц. Если же невозможно предугадать координаты, которые получит слой при размещении различных элементов и добавлении контента, на помощь приходит относительное позиционирование, где нет необходимости задумываться над значениями координат слоев.

Интересных эффектов можно достичь при комбинации относительного и абсолютного позиционирования. Рассмотрим следующий пример. Возьмем два слоя и для первого из них, который будет располагаться на заднем плане, указываем абсолютное позиционирование. Второй же слой, расположенный поверх первого, должен иметь относительное позиционирование, тогда его координаты будут вычисляться относительно первого (Рисунок 3). Положение в пространстве определим с помощью z-index.

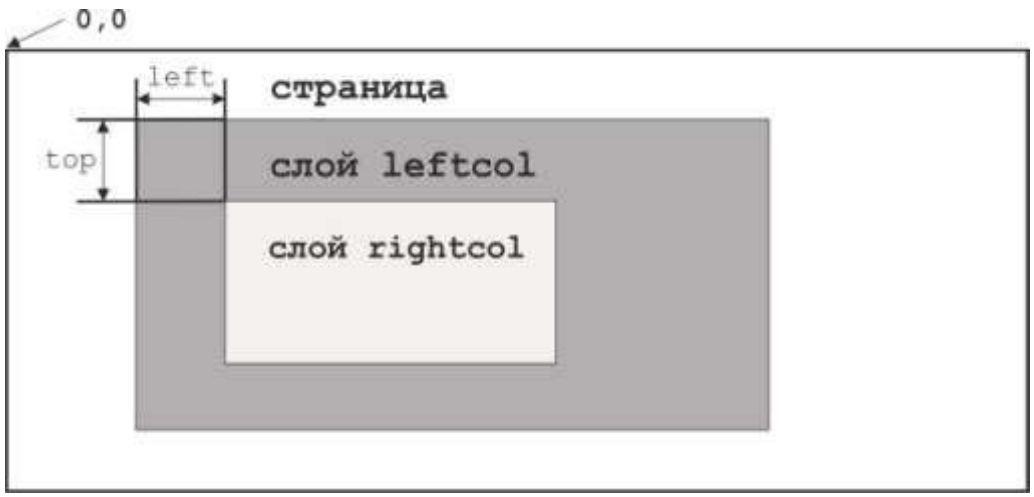

Рисунок 3. Пример работы со слоями.

Для задания данных слоев необходимо написать следующий код:

```
<style type="text/css">
#leftcol {
 position: absolute; /* Абсолютное позиционирование */
width: 550px; /* Ширина левой колонки */  background:
#e0e0e0; /* Цвет фона содержимого */
}
#rightcol { /* Этот слой накладывается поверх */
position: relative; /* Относительное позиционирование
*/  left: 500px; /* Положение от левого края
*/  top: 20px; /* Положение от верхнего
края */  width: 200px; /* Ширина правой
```

```
 колонки */  background: #800000; /* Цвет
 фона */  color: #fff; /* Цвет текста */
 }
 </style>
 …
 <body>

 <div id="leftcol">Левая колонка</div>
 <div id="rightcol">Правая колонка</div>

 </body>
```

Фоновое оформление

Практически для каждого элемента web-страницы, да и нее самой, можно задать фоновое оформление. В HTML-тегах за цвет фоновой заливки отвечал bgcolor, в таблицах стилей данное свойство определяется с помощью background-color. Изначально значение данного свойства задается как transparent – прозрачный. Цвет фона можно задавать аналогично цвету текста: по его названию, по шестнадцатеричному значению и с помощью RGB.

```
 BODY {     background-color: #3344CC; /* Цвет фона
 web-
страницы */
 }
   H1 {     background-color: RGB(249, 201, 16); /*
 Цвет
фона под заголовком */
    }
```

Кроме цвета фона и его прозрачности можно управлять фоновой картинкой, в том числе недоступными для HTML координатами ее размещения и способами повторения.

За задание фоновой картинки отвечает свойство background-image. В качестве значения используется путь к графическому файлу, который указывается внутри конструкции url(). Путь к файлу при этом можно

писать как в кавычках (двойных или одинарных), так и без них. Когда фоновое изображение не требуется, аргумент может принимать значение none.

Аналогично тегам если одновременно для элемента задан цвет фона и фоновая картинка, то пока фоновая картинка не загрузится полностью, отображается цвет фона. В случае наличия в рисунке прозрачных областей, через них будет проглядывать фоновый цвет.

```
BODY {
  background-image: url(bg.jpg);
 }
```

Свойство background-attachment устанавливает, будет ли прокручиваться фоновое изображение вместе с содержимым элемента. Изображение может быть зафиксировано и оставаться неподвижным (fixed), либо перемещаться совместно с документом (scroll), что принято по умолчанию. Аналогом данного свойства является параметр bgproperties в параметрах тега body.

```
   BODY {
    background-image: url('sample.gif');
background-attachment: fixed;
   }
```

Недоступное для HTML-разметки точное расположение фонового изображения на странице и его повторение по горизонтали может эффективно управляться с помощью таблицы стилей.

Так свойство background-position позволяет выровнять изображение по горизонтали, вертикали или задать положение относительно левого верхнего угла браузера или другого элемента.

Для задания выравнивания по горизонтали используются значения left, center или right, а для выравнивания по вертикали - top, center, bottom. Порядок записи top left или left top при этом не важен, главное указать значения через пробел. Если же указана только одна из характеристик – вторая принимается равной значению center.

При задании положения фонового изображения относительно левого верхнего угла точки отсчета можно использовать пиксели или проценты, указывая сначала абсциссу, а потом ординату левой верхней точки изображения также через пробел. Соотношение между используемыми ключевыми словами и процентной записью приведено в таблице 11.

Таблица 6. Соотношение между используемыми ключевыми словами и процентной записью.

Вариант № 1	**Вариант № 2**	**Сокращенный вариант**	**Процентная запись**	**Положение**
top left	left top		0% 0%	в левом верхнем углу
top center	center top	top	50% 0%	по центру вверху
right top	top right		100% 0%	в правом верхнем углу
left center	center left	left	0% 50%	по левому краю и по центру
Вариант № 1	**Вариант № 2**	**Сокращенный вариант**	**Процентная запись**	**Положение**
center center	center center	center	50% 50%	по центру
right center	center right	right	100% 50%	по правому краю и по центру
bottom left	left bottom		0% 100%	в левом нижнем углу
bottom center	center bottom	bottom	50% 100%	по центру внизу
bottom right	right bottom		100% 100%	в правом нижнем углу

За повторение фонового изображения по горизонтали и вертикали отвечает свойство background-repeat. С помощью него можно заставить

браузер повторять картинку только по одной из осей или не повторять вовсе. Возможными значениями данного свойства являются:

- no-repeat - без повторений
- repeat-x - повторение только по горизонтали
- repeat-y - повторение только по вертикали
- repeat – повторение по обеим осям (значение по умолчанию)

В заключении заметим, что также как и при работе с шрифтами свойство background объединяет все рассмотренные выше. Значения указываются в любом порядке и количестве через пробел.

```
table {
 background: #fc0 url(nd.png) repeat-y; }
```

Внутренние и внешние отступы

Делая записи в школьной тетради, мы оставляли слева или справа четыре свободные клетки – поля. Работая с документами, как бумажными, так и электронными мы также оставляем припуски для переплета и удобства чтения. Газетные полосы также отделены друг от друга пространством, позволяющим не затеряться во множестве символов. Присутствуют поля и на web-страницах, причем их можно задавать относительно границы страницы и относительно самого блока текста. В первом случае мы имеем дело с внутренним отступом, а во втором - с внешним, а их сумма определяет ширину поля.

Итак, внутренний отступ текстовых блоков задается свойством padding и отсчитывается от внешней границы блока внутрь, в то время как внешний отступ margin - от внешней границы блока наружу. Внутренний отступ позволяет начинать текст не от самой границы, а оставлять некоторое свободное пространство, а внешний – добавлять пространства между блоками. По своему действию данные отступы очень похожи на параметры CELLPADDING и CELLSPACING тега <TABLE>, где ячейки таблицы выступают в виде блоков текста. Для них указанные свойства также применимы.

Можно задавать свой отступ для каждой из сторон слоя, или использовать общий для всех сторон: margin-left, margin-right, margintop, margin-bottom – задают внешние отступы для каждой из сторон, а padding-left, padding-right, padding-top, padding-bottom – соответствующие расстояния от края слоя до его содержания. В случае равноудаленности блока текста от всех границ используют margin и padding.

```
p {margin:5px; padding-left:100px;} table
{margin:0px; padding:10px;}
```

На рисунке ниже (Рисунок 4) наглядно представлены некоторые из отступов.

Рисунок 4. Внутренние и внешние отступы.

Возможно применение общих свойств padding и margin и в случае неравномерных отступов по четырем сторонам - достаточно указать последовательно значения левого, правого, верхнего и нижнего отступа через пробел.

```
p {margin:5px 0px 0px 0px;}
```

Границы элементов

У каждого блочного элемента есть границы, зрительно определяющие размеры блока. От границы отсчитываются отступы, вдоль границы происходит обтекание блока текстом.

