МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
“Оренбургский государственный университет”

Л.Ф. НАСЕЙКИНА

ПРОГРАММИРОВАНИЕ ДИНАМИЧЕСКИХ WEB-САЙТОВ СРЕДСТВАМИ КАСКАДНЫХ ТАБЛИЦ СТИЛЕЙ (CSS)

Рекомендовано Ученым советом государственного образовательного учреждения высшего профессионального образования “Оренбургский государственный университет” в качестве учебного пособия для студентов, обучающихся по программам высшего профессионального образования по специальности “Программное обеспечение вычислительной техники и автоматизированных систем”

Оренбург 2007

УДК 004.7(075)
ББК 32.973.202я73
Н 31

Рецензент
доктор технических наук, профессор Пищухин А.М.

Насейкина Л.Ф.

Н – 31 **Программирование динамических Web-сайтов средствами каскадных таблиц стилей (CSS): учебное пособие. / Л.Ф. Насейкина. - Оренбург: РИК ГОУ ОГУ, 2007. – 155 с.**

ISBN ……

Учебное пособие посвящено изучению каскадных таблиц стилей (CSS). В рамках пособия рассматриваются особенности создания Web-сайтов, оформленных в едином стиле с использованием CSS. Изучаются способы внедрения каскадных таблиц стилей в тело HTML-документа: внутренние, внешние, внедренные таблицы стилей.

Раскрываются вопросы совестного использования CSS с элементами языка управления сценариями JavaScript при создании динамических эффектов.

Учебное пособие предназначено для студентов, обучающихся по программам высшего образования по специальности 230105.65 - “Программное обеспечение вычислительной техники и автоматизированных систем”, при изучении дисциплины “Сетевые информационные технологии”.

ISBN ……

Н 1404000000
6Л9-06

ББК 32.973.202я73

Содержание

Введение

Каскадные таблицы стилей - CSS (Cascading Style Sheets) были революцией, потрясшей WWW. CSS - это язык, содержащий набор свойств для описания внешнего вида любых HTML-документов. С его помощью дизайнер имеет полный контроль над стилем и расположением каждого элемента Web-страницы, что проще и гораздо функциональнее использования обычного набора HTML-тегов.

Язык CSS позволяет создать единый стиль всех страниц Web-сайта, позволяющтй отличить уникальный авторский дизайн разрабатываемого сайта от других Web-сайтов.

Другой отличительной особенностью каскадных таблиц стилей является возможноть превратить статичную страницу, написанную на языке HTML в яркий динамический Web-сайт. Во многом это достигается за счет совестного использования CSS с элементами языка управления сценариями JavaScript.

Целью данного пособия является подготовка студентов к умелому использованию всех возможностей каскадных таблиц стилей при создании своих динамических Web-сайтов. Изложение материала начинается с введения в язык CSS, рассматриваются версии этого языка CSS 1 и CSS 2. Далее студентам предлагается изучить синтаксис языка и понятие “селектор”, рассматриваются различные виды селекторов.

Затем представлены правила внедрения языка CSS в тело HTML-документа, изучаются внешние, внутренние и внедренные таблицы стилей. После изучаются свойства и единицы измерения, используемые в CSS. Также студентам предоставляется возможность изучения понятия “позиционирование” элементов и предлагается изучение таких видов как абсолютное, относительное позиционирование, фиксированные блоки и стандартный поток.

Далее рассматриваются фильтры, как статические, так и динамические. В завершающей главе студентам предлагаются примеры создания динамических эффектов за счет совместного использования CSS и элементов языка JavaScript.

После окончания изучения предлагаемого учебного пособия студенты будут уметь разрабатывать динамические Web-сайты, оформленные единым стилем и использовать обработчики языка JavaScript для создания визуальных эффектов.

Предлагаемое учебное пособие состоит из введения, семи глав, приложений, заключения и списка рекомендуемой литературы. Изложение материала во всех разделах опирается на множество примеров. В приложениях представлены контрольные вопросы для закрепления изученного материала, а также тесты для самопроверки.

В заключение, автор выражает признательность рецензенту и научному редактору за внимательное прочтение работы и замечания, способствовавшие улучшению качества предлагаемого учебного пособия.

1 Введение в CSS

Язык HTML в своем развитии ушел далеко от простого языка разметки. В него постепенно добавлялись элементы и атрибуты, отвечающие за представления информации в окне броузера. В научных средах, в которых и зародилась Web, люди более сосредоточены на содержании документов, нежели на их представление. Разработчики четко осознали несостоятельность такого подхода. Вскоре была найдена другая технология, позволяющая описывать внешний вид элементов, создаваемых языком разметки.

Cascading Style Sheets (CSS) - ***таблицы каскадных стилей***, представляют собой наибольшее достижение для дизайнеров Web-страниц, расширяющее возможности улучшения внешнего вида страниц.

Известно, что компонентами страницы являются блоки текста, графика и встроенные в страницу приложения. Размер и границы каждого из этих компонентов в рамках HTML-разметки задаются с разной степенью точности. Размер графики и приложений можно задать с точностью до пикселя. Размеры текстовых блоков в HTML задать нельзя. Они вычисляются броузером на основе относительного размера шрифта умолчания. Автор страницы не может заранее определить настроек броузера пользователя, что существенно ограничивает число вариантов представления информации на странице.

Таблица стилей представляет собой набор свойств, записанный в виде правил, которые определяют, как некоторый элемент HTML должен выглядеть в окне броузера или на другом устройстве вывода.

Таблицы стилей упрощают разметку HTML и существенно снижают участие языка HTML в представлении документа. Они предоставляют как авторам, так и пользователям возможность управлять представлением документов - шрифтами, выравниванием, цветами и т.д. Информацию о стиле можно указать для отдельных элементов или групп элементов, в документе HTML или во внешних таблицах стилей /1/.