Для описания их свойств применяют border-width, задающий толщину границы, border-color, определяющий ее цвет, и border-style, позволяющий задать различные стили границы.

Каждое из свойств может быть задано универсально для всех четырех сторон границ, или же уникальным для каждой из них. Так, например, border-top-width – толщина верхней границы блока, а borderright-color – цвет правой границы. Правила задания цвета для bordercolor абсолютно аналогичны значениям color, в то время как borderstyle может принимать следующие значения:

- none – отсутствуют
- dotted – пунктирная линия
- dashed – точки и тире
- solid – сплошная линия
- double – двойная линия
- inset – «вдавленная»
- outset – «выпуклая».

Для описания границы нет необходимости указывать в стиле все свойства, достаточно их перечислить через пробел для общего свойства border или, например, border-right.

```
p {border-top:1px dotted red;}
```

Обтекание блока текстом

Под обтеканием блока (слоя) текстом понимают тот же самый эффект, который можно реализовать для изображения, когда текст окружает изображение с двух сторон. «Обтекание» картинки текстом от обычного встраивания картинки в текст документа отличается тем, что вдоль вертикальной границы картинки располагается несколько строк текста, а не одна. Получаем такой «плавающий» блок.

Данным свойством блоков текста управляют два параметра CSS: float и clear, один отвечает за плавающий блок, второй – за обтекающий текст.

Свойство float определяет положение плавающего блок текста. Он может принимать значения:

- left - блок выровнен по правому краю, текст обтекает справа,
- rigth – блок выровнен по правому краю, текст обтекает слева

Второе свойство описания стилей clear позволяет управлять собственно обтеканием. Собственно, оно запрещает наличие плавающих блоков около текста.

- right – текст нельзя использовать для обтекания справа,
- left – текст нельзя использовать для обтекания слева,
- none - текст можно использовать для обтекания с обеих сторон,
- both - текст нельзя использовать для обтекания с двух сторон.

Примеры задания слоев.

Пример №1. Центрирование слоев по горизонтали

Если необходимо создать слой, располагающийся строго по центру, следует действовать следующим образом. Вначале указать ширину и высоту слоя с помощью параметров width и height. Размеры можно задавать в пикселях, процентах или других единицах. Ширину, например, можно определить в процентах, а высоту в пикселях. Из-за этой особенности предлагаемый метод размещения по центру является наиболее универсальным.

Следующий шаг — задаем абсолютное позиционирование слоя через аргумент position: absolute. Положение слоя следует определить как 50% по горизонтали и вертикали с помощью свойств left и top. Эти значения остаются неизменными, независимо от используемых единиц измерения.

Так как координаты слоя определяются от его левого верхнего угла, для точного выравнивания следует добавить параметры margin-left и

margin-top с отрицательными значениями. Их величина должна быть равна половине ширины слоя (для margin-left) и высоты (для margintop). Чтобы высота слоя не менялась из-за его контента, включен параметр overflow: auto, он добавляет полосы прокрутки, если в них возникнет нужда, высота при этом остается неизменной[10,12].

```
#centerLayer {  position:
absolute;   width: 400px;
height: 300px;   left: 50%;
top: 50%;
 margin-left: -200px; margin-top: -150px;
background: #fc0; padding: 10px;  overflow: auto;  }
```

Пример №2. Наложение прозрачного слоя, эффект тени

Для создания тени традиционным способом делают дубликат данного слоя и смещают его на несколько пикселей по горизонтали и вертикали, изменяя цвет на серый, получая, таким образом, тень объекта (Рисунок 5).

```
 <style type="text/css">
 body {font-family: Verdana; font-size: 30pt; font-
weight: bold;}
 .d1 { position: absolute; color: silver; top: 20%;
left: 20%; }
 .d2 { position: absolute; color:green; top: 19.6%;
left: 19.6%; }
 </style>
 </head>
 <body>
 <div class="d1"> ИМИКТ</div>
 <div class="d2"> ИМИКТ</div>
 </body>
```

Рисунок 5. Наложение прозрачного слоя, эффект тени.

Пример №3. Позиционирование без таблицы.

Используя абсолютное позиционирование слоев можно добиться эффекта размещения слоев в таблице, просто задавая их координаты в соответствии с воображаемой сеткой таблицы (Рисунок 6).

Рисунок 6. Позиционирование без таблицы.

Для этого вполне достаточно написать код.

```
<style type="text/css"> div
{ position: absolute;}  .d1
{ top: 5%; left: 5%; }
.d2 { top: 5%; left: 55%; }
.d3 { top: 55%; left: 5%; }
.d4 { top: 55%; left: 55%; }
</style>
</head>
<body>
<div class="d1">Ячейка1</div>
```

```
<div class="d2">Ячейка2</div>
<div class="d3">Ячейка3</div>
<div class="d4">Ячейка4</div>
</body>
```

Пример №4. Декоративное оформление страницы.

Интересного эффекта можно достичь, располагая на заднем плане полупрозрачные слои и неброскими элементами. Таким образом, можно реализовать динамического создание фона (Рисунок 7).

```
<style type="text/css">
body {font-family: Verdana; font-size: 100%;
fontweight: bold;}
.d1 { position: absolute; font-size: 1000%;  color:
#ffccff;
top: 10%;  left: 10%; width: 70%;}
.d2 { position: absolute; font-size: 100%;  color:
navy;
top: 10%;  left: 25%; width: 35%;}
</style>
</head>
<body>
<div class="d1">ИМИКТ</div>
<div class="d2">
<p> 28 июня 2012 года решением ученого совета САФУ
институт математики и компьютерных наук и институт
информационных и космических технологий вошли в
состав института математики, информационных и
космических технологий (ИМИКТ). ИМИКТ начал свою
работу с 1 сентября 2012 года.
</p>
</div>
</body>
```

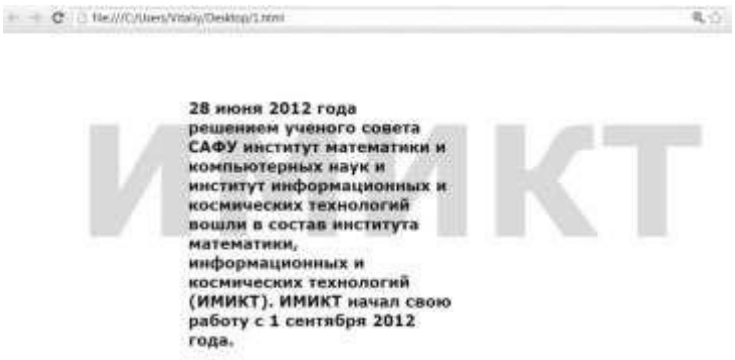

Рисунок 7. Декоративное оформление страницы.

☑ Вопросы для самоконтроля

1. Какие свойства можно задавать для блочных элементов?
2. Для чего нужны слои?
3. Чем отличается слой от других элементов страницы? Что общего у него с другими элементами?
4. Что общего у блочных элементов с изображениями?
5. Какие способы размещения слоев существуют? В чем их особенности?

§ 3.4. Дополнительные возможности CSS

В предыдущем параграфе мы рассмотрели основные стилевые свойства, которые позволяют управлять внешним видом элементов. Однако существует ряд дополнительных возможностей для более тонкой настройки оформления элементов.

Универсальный селектор

Для задания общего стиля для абсолютно всех элементов на странице можно воспользоваться универсальным селектором, задаваемым символом звездочки (*). С его помощью можно задать, например, задать шрифт или начертание текста.

```
<style type="text/css">
```

```
   * { font-family: Verdana, sans-serif;
fontsize: 12pt; }
  </style>
</head>
 <body>
<p>С любимыми не расставайтесь!<br> Всей
кровью прорастайте в них,-
И каждый раз навек прощайтесь!
Когда уходите на миг!</p>
</body>
```

Заметим, что иногда указывать универсальный селектор не обязательно, так *.class и .class дают одинаковый результат[12]. Кроме того, чаще всего универсальный селектор * можно безболезненно заметить на BODY.

Соседние и дочерние элементы

Элементы web-страницы, которые записаны в коде последовательно, называют соседними. Элементы, которые располагаются внутри других элементов – называют дочерними для них.