Кроме размера компонентов, таблицы стилей позволяют очень гибко оформлять документ - определять цвет и начертание текстового фрагмента, изменять эти параметры внутри текстового блока, выполнять выравнивание текстового блока относительно других блоков и компонентов страницы, создавать рамки вокруг объектов и многое другое. Таблица стилей единожды описывает специфический формат, присваивает ему имя, и далее он употребляется там, где потребуется. Причем, действие таблиц стилей может распространяться как на конкретный элемент, так и целиком на страницу, или даже на множество страниц сразу.

Наличие всех этих возможностей позволяет говорить о CSS как о средстве разделения логической структуры документа и формы его представления. Логическая структура документа определяется элементами HTML-разметки, в то время как форма представления каждого из этих элементов задается CSS-описанием элемента.

На самом простом уровне, стиль (style) является всего лишь правилом, сообщающим броузеру, как интерпретировать отдельные теги HTML. С каждым тегом связан ряд свойств, значения которых определяют его вид при выводе броузером.

Само описание стиля включает селектор тега, указывающий имя тега, на который распространяется правило стиля, и фигурные скобки, содержащие одну или более пар объявлений вида **“свойство_стиля:значение”**, разделенных точкой с запятой:

```
селектор_тега
{
 свойство_стиля1:значение1;
 свойство_стиля2:значение2;
 свойство_стиля3:значение3;
}
```

Все свойства, использующиеся в таблицах стилей, условно можно разделить на несколько групп:

- управляющие видом шрифта (гарнитура, кегль, цвет, наклон, жирность);
- управляющие форматированием абзаца (отступ красной строки, выравнивание, интерлиньяж, расстояние между словами);
- управляющие свойствами блока (отступы слева-сверху-справа-снизу, местоположение блока в документе, видимость блока);
- другие (цвет ссылок, изменение внешнего вида курсора).

Одной из основных особенностей таблиц стилей является каскадность. Она означает, что при обработке стилей используется специальный механизм *каскадирования.* Cascading Style Sheets (Таблицы Каскадных Стилей) - это язык, содержащий набор свойств для описания внешнего вида любых HTML документов. С его помощью дизайнер имеет полный контроль над стилем и расположением каждого элемента Web-страницы, что проще и гораздо функциональнее использования обычного набора HTML тегов.

Для одного документа могут быть заданы различные правила. Один набор правил может определяться разработчиком документа, другие правила могут быть заданы пользователем. Кроме того, в каждом броузере имеется набор правил, используемых по умолчанию; они используются, если какое-либо из свойств явно не определено. Средства каскадирования собирают и объединяют различные правила, а также разрешают возникающие при этом конфликты /2/.

Свойство каскадности тесно связано со способом подключения стилей к HTML-документу, которое возможно осуществить тремя способами: **inline-описание (внедренные стили)**, **описание в секции заголовка HTML-документа (внутренние) и использование внешнего файла (внешние).**

Внедренные стили (встроенные с помощью атрибута style непосредственно в теги документа) имеют наивысший приоритет и будут

применены броузером в любом случае, даже если возникает конфликт с внедренными или внешними стилями. Наивысший приоритет у тега.

Внутренние стили (перечисленные в теге <style> в заголовке документа) - чуть меньший приоритет, будут применяться во всех случаях, кроме случаев возникновения конфликта с inline-стилями (при возникновении такого конфликта будут применены inline-стили).

Внешние или связанные стили (стили, присоединенные к html-файлу посредством тега <link>) - наименьший приоритет, будут применены только после того, как броузер убедится в отсутствии аналогичных правил во всех остальных типах стилей.

На практике приходится пользоваться всеми тремя способами, и здесь в игру вступает “каскадность” стилей, позволяющая переопределять стили.

Что касается различных версий CSS, то в декабре 1996 году был стандартизован первый уровень *каскадных таблиц стилей* (CSS 1), который определил правила описания стилей визуального отображения элементов HTML-документов. В мае 1998 года был принят стандарт второго уровня таблиц стилей (CSS 2), который существенно расширил возможности таблиц стилей /3/.

Первая реализация таблиц стилей (CSS 1) предоставляла возможность выбора элемента, к которому применяется стиль по следующим признакам:

- к любому элементу заданного типа, например ко всем заголовкам H1;
- к определенному классу элемента. Класс определяется заданием атрибута class внутри элемента HTML: <P class=“blue”> Текст параграфа будет синим </P>;
- к уникальному элементу, определенному идентификатором внутри элемента HTML при помощи атрибута id: <H1 id=“unique”> Уникальный заголовок </H1>;
- к элементам, находящимся непосредственно внутри другого элемента (или в контексте другого элемента). Например, для всех элементов STRONG, находящихся внутри элемента параграфа, можно задать выделение нежирным шрифтом, красным цветом: <P> Можно задать <STRONG> выделение </STRONG> внутри текста </P>;
- к первой букве элемента, т.е. создавать буквицы;
- к первой строке элемента;
- к активным, посещенным и не посещенным ссылкам.

Блоковая модель в CSS 1 была реализована таким образом, что позволяла задавать:

- размеры блоков;
- размеры отступов, полей и рамок, тип и цвет рамок;
- позиционирование блоков: выравнивание и обтекание текстом.

Для управления шрифтом были доступны такие параметры:

- начертание и насыщенность (степень жирности);

– размер в разных единицах измерения;
– гарнитура шрифта.

Можно было устанавливать цвет фона и фоновое изображение для страниц и отдельных блоков, управлять его повторением по горизонтали и вертикали, а также помещать фоновое изображение в любую точку экрана, задав координаты этой точки. Это невозможно было сделать с помощью средств только HTML.

Для текстовых блоков можно устанавливать свойства оформления, регулировать расстояние между словами и
высоту строки, задавать выравнивание и
цвет текста. Для списков можно устанавливать тип маркера и использовать в качестве маркера любое графическое изображение, подключаемое из файла.

Стандарт CSS 2 основывается на CSS 1 и дополняет его некоторыми свойствами и методами, позволяющими в значительной степени отделить структуру HTML-документа от его визуального оформления.

Появилась возможность писать таблицы стилей для различных устройств, указывая тип устройства в атрибуте media. Поддерживаются различные типы устройств: голосовые броузеры, проекторы, печатающие устройства и карманные компьютеры /4/.

Появились новые возможности выбора элемента, к которому применяется стиль. Кроме тех, что были в CSS 1, добавилась возможность применить стили к следующим элементам:

– к элементу, обладающему фокусом ввода, или к элементу при наведении на него курсора мыши;
– к элементу, который является прямым потомком другого элемента в дереве элементов;
– к группе элементов, имеющих одинаковые атрибуты. Например, ко всем элементам, имеющим заданный атрибут в открывающем теге, или же к элементам, имеющим одинаковые значения некоторого атрибута, - скажем, ко всем элементам, выровненным по центру (align=“center”);
– ко всем элементам сразу, благодаря введению универсального селектора, который обозначается символом звездочка (*).

Управление шрифтами на странице позволяет теперь скачивать недостающие шрифты с сервера, что позволяет значительно расширить перечень используемых в работе шрифтов. Это, соответственно, может уменьшить количество графических элементов на странице с надписями, написанными декоративными или нестандартными шрифтами. Теперь такие надписи можно оформлять в виде текста, а если у пользователя нет нужного шрифта, его можно скачать.

Появилась возможность изменять вид курсора, например, при наведении его на некий элемент и использовать собственные графические изображения вместо стандартных курсоров операционной системы.

Улучшилась блоковая модель, стало возможным более точно размещать блоки на странице, делать их невидимыми, а также многие появились другие возможности /5/.

2 Синтаксис таблиц стилей

CSS - это язык, в котором параметры оформления документа выражаются с помощью правил. Правило (rule) состоит из селектора и блока деклараций, заключенного в фигурные скобки. Основной формат правила CSS выглядит так: ***selector { propeties }.***

Например:

```
H1 {
  font-weight: bold;
  font-size: 12pt;
  font-family: Helvetica;
  font-variant: normal;
  font-style: normal;
}
```

Здесь H1 - это селектор, за которым следует блок деклараций. Каждая декларация, состоит из названия свойства (например, font-weight), символа двоеточия (:) и значения свойства (например, bold). Декларации должны разделяться меду собой точкой с запятой (;).

Селектор определяет правило и управляет его применением. Свойства (propeties) определяют, как будут выглядеть элементы Web - страницы, определенные правилом. В настоящий момент в CSS2 используются следующие виды селекторов: селектор типа, селектор класса, ID-селектор, контекстные селекторы /7/. Рассмотрим подробнее каждый из них.

2.1 Селектор типа

Селектор типа *(Selectors)* переопределяет действие (определенное броузером пользователя) стандартных тегов HTML, таких как <H1> и <P> и т.д.

Синтаксис: селектор {свойства}

Селектор типа, как и HTML-тег, не обращает внимание на регистр, зато каждый параграф <P> должен иметь закрывающий тег.

Любой элемент HTML - это возможный CSS-селектор. Свойства селектора определяют стиль элемента, для которого он определен.

Например, H1 {color: red; size:20pt;}

При реализации данного кода все элементы H1 в документе будут красного цвета, размером в 20 точек (pt, point).

В CSS имеется возможность придания одинаковых свойств сразу нескольким элементам HTML. В этом случае при определении селекторы перечисляются через запятую перед блоком свойств.

Например,

```
h1, h2, h3, p, strong
{color: green; font-style: italic;}
```

В результате все элементы h1, h2, h3, p и strong станут зелеными, будут иметь начертание курсив.

2.2 Селектор класса

Class - атрибут элемента в HTML, определяющий его класс. В CSS можно описать собственные стили для различных классов одних и тех же элементов /8/.

Селектор класса *(Class Selectors)* не связан со специфическим HTML – тегом, можно применить этот селектор к любому тегу, написав Class=“selector” в пределах действия тега.

Синтаксис: селектор.класс {свойства}

Например, H1.blue {color: blue; size: 20pt;}

В результате, все элементы H1 с атрибутом CLASS=“blue” станут синими.

Приведем пример использования атрибута class для создания собственного стиля. Здесь информация о стиле применяется только к элементам H1 определенного класса. При этом влияние на другие теги не оказывается.

```
<HTML>
<HEAD>
<TITLE>Пример селектора класса</TITLE>
<STYLE type= “text/css”>
H1.myclass {font-family: courier; font-color: navy; text-align: center}
</STYLE>
</HEAD>
<BODY>
<H1 class=“myclass”> Наш стиль влияет на этот заголовок </H1>
<H1> А на этот заголовок наш стиль не влияет</H1>
</BODY>
</HTML>
```

Реализация данной HTML-страницы в броузере представлена на рисунке 1.