Так элементы <b> и
 в примере ниже являются соседними, а <b> дочерним для <p>. Но
 и <i> соседними не будут – их разделяет тег <b>.

```
<p>С любимыми не расставайтесь!<br>
<b>Всей кровью прорастайте в них,- </b>
И каждый раз навек прощайтесь!
Когда уходите <i>на миг</i>!</p>
```

Для управления стилем соседних элементов используется символ плюса (+), который устанавливается между двумя селекторами. Пробелы вокруг плюса не обязательны, стиль при такой записи применяется к Селектору 2, но только в том случае, если он является соседним для Селектора 1 и следует сразу после него[12].

```
Селектор 1 + Селектор 2 {Описание правил стиля }
```

Соседние теги, а значит и селекторы в таблицах стилей удобно применять в случаях, когда к тегам автоматически добавляются большие отступы: между абзацами и заголовком, между заголовками разного уровня, при добавлении списков всех видов и т.п. Соседние селекторы позволят задать размер данного отступа по своему усмотрению. Например, для того чтобы убрать большой отступ между абзацем и заголовков 1 уровня достаточно задать следующую стилевую конструкцию:

```
  <style type="text/css">
   H1 + P {margin-top: -10px; /* Смещаем 1й абзац
вверх к заголовку */   }
  </style>
 </head>
 <body>
  <h1>Заголовок </h1>
  <p>Абзац!</p>  <p>Абзац!</p>
 </body>
```

Рассмотрим еще один практичный пример. Часто возникает необходимость в текст статьи включать различные сноски и примечания. Обычно для этой цели создается новый стилевой класс, и он применяется к абзацу, таким способом можно легко изменить вид текста. Но решение данной задачи можно реализовать и с помощью соседних селекторов[13]. Для выделения замечаний создадим новый класс, назовем его prim, и станем применять его к тегу <h3>. Первый абзац после такого заголовка будет выделяться цветом фона и отступом.
Вид остальных абзацев останется неизменным.

```
<style type="text/css">
H1.prim {     font-
size: 80%; text-
align:right;
color: gray;
    margin-left: 30px;  margin-bottom: 0px;
```

```
}
   H1.prim + P {
   background: #ddd; width:400px;
margin-left: 30px;     padding:
7px;
  }
  </style>
 </head>
 <body>
  <h1> КАК МНЕ ТЕБЯ ЗАБЫТЬ? </h1>
  <h1 class=prim> Татьяна Снежина (Печѐнкина)</h1>
<p>российская поэтесса, композитор, автор и
исполнитель своих песен, лауреат премии «Песня года»
посмертно. Годы жизни: 1972 – 1995. Татьяна Снежина
– самая популярная поэтесса любовной лирики конца XX
века. Еѐ творчество получило признание уже после ее
гибели.<p>
<p>
Ах, как бы мне тебя забыть? <br>
Ведь вместе нам уже не быть. <br> Но
память – тонкая струна, <br>
Чуть тронешь – и она звучна. <br>
</p>
<h1> ПИСЬМА ТВОИ </h1>
<p>Когда от забот опускаются руки<br>
И больше нет смысла о чѐм-то мечтать, <br>
Проходит печаль от долгой разлуки, <br>
Когда я сажусь твои письма читать. <br>
</p>
```

Результат данного примера показан на рисунке ниже (Рисунок 8).

Рисунок 8. Изменение вида абзаца за счет использования соседних селекторов.

По своей логике дочерние селекторы похожи на селекторы контекстные. Разница между ними следующая. Стиль к дочернему селектору применяется только в том случае, когда он является прямым потомком, иными словами, непосредственно располагается внутри родительского элемента. Для контекстного селектора же допустим любой уровень вложенности. Заметим, что в большинстве случаев от добавления дочерних селекторов можно отказаться, заменив их контекстными селекторами. Однако использование дочерних селекторов расширяет возможности по управлению стилями элементов, что в итоге позволяет получить нужный результат, а также простой и наглядный код.

Для записи дочерних селекторов используют знак «больше».

```
Селектор 1 > Селектор 2 { Описание правил стиля }
```

Стиль применяется к Селектору 2, но только в том случае, если он является дочерним для Селектора 1.

Удобнее всего применять указанные селекторы для элементов, которые обладают иерархической структурой — сюда относятся, например, таблицы и разные списки. За счет вложения одного списка в другой получаем разновидность меню. Заголовки при этом располагаются горизонтально, а набор ссылок — вертикально под заголовками (Рисунок 9).

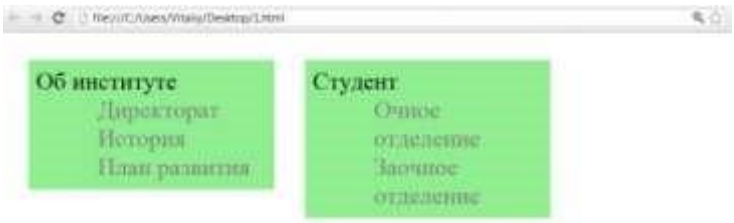

Рисунок 9. Список в виде меню.

Для размещения текста по горизонтали к селектору LI добавляется стилевое свойство float. Чтобы при этом разделить между собой стиль горизонтального и вертикального списка и применяются дочерние селекторы

```
  <style type="text/css">
   OL#menu {  margin: 0; padding: 0; }
OL#menu > LI {
    width: 150px; /* Ширина элемента в пикселях */
padding: 5px; /* Поля вокруг текста */  margin:10px;
 background-color: lightgreen; /*Задаем фон */
float: left; /*Располагаем элементы по горизонтали
*/    }    LI > OL {
    list-style: none; /*Убираем маркеры списка */
margin: 0; padding: 0
    padding-top: 5px; /*Добавляем отступ сверху */
color:gray;
   }
 </style>
 </head>
 <body>
  <ol id="menu">
   <li>Об институте
    <ol>
      <li> Директорат
      <li> История
      <li> План развития
     </ol>
   </li>
```

```
   <li>Студент
    <ol>
      <li> Очное отделение
      <li> Заочное отделение
    </ol>
   </li>
   </li>
  </ol>
 </body>
```

В данном примере дочерние селекторы требуются, чтобы разделить стиль элементов списка верхнего уровня и вложенные списки, которые выполняют разные задачи, поэтому стиль для них не должен пересекаться.

Форматирование по значениям параметров

Существует целая группа тегов, кардинально меняющихся в зависимости от параметров и их значений. Несомненным лидером является несомненно тег <input>, который может быть представлен, например, и в виде кнопки, и в виде текстового поля или даже радиокнопки. Во что превратиться данный элемент определяется, как мы помним, значением параметра type. В зависимости от значения которого, задаются остальные параметры: для текстового поля свои, а для радиокнопки совершенно другие. Как же в данном случае поступать со стилевыми свойствами?

Раз свойства таких элементов зависят от параметров и их значений, именно их и указывают рядом с селектором стиля.

Например, если параметр не имеет значений, но его присутствие должно изменять внешний вид элемента необходимо использовать следующую стилевую конструкцию:

```
Селектор[параметр] { Описание правил стиля }
```

Возможно также и обобщение для всех элементов, содержащих данный параметр.

```
[параметр] { Описание правил стиля }
```

Обращаем внимание, что пробел между именем селектора и квадратными скобками не допускается.

Например, для выделения отступами значения по умолчанию при работе с радиокнопками стилевая таблица будет выглядеть следующим образом:

```
<style type="text/css">
input[checked] {
margin:10px;
   }
  </style>
 </head>
 <body>
  <input type=radio name=r1>Зима <br>
<input type=radio name=r1>Весна <br>
<input type=radio checked name=r1> Лето<br>
<input type=radio name=r1> Осень <br>
 </body>
```

Если важен не только параметр, но и его значение, то стилевая конструкция принимает вид:

```
Селектор[параметр="значение"]{Описание стиля }
```

Или, обобщая для разных тегов,

```
[параметр="значение"] { Описание правил стиля }
```

Например, если необходимо все абзацы, выровненные по правому краю писать чуть меньше основного текста и серым начертанием достаточно указать

```
P[align="right"]{font-size:80%; color:gray;}
```

Часто данную возможность применяют для ссылок, загружаемых в новом окне браузера, задавая им уникальное оформление.

```
   A[target="_blank"] {background: url(blank.png)
0 6px no-repeat; padding-left: 15px;}
```

В данном примере рисунок к ссылке добавляется с помощью свойства background. В его функции входит создание повторяющейся фоновой картинки, но повторение фона можно отменить через значение no-repeat, что в итоге даст единственное изображение[12].

Еще более интересная и сложная ситуация возникает, если необходимое значение присутствует, но оно не единственно, а является одним из элементов списка значений, разделяемых пробелом. В таком случае поможет следующая стилевая конструкция:

```
Селектор[параметр~="значение"]{Описание стиля }
```

Или, опять обобщая для всех тегов

```
 [параметр~="значение"] { Описание правил стиля }
```

Например, стиль, применяемый для всех картинок, содержащихся в слоях с классом icon, может выглядеть таким образом

```
 <style type="text/css">
   [class~="icon"] img { width:10px; height:10px;}
  </style>
 </head>
 <body>
  <div class="menu icon">
   <img src="1.gif"> </div>
<div class="block icon">
   <img src="2.gif"> </div>
<div class="menu">
   <img src="3.gif"> </div>
```

```
</body>
```

В данном примере размер изображения 10x10 пикселей применяется к изображениям, если имя класса у слоя задано как icon. Отметим, что аналогичный результат можно получить, если использовать конструкцию *= вместо ~=.

Стиль, задаваемый по тексту, содержащемуся в значении параметра

Иногда сложно определить точное значение параметра, известно только, что он заканчивается определенной комбинацией символов (например, все изображения формата GIF заканчиваются на соответствующую комбинацию, которая содержится в параметре SRC тега <IMG>), начинается с определенной комбинации (например, все внешние гиперссылки начинаются с «http://»), или она встречается внутри значения параметра элемента (например, все изображения, размещенные в определенной папке содержат ее имя в параметре SRC). Во всех этих случаях также возможно контролировать параметры форматирования элементов.