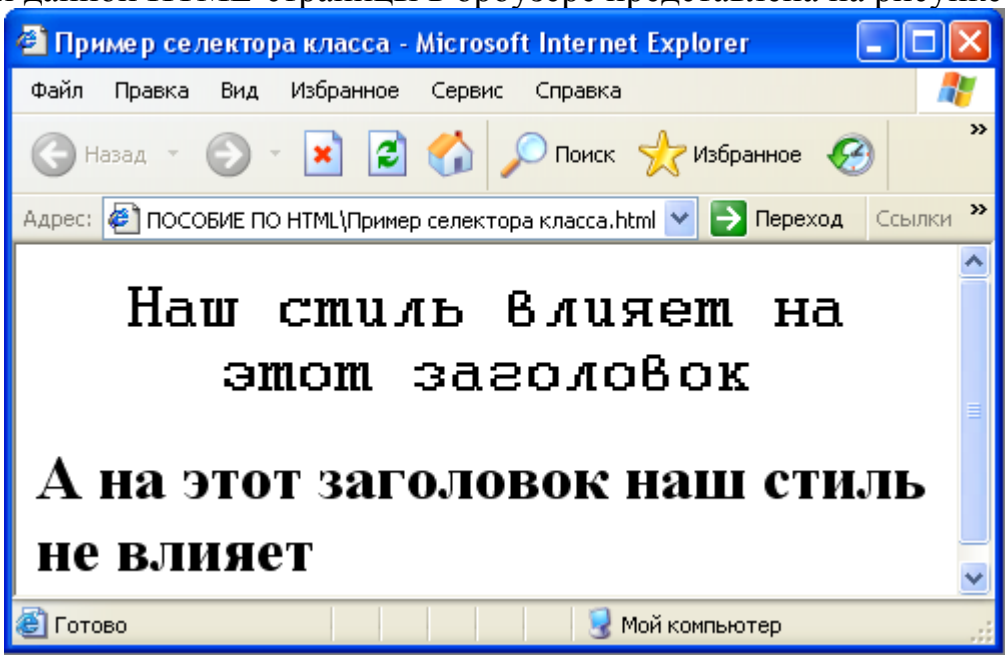

Рисунок 1 – Пример использования селектора класса

Другой пример использования атрибута class. Допустим, необходимо создать Web-страницу, на которой будет два вида абзацев <P>, причем оба вида будут постоянно чередоваться и часто повторяться.

Типичный пример такой страницы - интервью, в котором чередуются вопросы журналиста и ответы интервьюируемого. Естественно, при создании такой страницы необходимо визуально отделить вопросы и ответы друг от друга. Если бы пришлось это делать теми средствами CSS, которые рассмотрены выше, то пришлось бы создавать две различные таблицы стилей. Но можно создать в одной таблице стилей два различных класса абзацев с помощью селектора класса. Это будет выглядеть следующим образом:

```
<HTML>
<HEAD><TITLE>Пример использования селектора класса</TITLE>
<STYLE>
p.ask{
font-style: italic;
font-weight: bold;
font-family: Arial, sans-serif;
font-size: 10pt;
color: gray;
margin-left: 15px;}
p.answer {
font-family: Courier;
font-size: 10 pt; color: black;
margin-left: 10px;}
</STYLE>
</HEAD>
<BODY bgcolor= “#cococo”>
<H3>Интервью:</H3>
<p class= “ask”>Вопрос журналиста</p>
<p class= “answer”>Ответ интервьюируемого. Ответ интервьюируемого.
Ответ интервьюируемого. Ответ интервьюируемого.</p>
<p class= “ask”>Вопрос журналиста</p>
<p class= “answer”>Ответ интервьюируемого. Ответ интервьюируемого.
Ответ интервьюируемого. Ответ интервьюируемого.</p>
</BODY>
</HTML>
```

В приведенном примере вопросы журналиста будут отображаться шрифтом Arial серого цвета, полужирным, курсивного начертания, размером 10 пунктов, с отступом в 15 пикселов от левого края страницы. Ответы же будут отображены шрифтом Times New Roman размером 12 пунктов черного цвета. Важно не забывать прописывать параметр class различным классам абзацев непосредственно в коде html. Таким образом можно создавать любое количество классов для любых элементов страницы.

В броузере эта HTML-страница будет выглядеть следующим образом:

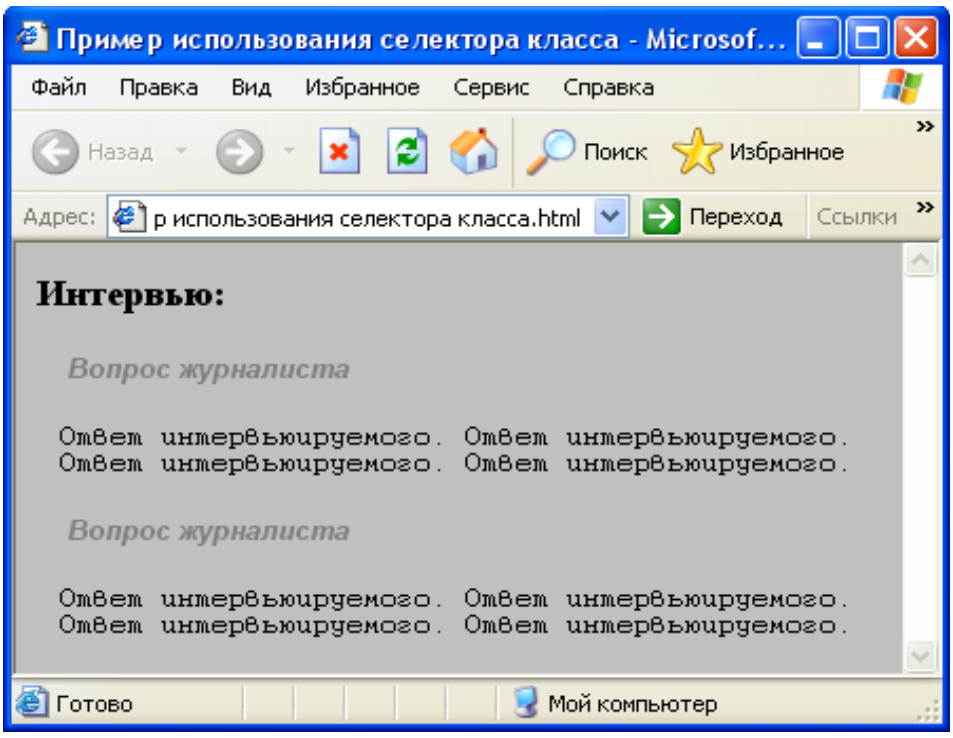

Рисунок 2 – Пример использования селектора класса

Классы могут так же быть описаны без явного привязывания их к определенным элементам.

Синтаксис: .класс {свойства}

Например, .green {color: green;}

В данном случае все элементы с атрибутом CLASS="green" станут зелеными.