Если значение параметра начинается с указанного текста, синтаксис применения будет следующий:

```
Селектор[параметр^="значение"] {Описание правил
стиля }
```

- для определенных тегов, и

```
[параметр^="значение"] {Описание правил стиля}
```

- в случае применения для всех элементов.

Тогда задача разграничения внешних и внутренних ссылок, обозначенная выше решается таким образом. Внешние ссылки характеризуются добавлением к адресу протокола HTTP, содержащегося в URL-адресе, задаваемом параметру HREF тега A.

```
   <style type="text/css">
 A[href^="http://"] {
     font-weight: bold /* Жирное начертание */   }
   </style> </head>
  <body>
     <p><a href="1.html">Обычная ссылка</a> |
  <a     href="http://www.narfu.ru"     target="_blank">
Внешняя   ссылка на сайт САФУ</a></p>
    </body>
```

В данном примере внешние ссылки выделяются жирным начертанием, также можно добавлять к ним небольшие иконки, как было задано в предыдущем пункте.

Для обозначения того, что значение параметра оканчивается указанным текстом, используется знак доллара «$», его также можно применять и к определенным селекторам и ко всему их разнообразию.

```
Селектор[параметр$="значение"]{Описание   стиля }
[параметр$="значение"] { Описание  стиля }
```

Решение задачи внешнего выделения всех gif-изображений будет решено следующим образом.

```
  <style type="text/css">
   IMG[src$=".gif"] {/*Если адрес изображения
заканчивается на .gif */ border: 3px solid;
}
  </style>
 </head>
 <body>
<img src="1.gif">
<img src="2.gif">
<img src="3.jpg" >
</body>
```

В данном примере содержатся две gif-картинки и одно jpg изображение. И только gif-картинки будут обрамлены рамкой (Рисунок 10).

Рисунок 10. Добавление рамки к gif-картинкам.

Возможен и такой вариант, когда стиль следует применить к тегу с параметром, где указанное сочетание присутствует в значении параметра, но неизвестно, в каком месте — в начале, середине или конце. В подобном случае следует использовать следующую стилевую конструкцию:

```
[параметр*="значение"] { Описание правил стиля }
Селектор[параметр*="значение"] {Описание правил
стиля }
```

Очень удобно таким образом все изображения, подключаемые из папки icon, делать сразу маленькими, например, 10x10 пикселей, не задумываясь каждый раз о размере картинки-иконки.

```
  <style type="text/css">
   IMG[src*="icon"] {/* Если изображение находится
в папке  icon */ width: 10px; height:10px;
}
  </style>
 </head>
```

```
 <body>
<img src="icon/1.gif">
<img src="2.gif">
<img src="icon/3.jpg" >
</body>
```

Только изображения 1.gif и 3.jpg будут уменьшены до размеров 10x10 пикселей, 2.gif сохранит свой реальный размер.

Частным случаем форматирования по содержащемуся в значении параметра тексту является наличие там символа дефиса «-», позволяющего употреблять его в именах классов и идентификаторов, изменяя их стиль. Данный стиль применяется к элементам, у которых параметр начинается с указанного значения или с фрагмента значения, после которого идет дефис.

```
[параметр|="значение"] { Описание стиля }
Селектор[параметр|="значение"] {Описание стиля }
```

```
  <style type="text/css">
DIV[class|="block"] {
    background: #306589; /* Цвет фона */
color: #acdb4c; /* Цвет текста */     padding:
5px; /* Поля */
   }
   DIV[class|="block"] A {
color: #fff; /* Цвет ссылок */
   }
  </style>
 </head>
 <body>
  <div class="block-menu-therm">
   <h2>Термины</h2>
   <div class="content">
     <ul class="menu">
      <li><a href="t1.html">Буквица</a></li>
      <li><a href="t2.html">Капитель</a></li>
      <li><a href="t3.html">Начертание</a></li>
     </ul>
   </div>
```

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

В данном примере имя класса задано как block-menu-therm, поэтому в стилях используется конструкция |="block", поскольку значение начинается именно с этого слова и в значении встречаются дефисы[12].

Все перечисленные выше методы применения стиля можно комбинировать между собой, в том числе определяя стиль для элементов, которые содержат два и более параметра. В подобных случаях квадратные скобки идут подряд[12].

```
[параметр1="значение1"][параметр2="значение2"]
{ Описание правил стиля }
Селектор[параметр1="значение1"][параметр2=
"значение 2"]
{ Описание правил стиля }
```

Псевдоклассы

Еще одним инструментом тонкой настройки стилей страниц является использование псевдоклассов. «Псевдоклассы определяют динамическое состояние элементов, которое изменяется со временем или с помощью действий пользователя, а также положение в дереве документа»[12]. Так, например, гиперссылка меняет свой цвет или подсвечивается та строка таблицы, при наведении на них курсора. С помощью псевдоклассов можно реализовать такие динамические эффекты.

Задание псевдокласса для селектора не представляет никакого труда – достаточно указать его имя после имени селектора и двоеточия. В качестве селектора могут выступать имена тегов, классов, идентификаторов, или при их опускании, – все элементы страницы.

```
Селектор:Псевдокласс { Описание правил стиля }
```

Принято все псевдоклассы делить на три группы:

- определяющие состояние элементов;
- имеющие отношение к дереву элементов;
- указывающие язык текста.

К первой группе относятся псевдоклассы, которые распознают текущее состояние элемента и применяют стиль только для этого состояния.

Псевдокласс active происходит при активации пользователем элемента, например, когда ссылка становится активной (в момент нажатия на нее). Псевдокласс link применяется по аналогии с одноименным параметром для непосещенных ссылок, а visited – для посещенных.

```
A:link {color: #055; }
A:visited {color:#333;}
A:active {color: #0f0;}
```

Очень популярным является псевдокласс hover , который активизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит. Именно с помощью него и организуют подсветку элементов. Например, таким образом, обеспечивают подсветку строк в таблице.

```
TR:hover { background:
#fc0;
/*Меняем цвет фона строки таблицы */
}
```

Еще одним способом повлиять на внешний вид текущего состояния элемента – псевдокласс focus, он активизируется при получении им фокуса. Псевдокласс применим для тех элементов, которые могут получить фокус: <a>, <input>, <select> и <textarea>.

Например, он может использоваться для подсветки текстовых полей формы, подсказывая, что сюда можно вводить информацию.

```
  <style type="text/css">
   INPUT:focus {color: red;}
  </style>
</head>
<body>
<input type="text" value="текст">
</body>
```

В данном примере в текстовом поле содержится текст, заданный в параметре value тега <input>. При щелчке по элементу формы происходит получение полем фокуса, и цвет текста меняется на красный. При щелчке на любом другом месте страницы – он восстановит свой первоначальный цвет.

Селекторы могут содержать более одного псевдокласса, которые перечисляются подряд через двоеточие, но только в том случае, если их действия не противоречат друг другу.

```
A:visited:hover {color: red;}
```

Ко второй группе псевдоклассов относят те, которые определяют положение элемента в дереве документа (родительский/дочерний элемент) и применяют к нему стиль в зависимости от его статуса.

Так first-child применяется к первому дочернему элементу селектора. То есть, если в пределах одного тега-контейнера встречается несколько одинаковых элементов, свойства применяться только к первому из них. Так в примере ниже цветом фона будет выделена только первая ячейка каждой строки таблицы.

```
<style type="text/css">
   td:first-child {background-color:gray; }
  </style>
</head>
 <body>
<table border=1 cellspacing=0 cellpadding=10>
<tr>
```

```
<td>Весна</td><td>март</td>
<td>апрель</td><td>май</td>
<tr>
<tr>
<td>Лето</td><td>июнь</td>
<td>июль</td><td>август</td>
<tr>
<tr>
<td>Осень</td><td>сентябрь</td>
<td>октябрь</td><td>ноябрь</td>
<tr>
<tr>
<td>Зима</td><td>декабрь</td>
<td>январь</td><td>февраль</td>
<tr>
</table>
```

Результат примера показан на рисунке ниже (Рисунок 11).

Весна	март	апрель	май
Лето	июнь	июль	август
Осень	сентябрь	октябрь	ноябрь
Зима	декабрь	январь	февраль

Рисунок 11. Использование псевдокласса first-child.

В данном примере псевдокласс first-child добавляется к селектору td и устанавливает для него серый фон. Хотя ячейки <td> в первой строке четыре, цветом будет выделено лишь первое упоминание, т.е. название сезона. В остальных случаях ячейки выделены цветом не будут. Со следующей строкой все начинается сначала, поскольку родительский элемент сменился. Итак, псевдокласс first-child удобнее использовать в тех случаях, когда требуется задать разный стиль для первого и остальных однотипных элементов.

К псевдоклассам, задающим язык текста относят lang. В коде HTML язык задается параметром charset тега <meta>. Псевдокласс позволяет для документов, одновременно содержащих тексты на нескольких языках соблюдать правила синтаксиса, характерные них (например, вид кавычек в цитатах).

```
Элемент:lang(язык) { ... }
```

В качестве параметра данного псевдокласса могут выступать: ru — русский, en — английский, de — немецкий, fr — французский, it — итальянский.

Итак, для задания кавычек разного типа достаточно написать следующую стилевую конструкцию[12,13]:

```
<style>
   q:lang(de) { quotes: "\201E" "\201C"; }
q:lang(en) {quotes: "\201C" "\201D"; }
   q:lang(fr), q:lang(ru) {quotes: "\00AB" "\00BB";
}
</style>
  <p>Цитата на французском языке: <q lang="fr">Ce
que femme veut, Dieu le veut</q>.</p>
  <p>Цитата на немецком: <q lang="de">Der Mensch,
versuche die Gotter nicht</q>.</p>
```

Для отображения типовых кавычек в примере используется стилевое свойство quotes, а само переключение языка и соответствующего вида кавычек происходит через параметр lang, добавляемый к тегу <q>.

Псевдоэлементы

Помимо псевдоклассов, каскадные таблицы стилей позволяют работать и с псевдоэлементами. Они позволяют добавлять к тегам новое содержимое, не содержащееся в исходном коде. Синтаксис использования псевдоэлементов такой же, как и у псевдоклассов.

```
Селектор:Псевдоэлемент { Описание правил стиля }
```

Но, в отличие от последних каждый псевдоэлемент может применяться только к одному селектору. Кроме того, псевдоэлементы не могут применяться к внутренним стилям, только к таблицам связанных или глобальных стилей.

Псевдоэлемент after позволяет добавить любое содержимое (текст, символ, картинку и т.д.) после элемента, к которому он назначен. Псевдоэлемент before наоборот вставляет контент до элемента. Параметр content определяет содержимое для вставки, в том числе символы юникода. Таблицу символов юникода можно найти, например, по ссылке http://unicode-table.com/ru/.

Например, для создания списков с уникальными маркерами можно использовать before. В данном примере псевдокласс before устанавливается для тега LI, определяющего элементы списка. Добавление желаемых символов происходит путем задания значения свойства content.

```
 <style> UL
{
    list-style-type: none; /* Прячем маркеры списка
*/
   }
   LI:before {content: "\2605 "; /* Добавляем символ
в юникоде */}
</style>
  <ul>
   <li>Весна</li>
   <li>Осень</li>
  </ul>
```

Результат примера показан на рисунке ниже (Рисунок 12).

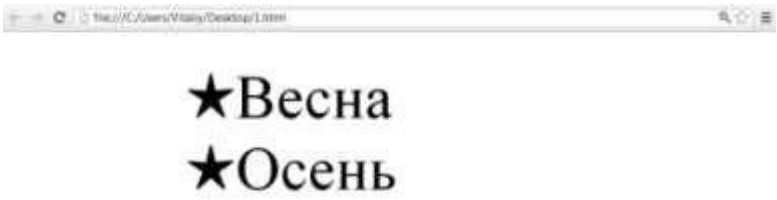

Рисунок 12. Список с нестандартными маркерами.

Заглянув в детские книжки с русскими сказками, можно непременно увидеть «буквицу» - красиво, каллиграфически оформленную, традиционно большую букву. Для ее задания на webстраницах используется псевдоэлемент first-letter. Итак, first-letter определяет стиль первого символа в тексте элемента, к которому добавляется.

Добавим стиль русских сказок на web-страницу. Для этого достаточно добавить к селектору P псевдоэлемент first-letter и установить желаемый стиль буквицы: размер и цвет текста.

```
<style>
P { font-family: sans-serif; color: black;}
P:first-letter { font-family: serif; font-
size: 200%;  color: red; }
</style>
    <p> Жили-были старик со старухой. Вот и говорит
старик старухе: — Поди-ка, старуха, по коробу
поскреби, по сусеку помети, не наскребешь ли муки на
колобок.</p>
```

Результат примера показан на рисунке ниже (Рисунок 13).

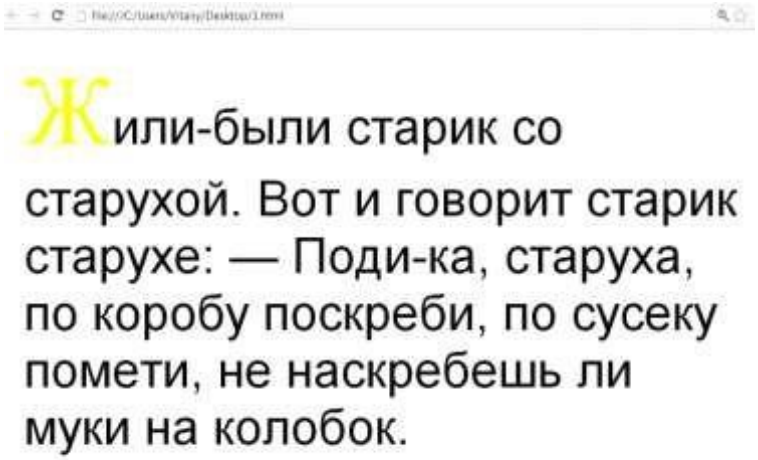

Рисунок 13. Создание выступающего символа.

В данном примере изменяется шрифт, размер и цвет первой буквы каждого абзаца текста.

Псевдокласс first-line в отличие от предыдущего позволяет оформить первую строку блока текста стилями текста, шрифта, цвета текста и фона, а также его обтекания.

```
P:first-line {color: green; font-style: italic; }
```

Длина строки, к которой применяться стиль зависит от шрифта, размеров окна браузера и др. В примере первая строка выделяется цветом и курсивным начертанием.

Специфичность селекторов

В первом параграфе данной главы говорилось о том, что к одному и потому же элементу могут применяться различные стили: стили браузера, стили автора и стили пользователя, имеющие разные приоритеты. Кроме того, на конечный результат представления элемента web-страницы может оказывать его специфичность. Чем специфичность селектора, тем более высокий приоритет имеет правило[12,13].

Специфичность определяется по следующим критериям:

Таблица 7. Расчет специфичности селектора.

показатель	коэффициент
идентификатор	100
класс и псевдокласс	10
селектор тега и псевдоэлемент	1
Встроенный стиль, (параметр style тега)	1000
!important	10000

Если два селектора имеют одинаковую специфичность, то применяться будет тот стиль, что определен в коде ниже.

Попробуем рассчитать специфичность некоторых селекторов.

- * – специфичность =0;
- P – специфичность=1
- P:first-letter – специфичность=1+1=2

- ul ol+li – специфичность=1+1+1=3
- p.red.sl1 – специфичность=10+10+1=21
- #menu ul li – специфичность=100+1+1=102

Чтобы исправить ситуацию, необходимо либо понизить, либо повысить специфичность селектора. Для понижения часто убирают идентификаторы, для повышения – их добавляют, если и этого не достаточно используют ключевое слово !important

☑ Вопросы для самоконтроля

1. В каких случаях стоит применять универсальный селектор?
2. В чем отличие контекстных и дочерних селекторов?
3. Приведите примеры применения соседних селекторов.
4. Что такое псевдокласс?
5. Какие псевдоэлементы Вы знаете?
6. Как рассчитать специфичность селектора?
7. Можно ли выделить среди ссылок те, что ссылаются на почтовый ящик? Если да, то, каким образом?

§ 3.5. CSS спрайты изображений

Закончить рассмотрение возможностей, открываемых использованием таблиц стилей, хочется темой применения для webразметки спрайтов изображений. Данный метод применения CSS оптимизирует технологию работы с фоновыми изображениями на webсайтах, уменьшая количество HTTP запросов и размеров файлов. Существуют некоторые очень креативные техники, используемые настоящими CSS фанатами по всей сети[2]. Среди наиболее знаменитых веб-приложений, использующих сегодня CSS спрайты изображений с более чем 100 включенных различных изображений Amazon, Yahoo, Apple.

CSS-спрайт – это изображение, содержащее внутри себя множество других. По существу берется несколько изображений и складывается в одно (иногда его называют «главное изображение»). Затем, используя параметр CSS background-position, указывается браузеру, какая часть

изображения в данный момент будет видна. Это так называемое поле просмотра (viewport) элемента ее содержащего. Оно обычно зависит от нескольких параметров, таких как ширина, высота, заполнение и т.д. Как в старом советском детстве диафильм покадрово показывал нам кусочки пленки – кадры.

Например, у нас есть главное изображение (пленка) шириной в 200 пикселей и высотой в 50 пикселей (Рисунок 14). Это изображение содержит 4 иконки (кадра на пленке), поэтому каждая иконка имеет 50 пикселей в ширину (200px ÷ 4 = 50px) и высоту также в 50 пикселей.
Слой (окошко диафильма, которое высвечивает кадр), с которым мы будем работать, также имеет размеры 50x50 пикселей. Именно в этом слое будет размещено фоновое изображение.