Рассмотрим пример описания класса без явного привязывания их к элементам:

```
<HTML>
<HEAD>
<STYLE>
.warning {font-family: Arial Narrow, sans-serif;
font-size: 10pt;
color: red}
</STYLE>
</HEAD>
<BODY>
<P CLASS= "warning">Внимание!</P>
</BODY>
</HTML>
```

Эта стилевая таблица определяет селектор класса, называемый *.warning*, а HTML - код применяет этот класс к одному параграфу. Если имя начинается с точки - это определяемые пользователем имена *классов*. Код CLASS="warning" с тегом <P> отображает весь текст параграфа шрифтом без засечек Arial Narrow высотой 10 точек красным цветом. Селектор класса лучше всегда писать строчными буквами.

2.3 ID - селектор

ID - селектор (ID-Selectors) или селектор-идентификатор, определяет свойства отдельных элементов на странице. В каскадных таблицах стилей имеется возможность присвоения уникальным элементам идентификаторов (id). ID - индивидуально именованный стиль. С его помощью можно создавать стилистические исключения среди элементов одного класса.

Идентификаторы используются, в основном, для придания одному или нескольким элементам одного класса индивидуальных свойств /8/.

Синтакс: #id {свойства}

При этом, необходимо поставить перед именем селектора знак #, если определяется правило для стиля, если же определяется ID, то этот знак не ставится.

Например, атрибут ID="tulip" определяет параграф в последней строке, а селектор, определенный во второй строке, определяет два свойства CSS для этого параграфа.

```
<HTML>
<HEAD>
<STYLE >
#tulip {color: yellow; font-size:20pt;}
</STYLE>
</HEAD>
<BODY>
<P id="tulip">На этот абзац действует заданный стиль</P>
</BODY>
</HTML>
```

Приведем пример использования атрибута id, чтобы задать для абзаца синий цвет, начертание курсив:

```
<HTML>
<HEAD>
<STYLE type="text/css">
.blue {font-style: italic; color: blue}
</STYLE>
</HEAD>
<BODY>
<P class= "blue">...Текст синим курсивом...</P>
</BODY>
</HTML>
```

А теперь добавим к этому классу ID-селектор "boldunderline", в котором все элементы будут отображены жирными шрифтом с подчеркиванием. Произойдет как бы синтез свойств класса blue и идентификатора boldunderline. Элементы класса blue со значением ID "boldunderline" станут отображаться жирным шрифтом, подчеркнутым, синего цвета, начертание курсив.

```
<HTML>
```

<HEAD>
<TITLE> Пример ID-селектора </TITLE>
</HEAD>
<STYLE>
.blue {color:blue; font-style:italic}
#boldunderline {text-decoration: underline; font-weight: bold}
</STYLE>
<BODY>
<P class= “blue”>Пример использования селектора класса blue – текст синего цвета, курсив
</P><p id=“boldunderline”>Пример ID-селектора – текст полужирный, подчеркнутый
</P>
<p class= “blue” id= “boldunderline”> Пример наложения селектора класса и ID-селектора - текст синего цвета, курсив, полужирный, подчеркнутый </P>
</BODY>
</HTML>

Реализация данной HTML-страницы в броузере представлена рисунке 3.

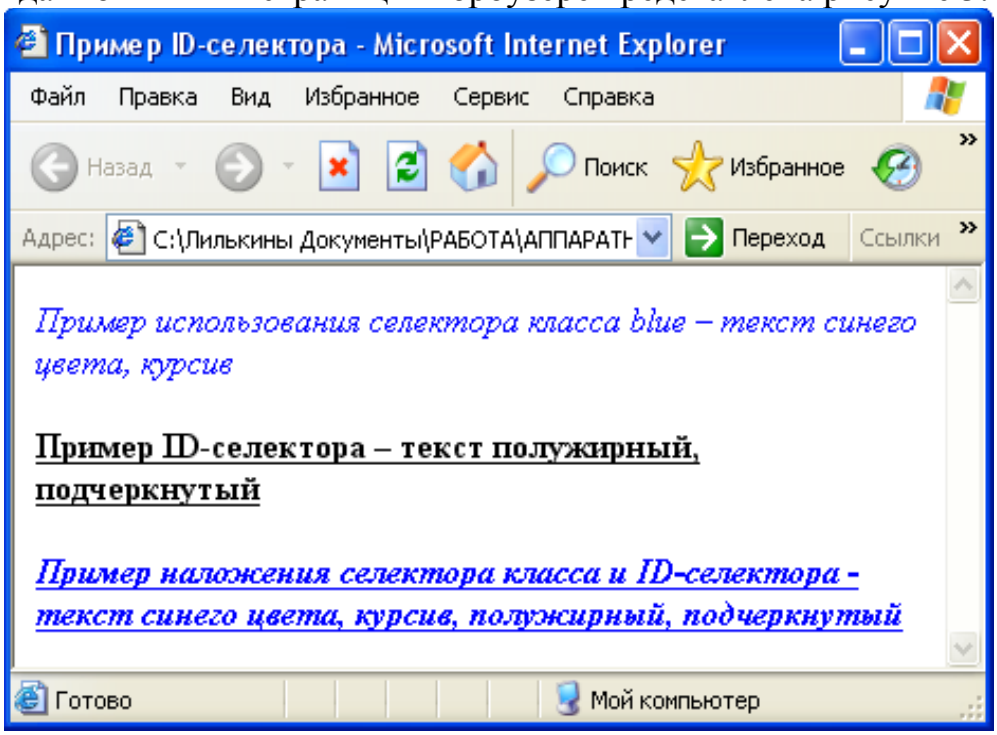

Рисунок 3 – Пример использования ID-селектора

Как видно из примера, атрибут ID может быть использован без указания класса (последний параграф примера) Тогда параграф будет обладать только свойствами ID “boldunderline” (в примере - жирный, подчеркнутый текст).