Рисунок 14. Изображение 200x50 пикселей

Теперь попробуем создать HTML код и CSS для отображения нашего примера. Для этого задаем слою размеры 50 на 50 пикселей, положение фонового изображения в точке (0,0), изображение не должно дублироваться ни по горизонтали, ни по вертикали (background-repeat: no repeat). Фон отображает иконку в левом углу (картинку Google Chrome) главного изображения. Это происходит потому, что установка позиции и по x и по y на 0 означает то же, что поставить положение по x на left(лево), а по y на top(вверх).

```
<style type="text/css"> div.one
{
  background-image: url(icons.gif);
background-position: 0 0;
background-repeat: no-repeat;
width: 50px;   height: 50px;
}
```

```
</style>
</head>
 <body>
<div class="one"></div>
</body>
```

Так нам удастся увидеть только первую иконку. Как же показать остальные? И здесь все так же, как в диафильме – достаточно прокрутить барабан до следующего кадра. Раз каждый кадр имеет размеры 50x50, «прокручивать» необходимо ровно на 50 пикселей по горизонтали, тогда следующая картинка встанет прямо напротив «окошка». Если начальное положение left фоновой картинки равно нулю, значит «при прокручивании», оно примет значение -50px. Так как мы перемещали фон только по оси OX, значение top останется равным нулю (Рисунок 15). Помимо измерения в пикселях, можно использовать и ключевые слова (left, center, right, top, bottom)[2].

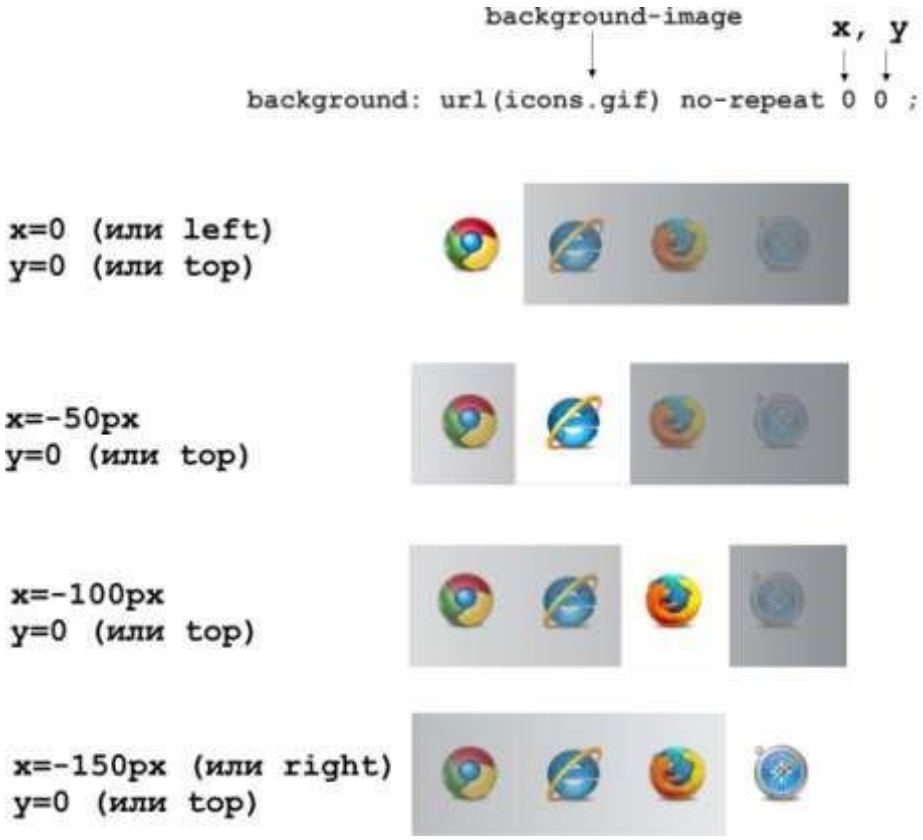

Рисунок 15. Кадрирование спрайта.

Итак, отобразим все четыре иконки в разных слоях. Слои будут отличаться только положением фоновой картинки, потому все общие характеристики соберем в одну группу селекторов.

```
 div.one,div.two,
div.three,div.four
{
  background: url(three-icons.gif) no-repeat;
width: 50px;  height: 50px;
  margin-bottom: 10px;
}

div.one{background-position: 0 0;} div.two
{background-position: -50px 0;} div.three
{background-position: -100px 0;} div.four
{background-position: -150px 0;}
```

Вот так просто и работают CSS спрайты изображений.

Создание спрайтов

Теперь мы умеем работать с готовыми спрайтами. Но не всегда готовые спрайты удовлетворяют требованиям дизайнера. Создание их также не представляет большого труда. Достаточно открыть графический редактор, например, MS Photoshop. Допустим, необходим спрайт из 4 иконки по 32 пикселя. Остается определить горизонтально, вертикально и таблицей будут располагаться иконки в главном изображении. Если иконки квадратные, то принципиального значения не имеет, в случае прямоугольные изображений могут возникать небольшие трудности[2].

В данном случае изображения размером 32x32 пикселя, будем размещать их вертикально. Для создания спрайта необходимо проделать следующую последовательность действий.

1. Создать новый документ шириной в 32 пикселя и высотой в 256 пикселей. (на каждую иконку по 64 пикселя)
5. Залить фон цветом.
6. Установите следующие горизонтальные ориентиры (всего 3): 64px, 128px, 192px.

7. Подставить иконки под каждый ориентир, отцентрировав их по вертикали (желательно применяя инструменты выравнивания, и линии сетки).
8. Сохранить полученное изображение в jpg, png, gif.

Несмотря на то, что иконки занимают лишь 128 пикселей в высоту, итоговая высота была выбрана в два раза большая. Дополнительное место между иконками поможет решить проблемы несовместимости браузеров и увеличение содержащего их элемента[2].

Создание маркированных списков с помощью спрайтов

Рассмотрим пример создания маркированного списка, размещенного в 2 колонки с применением CSS спрайтов (Рисунок 16).

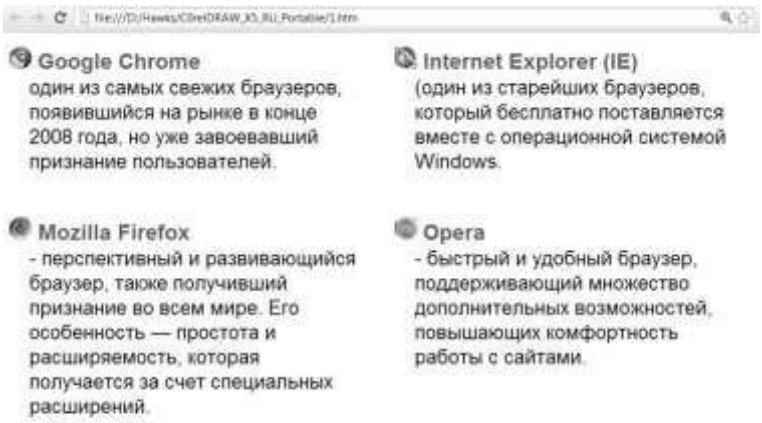

Рисунок 16. Список с маркерами- частями спрайт-изображениями.

Изначально необходимо создать изображение с 4 иконками размером 32x32. Следующий вопрос, который необходимо решить, какой элемент будет содержать фоновые изображения. Элемент li для данной цели мало подходит – его размеры трудно предсказать. В данном случае, исходя из шаблона дизайна, элемент h2 наилучшим образом подходит для содержания спрайта.

Будем использовать неупорядоченный список с фиксированной шириной элементов li, облегчая процесс создания стилей.

HTML-разметка данного примера будет выглядеть следующим образом (указаны только значащие части).

```
<link href="styles.css" rel="stylesheet"
type="text/css">
</head>
<body>
<ul id="list">
      <li>
          <h2 id="google">Google Chrome</h2> <p>
один из самых свежих браузеров, появившийся на
рынке в конце 2008 года, но уже завоевавший
признание пользователей. </p>
        <li>
          <h2 id="ie"> Microsoft Internet Explorer
(IE)  </h2>
   <p> (один из старейших браузеров, который
бесплатно поставляется вместе с операционной
системой Windows. </p>
        <li>
          <h2 id="fox"> Mozilla Firefox </h2>
<p> - перспективный и развивающийся браузер,
также получивший признание во всем мире. Его
особенность — простота и расширяемость, которая
получается за счет специальных расширений.</p>
        <li>
          <h2 id="opera"> Opera </h2>      <p>
- быстрый и удобный браузер, поддерживающий
множество дополнительных возможностей,
повышающих комфортность работы с сайтами.</p>
    </ul>
</body>
```

Теперь можно приступать к формированию таблицы стилей styles.css. Для начала сбросим все отступы, определим свойства шрифта и цвета.

```
* {   margin:
0;   padding:
0;
}
body {
  font: 14px Arial, sans-serif;
color: black;
}
```

Далее определим стили неупорядоченного списка. Убираем маркеры (list-style: none;), верхнее и нижнее поля списка делаем равными по 100 пикселей, а левое и правое - auto, для центрирования в браузере. Ширину элементов li установим равной 270 пикселей и выровняем их слева (float: left). Исходя из того, что ширина всего списка равна 600, каждый третий элемент автоматически будет переведен на новую строку. Этакие «псевдо колонки»[2].

```
ul#list {   list-
style: none;
  width: 600px;   margin: 100px auto;
}   ul# list
li{   width:
270px;
  float: left;   margin: 0 35px 15px 0;
}
```

Теперь зададим стили h2. Очень важный параметр здесь это padding. Прежде всего, задав верхний отступ равным 10 пикселей, а размер шрифта 22 пикселя, определим минимальную высоту элементов h2 равной высоте иконок (32 пикселя). Отступ слева на 42 пикселя - 32 пикселя, необходимые для показа иконки и 10 пикселей полей.

```
 ul#list li h2{
color: green;
font-size: 22px;
 background: url(bg.png) no-repeat;
padding: 10px 0 0 42px;
}
```

Итак, мы получили список в две колонки, но пока с одинаковыми иконками. Берем наше фоновое изображение с 3 вертикальными ориентирами: 64px, 128px, 192px. Задаем параметры backgroundposition, используя данные ориентиры.

```
 ul# list li h2#google{ background-position: 0 0}
ul# list li h2#ie{ background-position: 0 -64px;}
ul# list li h2#fox{ background-position: 0 -128px;}
ul# list li h2#opera{background-position:0 -192px;}
```

Осталось лишь подправить немного текст в каждом заголовке.

```
 ul#list li p{ margin-left:
42px;}
```

Теперь все готово! В заключение отметим, что техника использования «псевдо колонок» имеет свои особенности. «Если бы заголовки были длинными и переносились на следующую строку, это бы толкало контент вниз и заставило бы элементы li также сдвинуться. Если есть вероятность, что контента прибавится, вам лучше будет использовать двухколонную разметку. Также, если элементы h2 вырастут в высоту на достаточное количество, они станут заползать на иконки. Чтобы избежать этого, сделайте больший отступ между иконками на главном изображении»[2].

☑ Вопросы для самоконтроля

1. Что такое спрайт изображений?
2. Как создать спрайт изображений?
3. Благодаря каким свойствам можно управлять отображением изображений, являющимся спрайтом?
4. Какова основная сфера применения спрайтов изображений?

§ 3.6. CSS3. Новые параметры и возможности

CSS3 - это новый стандарт оформления HTML документов, значительно расширяющий возможности веб-разработчиков по сравнению со своим предшественником CSS2.1. Многие возможности, которые были ранее труднодоступны, или требовали использования

дополнительных внешних программ, скриптов или специальных «хитростей» могут теперь легко достигаться в CSS3 за счет использования новых свойств оформления.

С помощью CSS3 возможно:

- Создание элементов со сглаженными углами;
- Более гибкое оформление фоновых изображений элементов;
- Добавление к элементам и тексту тени;
- Использование «небезопасных» шрифтов;
- Применение верстки в несколько колонок;
- Создание различных эффектов переходов и многое другое.

К сожалению, не все рассматриваемые свойства на момент написания пособия поддерживаются браузерами. Для большинства свойств ограничения версий IE9+, Firefox 3.6+, Opera 10+, Chrome 12+, Safari 5+. Особые случаи будут оговариваться отдельно.

При желании можно эмулировать поведение новых свойств CSS3 в старых браузерах, установив предварительно специальную библиотеку css3pie (http://css3pie.com)[14].

Управление фоновыми изображениями

В CSS3 появилось свойство установки размера фоновых изображений background-size, которого так не хватало в традиционном CSS. Теперь возможно автоматически изменять размер фона под размер браузера. Значение можно указывать в пикселях или в процентах, первое значение отвечает за ширину, а второе – за высоту фона.

```
#element1 {
background-image: url("fon.jpg"); background-size:
350px 450px;
}
#element2 {
background-image: url("fon2.jpg"); background-size:
50% 50%;
}
```

Появилась и новая, интересная возможность использовать несколько фоновых изображений одновременно, одна будет проглядывать сквозь другую. Для того чтобы вставить в элемент несколько фоновых картинок, необходимо перечислить путь к ним через запятую в свойстве background-image. Картинки будут накладываться друг на друга в перечисленной очередности (т.е. первая заданная картинка будет отображаться поверх последующих). Для того чтобы применить к фоновым картинкам свойства оформления, необходимо перечислить необходимые значения в нужном порядке через запятую (к примеру, в свойстве «background-position: bottom right, center;» bottom right будет применено к первой, а center − ко второй фоновой картинке)[14].

```
#element1 { background-
image:url(img1.jpg),url(img2.jpg); background-
position: bottom right, center; background-size:
200px 50px,100% 100%; background-repeat: no-repeat,
no-repeat;
} body
{ background-image:url(img_flwr.gif),
url(img_tree.gif);
}
```

Оформление границ элементов

Еще одно свойство, которого жаждали многие дизайнеры – возможность добавления скругленных углов. К каким только ухищрениям они прибегали, пытаясь реализовать одно из направлений WEB 2.0. С помощью нового CSS3 свойства border-radius это стало возможно делать очень просто.

```
div { border:2px
solid; border-
radius:25px;
}
```

border-radius задает радиус скругления, в качестве единицы измерения используются пиксели. Свойство может задаваться единым для всех четырех углов, или уникальным для каждого угла с помощью border-top-left-radius, border-top-right-radius, border-bottom-left-radius и border-bottom-right-radius соответственно.

Да и сама граница не ограничивается теперь пунктирами, точками и валиками с выемками. CSS3 позволяет вставлять произвольные изображения в качестве границы элемента, применяя для этого новое свойство border-image.

Значениями данного свойства является набор параметров: путь, отступ, ширина, повторение. Итак, для того чтобы добавить изображение в качестве фона границы необходимо: указать путь к изображению-границе; указать величину отступа от каждого края изображения для того, чтобы разрезать изображения на 9 частей (верхний левый угол, верхняя сторона, верхний правый угол, левая сторона и т.д.); указать ширину границы в пикселях; и указать должно ли изображение повторяться (repeat), округляться (round) или растягиваться (stretch), чтобы заполнить границу элемента.

Например,

```
div { border-image:url(border.png) 30 30
round;
-webkit-border-image:url(border.png) 30 30 round;
-o-border-image:url(border.png) 30 30 round;
}
```

Заметим, что свойство border-image поддерживается Firefox, Chrome и Safari 6 (IE не поддерживает это свойство), Opera поддерживает альтернативное свойство -o-border-image, а Safari 5 webkit-border-image. Для того чтобы свойство применилось в большинстве браузеров, все эти варианты указываются в правилах описания стилей.

Добавление к элементам тени

CSS 3 сделал шаг и в сторону перемещения web-пространства в трехмерную среду, добавив возможность задания теней для текста и блочных элементов.

С помощью свойства box-shadow тень можно добавить к элементу страницы тень, причем тень может быть внешней и внутренней.

```
box-shadow: горизонтальное_смещение
вертикальное_смещение радиус_размытие размер_тени
цвет –вдавленность‖;
```

В самом простом случае можно указать 3 значения свойства boxshadow: два значения, задающие величину смещения тени по горизонтали и вертикали в пикселях, и третье значение - цвет тени.

```
box-shadow: 5px 5px black;
```

Также можно указать радиус разброса тени и размер тени.

```
box-shadow: 5px 5px 5px 3px black;
```

Кроме того, с помощью значения inset можно указать, что тень должна быть не внешней, а внутренней («вдавленной»).

```
box-shadow: 0px 0px 5px 3px black inset;
```

Также тень можно добавить к текстовым элементам, причем к тексту одного элемента может быть добавлено одновременно несколько теней. За добавление такого стиля форматирования отвечает textshadow.

При задании тени для текста необходимо указать последовательно: величину в пикселях смещения тени от текста по горизонтали и вертикали (может быть и отрицательной), а также радиус размытия и цвет тени.

```
h1 { text-shadow: 5px 5px 5px #FF0000;}
```

Новые стили оформления текста

С помощью нового CSS3 свойства word-wrap можно указать, чтобы длинные слова, выходящие за пределы границ элемента, разбивались и переносились на новую строку.

```
word-wrap: normal|break-word; div
{word-wrap:break-word;}
```

В предыдущих версиях CSS разработчики были вынуждены использовать только те шрифты, которые гарантированно установлены на компьютере пользователя. С помощью CSS3 стало возможным использование любых шрифтов, так называемых «небезопасных», путем загрузки их через свойство @font-face.

Поддержка шрифтов различных форматов приведена в таблице

Таблица 8. Поддержка шрифтов различных форматов браузерами.

Группы шрифтов\ браузеры	**Internet Explorer 9+**	**Firefox**	**Chrome**	**Opera**	**Safari**
WOFF (Web Open Font Format)	+	+	+	+	+
TTF (True Type Fonts) и OTF (OpenType Fonts)	-	+	+	+	+
SVG шрифты/формы	-	-	+	+	+
EOT (Embedded OpenType)	+	-	-	-	-

Для того чтобы шрифт отображался корректно во всех браузерах, его необходимо подключить к странице как минимум в двух форматах.

```
@font-face { font-family: sansation; /* Задаем
имя шрифта */ src:url('sansation.ttf'), /*
Указываем местонахождение шрифта в формате .ttf
```

```
*/ url('sansation.eot') /* Указываем
местонахождение шрифта в формате .eot */
}
```

Для использования подключенного шрифта достаточно указывать его имя в font-family элемента, который хотим отформатировать.