Наиболее часто идентификаторы используются для элементов форм, которые в спецификации HTML 4.0 имеют полную либо ограниченную

поддержку CSS (в зависимости от элемента). Вот пример назначения идентификатора и правил CSS таким элементам:

```
<HTML>
<HEAD>
<TITLE>Пример уникальных элементов форм</TITLE>
<STYLE>
input#green {color: green;}
input#red {color: red;}
</STYLE>
</HEAD>
<BODY>
<FORM>
<P>Текст, вводимый в это поле, будет отображен зеленым цветом:
<INPUT TYPE= “text” id=“green” name= “info1” size= “20”></P>
<BR><P>Текст, вводимый в это поле, будет отображен красным цветом:
<INPUT TYPE= “text” id= “red” name=“info2” size=“20”></P>
</FORM>
</BODY>
</HTML>
```

В броузере, когда пользователь введет текст, эта HTML-страница будет выглядеть следующим образом:

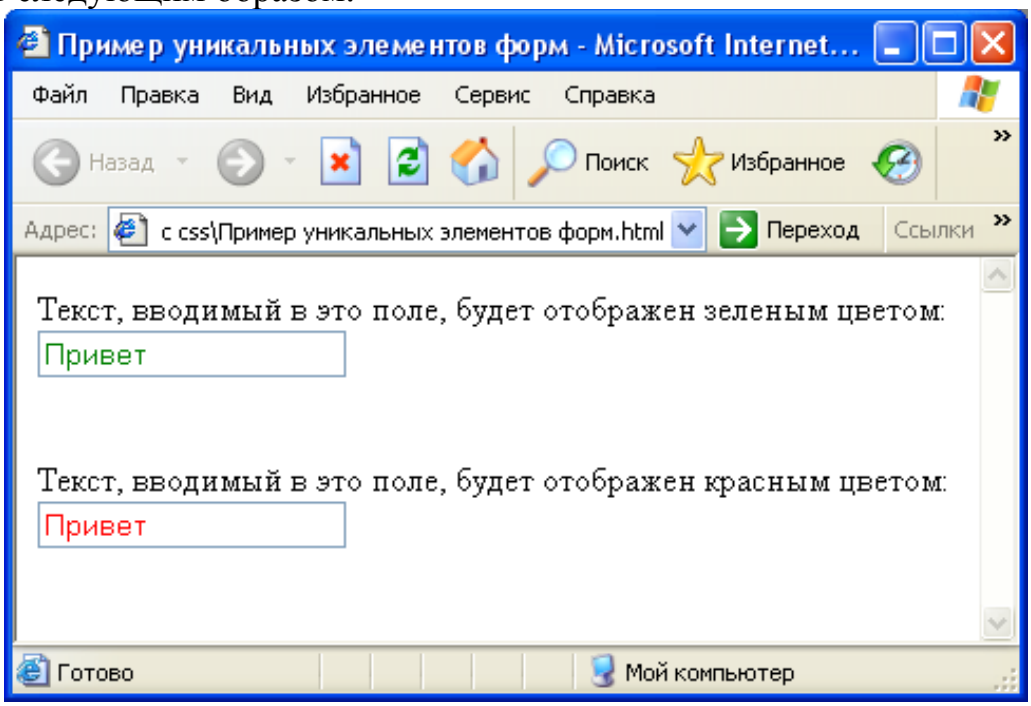

Рисунок 4 - Использование id для элементов форм

Аналогичным образом уникальные идентификаторы могут быть назначены любому количеству различных элементов страницы.

Это может быть полезно для обращения к элементу из программы на языке JavaScript и изменения стиля его отображения в ответ на действия пользователя, что позволяет создавать различные динамические эффекты на Web-странице.

2.4 Контекстные селекторы

Контекстные селекторы ***(Contextual Selectors)*** - это сочетания нескольких обыкновенных селекторов. Стиль задается только элементам в заданной последовательности в зависимости от каскадного порядка /8/.

Например, P EM {color: silver;}

В данном примере все элементы EM внутри элементов P будут иметь заданный стиль.

Для большей наглядности рассмотрим следующий пример. Допустим, создана таблица стилей, согласно которой все заголовки первого уровня на странице отображаются красным цветом на сером фоне, все абзацы - зеленым цветом на желтом фоне, а все выделенные посредством тега слова внутри абзацев - черным цветом на серебристом фоне. Код таблицы стилей, при этом выглядит следующим образом:

```
h1 {color: red;
background-color: gray;}
p {color: green;
background-color: yellow;}
em {color: black;
background-color: silver;}
```

При этом также требуется выделить какие-то слова в заголовке посредством того же тега , но необходимо выделить слова в заголовке бордовым цветом на сером фоне. Для этого использовались контекстные селекторы. Запись правила будет выглядеть следующим образом:

```
h1 em {
color: #CC0000;
background-color: gray;}
```

А вот пример кода HTML-страницы с использованием этого контекстного селектора:

```
<HTML>
<HEAD><TITLE>Пример контекстных селекторов</TITLE>
<STYLE>
h1 {color: red;
background-color: gray;}
p {color: green;
background-color: yellow;}
em {color: black;
background-color: silver;}
h1 em {color: #CC0000;
background-color: gray;}
</STYLE>
</HEAD>
<BODY>
<H1>Это - заголовок первого уровня с <em>выделенным</em>
словом</H1>
```

<P>А это - обычный абзац с выделенными словами</p>
</BODY>
</HTML>

Реализация данной HTML-страницы в броузере представлена на риснуке 5. При этом, на риснуке 6 представлена вариант той же HTML-страницы но без использования контекстных селекторов.

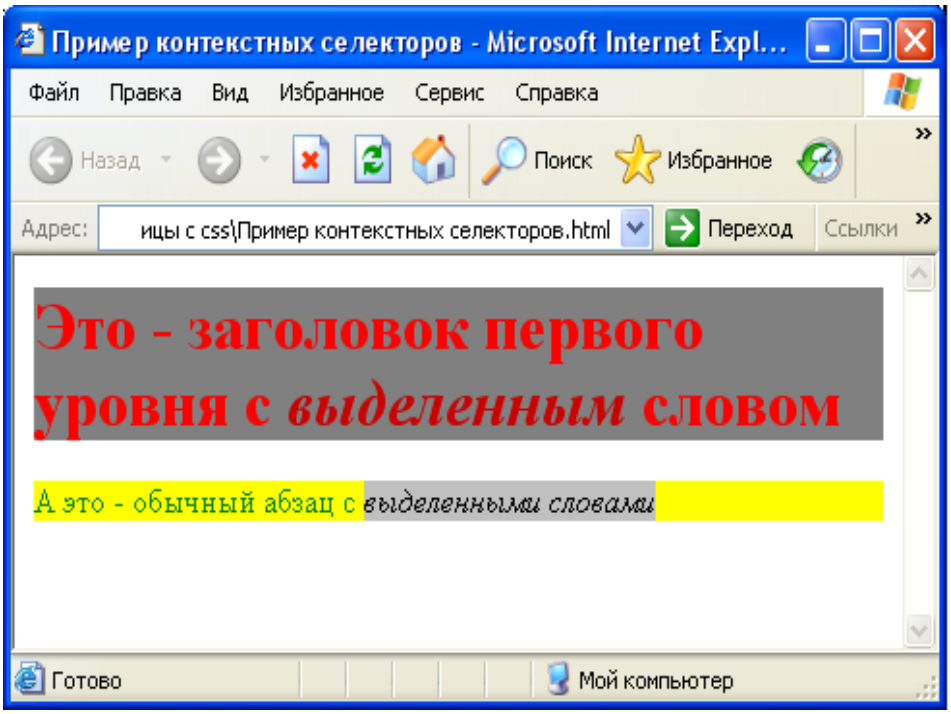

Рисунок 5 - Пример контекстных селекторов

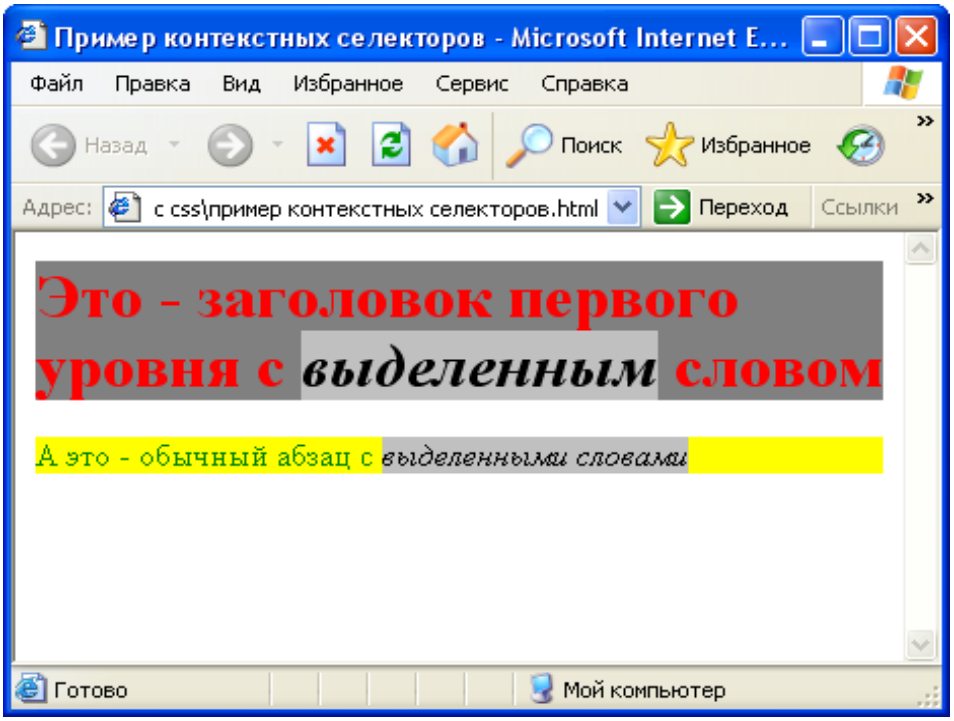

Рисунок 6 - Пример без использования контекстных селекторов

Рассмотрим более подробнее применение описанных выше селекторов в различных таблицах стилей.

3 Структура таблиц стилей

3.1 Внедренные таблицы стилей

Использование внедренных стилей мало чем отличается от использования обычных HTML-тегов. Они задают стиль только одному элементу документа при помощи атрибута STYLE в HTML теге.

** Пример -** обычный HTML-код.

** Пример **-код с использованием CSS.

Как можно заметить, код CSS получился больше чем HTML. Поэтому CSS следует использовать, только если необходимо задать определенному элементу свой индивидуальный стиль, существующий в классификации CSS и нереализованный в HTML. Другими словами, внедренные таблицы стилей следует использовать в том случае, когда данная таблица стилей используется только в данном HTML-документе.

Стиль можно задавать для всех HTML тегов. CSS позволяет переопределять стиль текста, заголовков, абзацев, рамок, осуществлять выравнивание и т.д.

Например, можно задать стиль элемента H1, который выравнивает заголовок по центру и задет ему рамку.