Также возможно добавить подключенному шрифту курсивное или жирное начертание. Для этого необходимо вставить в @font-face свойство font-style:italic или font-weight:bold и указать путь к шрифту в соответствующем начертании.

```
/* Подключаем шрифт */
@font-face{ font-family:sansation;
src:url('sansation.ttf'),url('sansation.eot')
}
/* Подключаем шрифт для курсивного начертания */
@font-face{ font-family:sansation; font-
style:italic;
src:url('sansationitalic.ttf'),url('sansationitalic
.eot');
}
/* Применяем подключенный шрифт к абзацам */ p
{font-family:sansation;}
```

Разбиение текста на столбцы

Благодаря CSS 3 web-страницы можно превращать в газеты с несколькими колонками. Свойство column-count задает количество столбцов, на которые необходимо разбить текст выбранного элемента, column-width - ширину столбцов, column-rule - ширину, цвет и стиль оформления пространства между столбцами, а с помощью column-gap можно определить величину отступа между столбцами текста.

```
div { column-count:4; column-
gap:50px; column-rule:2px
```

```
dotted #7F0055; column-
width:150px;
}
```

Примечание: данные свойства на данный момент не поддерживаются браузером Internet Explorer 9 и более ранних версий, для браузеров Chrome и Safari перед свойством требуется добавить префикс -webkit, а для Firefox префикс -moz.

CSS3 трансформирование

Теперь стало очень просто перемещать и поворачивать элементы в плоскости web-страницы с помощью многофункционального свойства transform. В качестве значения данного свойства должна указываться одна из функций трансформирования.

Так, с помощью функции translate(x,y) можно сместить элемент на указанное количество пикселей по горизонтали и вертикали. А с помощью функции rotate(градусы) - повернуть элемент на указанное количество градусов по часовой стрелке.

```
Div {transform: rotate(30deg);}
```

Растянуть элемент в ширину или высоту можно с помощью scale(x,y), а скосить элемент на указанное количество градусов по горизонтали и вертикали – с помощью skew(x,y).

```
Div {transform: skew(30deg,20deg);}
```

Полный список всех возможных преобразований приведен в таблице 41.

Таблица 9. Функции трансформирования в CSS3.

функция	**показат**	**действие**	**Ед.**
	ели		**изм**
translate(x,y)		Смещает элемент от изначальной позиции по горизонтали и вертикали.	Px, число

translateX(x)	e	Смещает элемент по горизонтали.	
translateY(y)	f	Смещает элемент по вертикали.	
scale(x,y)		Растягивает элемент по вертикали и горизонтали	
scaleX(x)	a	Растягивает элемент по горизонтали	
scaleY(y)	d	Растягивает элемент по вертикали	
rotate (градусы)		Поворачивает элемент по часовой стрелке	deg, число
skew(x,y)		Скашивает элемент по горизонтали и вертикали	
skewX(x)	b	Скашивает элемент по горизонтали	
skewY(y)	c	Скашивает элемент по вертикали	
matrix(a,b,c,d,e,f)		Трансформирует элемент с помощью 6 показателей	

Примечание: свойства transform работают во всех современных браузерах (IE10+, Firefox, Opera), но для браузеров Chrome и Safari требуется префикс -webkit-, а для браузера IE 9 требуется префикс -ms-. IE 8 и более ранние версии transform не поддерживают.

Эффекты динамического перехода и прозрачность

CSS3 вобрал в себя все хорошие задумки браузеров. Так, существовавшее некогда в IE свойство filter:alpha, позволяющее влиять на уровень прозрачности изображений переродилось в новом стандарте.

Сегодня для создания прозрачных элементов во всех браузерах кроме Internet Explorer используется свойство opacity:x, где x – значение, которое может изменяться от 0.0 (полностью прозрачный элемент) до 1.0 (полностью непрозрачный элемент). А в старом добром Internet Explorer продолжает использоваться свойство filter:alpha(opacity=x), где x – значение, которое может изменяться от 0 (полностью прозрачный элемент) до 100 (полностью непрозрачный элемент).

Итак, для задания изображений, степень прозрачности которых 20% достаточно написать

```
#element { opacity:0.8; filter:alpha(opacity=80);}
```

Та же участь постигла и transition. С помощью старого нового CSS3 свойства transition можно создавать эффекты перехода, для этого достаточно указать, какое CSS свойство будет изменяться и скорость выполнения этих изменений в секундах.

Например, изменение ширины элемента в течение 4 секунд.

```
#element
{width:200px; transition: width 4s;}
#element:hover
{width:500px;}
```

Для того, чтобы добавить эффект перехода к нескольким свойствам достаточно просто перечислить их названия через запятую.

```
#element { background-color:#E869AA; color:#000;
width:200px; transition: color 4s, width 4s,
background-color 4s;
}
#element:hover
{ color:#FFFFFF; width:400px;  background-
color:#880045;
}
```

Итак, transition позволяет задать значения четырех различных свойств перехода в одном определении. transition-property отвечает за имя CSS свойства, к которому будет применен эффект перехода.

С помощью transition-duration можно указать время выполнения перехода (по умолчанию имеет значение 0). При этом transition-timingfunction позволяет задать функцию смягчения, отвечающую за плавность выполнения перехода (по умолчанию имеет значение 'ease').

Для задания задержки перед началом выполнения достаточно указать значения свойству transition-delay перехода (по умолчанию имеет значение 0)[5].

```
div { transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s; /* Safari and
Google Chrome*/
-webkit-transition-property:width;
-webkit-transition-duration:1s;
-webkit-transition-timing-function:linear;
-webkit-transition-delay:2s;
}
```

Примечание. Internet Explorer 10, Firefox, Chrome, и Opera поддерживают свойство transition, для Safari и Chrome требует префикс -webkit-.

☑ Вопросы для самоконтроля

1. В чем преимущества и недостатки CSS3?
2. Какие примеры того, как ранее труднодоступные эффекты легко реализуются с помощью CSS3, вы можете привести?
3. Какие форматы шрифтов, используемые для подгрузки нестандартных начертаний, вы знаете? Какие форматы шрифтов поддерживают различные браузеры?

☑ Лабораторная работа № 2

ВАРИАНТ 1.

Используя полученные знания, примените к созданному в лабораторной №1 отрывку

☑ *Задание 1.*

- единый стиль оформления заголовков с разреженным межбуквенным расстоянием и отступом от основного текста,
- выделите цветом и размером все встреченные в параграфах жирные начертания
- добавьте страницу оглавление в виде маркированного списка с картинками, заданными через таблицу стилей
- оформите каждый параграф с отступом красной строки,
- создайте два класса ссылок: для оглавления и межстраничные.
- создайте буквицу для первого абзаца текста,
- добавьте на одну из страниц: фоновое изображение заданного размера, без повторений
- добавьте на одну из страниц таблицу и подсвечивайте текущую строку таблицы (выделение фоновым цветом и размером шрифта),

С помощью свойств CSS3 создайте страницу, обладающую следующими свойствами и элементами:

☑ *Задание 2*

- два разных фона страницы с заданными размерами,
- полупрозрачная фотография с тенью нестандартного цвета,
- заголовки оформлены нестандартным рукописным шрифтом,
- информация на странице представлена в виде трех колонок,
- оформите кнопку с текстом «Отправить» в виде красной кнопки со скругленными углами, чтобы при наведении в течение 5 секунд она постепенно увеличивалась в размере и меняла цвет. Добавьте к кнопке тень.

Оформите все в виде подключаемых таблиц стилей.

ВАРИАНТ 2.

Используя полученные знания, примените к созданному в лабораторной №1 отрывку

☑ *Задание 1.*

- единый стиль оформления заголовков с разреженным межсловным расстоянием и отступом от основного текста,
- выделите цветом и размером все встреченные в параграфах курсивные начертания
- добавьте страницу оглавление в виде маркированного списка с картинками, заданными через таблицу стилей
- оформите каждый параграф с отступом красной строки,
- создайте два класса ссылок: для оглавления и межстраничные.
- создайте буквицу для первого абзаца текста,
- добавьте на одну из страниц: фоновое изображение заданного размера, без повторений
- добавьте на одну из страниц таблицу и подсвечивайте текущую строку таблицы (выделение фоновым цветом и размером шрифта),

С помощью свойств CSS3 создайте страницу, обладающую следующими свойствами и элементами:

☑ *Задание 2*

- два разных фона страницы с заданными размерами,
- полупрозрачная фотография с тенью нестандартного цвета,
- заголовки оформлены нестандартным рукописным шрифтом,
- информация на странице представлена в виде двух колонок,
- оформите кнопку с текстом «Заказать» в виде кнопки со скругленными углами, чтобы при наведении в течение 7 секунд она постепенно увеличивалась в размере и меняла цвет. Добавьте к кнопке тень.

Оформите все в виде подключаемых таблиц стилей.