<H1 style=“border-width: 1; border: solid; text-align: center;”>
Этот заголовок имеет указанный стиль</H1>

Таблицы стилей могут содержать комментарии, позволяющие описать то или иное свойство. Текст, расположенный между символами /* и */ не обрабатывается броузером и не рассматривается как элемент таблицы стилей. Вложенные комментарии не допускаются. Пример:

```
<H1 style=
 “font-weight: bold;          /* жирность шрифта */
  font-size: 12pt;            /* размер шрифта */
  font-family: Helvetica;     /* название шрифта */
  font-variant: normal;       /* вариант шрифта */
  font-style: normal”         /* стиль шрифта */ >
```

Однако CSS позволяют назначить собственный стиль визуального представления не только любому тегу форматирования текста в HTML, но и тегу <body>. Если стиль задан для тега <body>, он наследуется всеми элементами (абзацами, заголовками и т.д.), помещенными внутри этого тега-контейнера, в случае отсутствия собственных стилей для этих элементов. Таким образом, уже не нужно прописывать теги font и атрибуты color, size и т.п. для каждого абзаца на странице - достаточно задать некий стиль для тега <body>, и все абзацы на странице будут отображены в соответствии с этим стилем.

Например, если необходимо, чтобы все абзацы отображались шрифтом Times New Roman размером 12 пунктов темно-зеленого цвета следует

прописать атрибут style тега <body>, присвоив ему соответствующее значение. Синтаксис такой:

<body style= “font-family: 'Times New Roman', serif; font-size: 12pt; color: darkgreen;”>

В приведенных примерах используется встраивание стиля непосредственно в тег документа - так называемый inline-стиль. Этот способ связывания CSS с HTML-файлом рекомендуется в единичных случаях - если данный стиль планируется применить только к одному элементу только на одной странице Web-сайта.

3.2 Внутренние таблицы стилей

Пока речь шла о задании только одного стиля для одного элемента. Для включения в документ внутренней таблицы стилей следует поместить в заголовок документа элемент STYLE, где и описать все параметры.

Синтаксис:

<HTML>
<HEAD>
<STYLE>
Тег {свойство 1: значение 1; свойство 2: значение2; свойство N: значение N;}
</STYLE>
</HEAD>
<BODY>
переопределение стиля…
</BODY>
</HTML>

Необходимо обратить внимание на использование тегов комментария (<!-- и -->) во внутренних таблицах стилей. Все содержимое контейнера <STYLE> находится между ними. Это позволяет скрыть информацию о стиле от броузеров, которые не распознают тег <STYLE> (в противном случае они могут принять правила за текст и вывести их на экран) /8/.

Приведем простой пример:

```
<HTML>
<HEAD><TITLE>Пример внутренней таблицы стилей</TITLE>
  <STYLE type= “text/css”>
  H1 {border-width: 1; border: solid; text-align: center}
</STYLE>
</HEAD>
<BODY>
<H1>Этот заголовок имеет указанный выше стиль</H1>
<H2>А этот заголовок не имеет указанный выше стиль</H2>
</BODY>
</HTML>
```

В броузере эта страница будет выглядеть следующим образом:

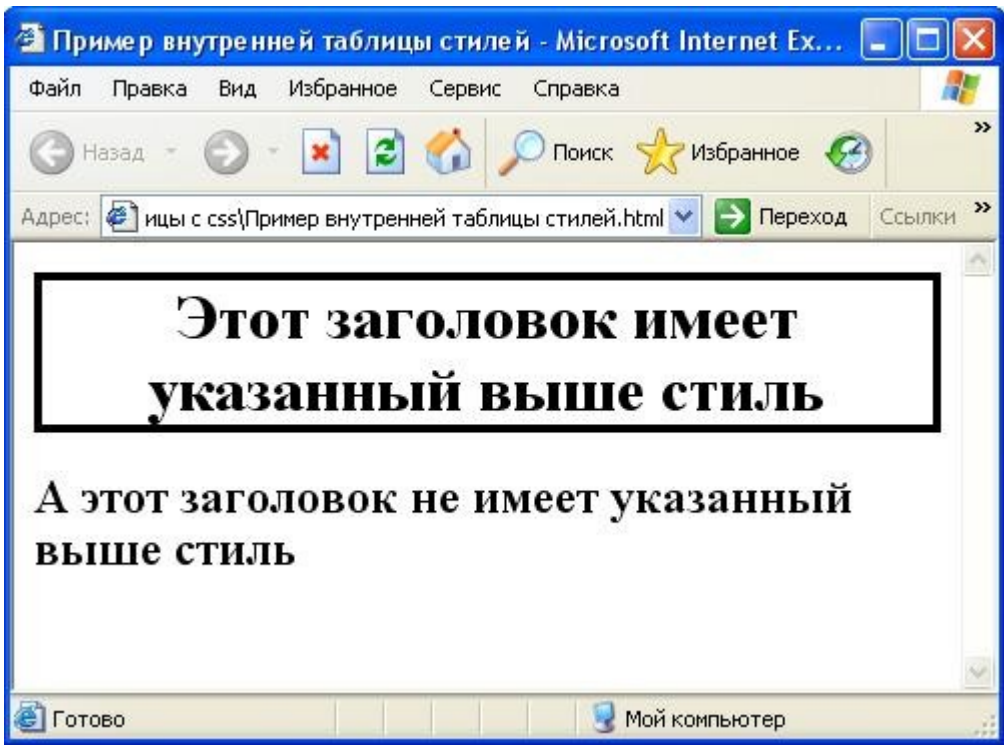

Рисунок 7 – Пример внутренней таблицы стилей

При этом, во внутренних таблицах стилей, также как и во внедренных можно задавать стиль тегу BODY. Например, установим белый цвет фона документа, цвет текста серый, семейство шрифта (гарнитура) Times, шрифт с засечками, размер шрифта 14 пунктов, отступ красной строки 10pt, выравнивание по ширине документа, отступ слева 20pt, темно-красный цвет заголовков с первого по третий уровень. Документ примет вид:

```
<HTML>
<HEAD>
<TITLE>Пример использования стилей</TITLE>
<STYLE TYPE= “text/css”>
BODY {background-color: white}
H1, H2, H3 {color: red}
P {color: gray;
font-family: Times, serif;
font-size: 14pt;
text-indent: 10pt;
text-align: justify;
padding-left: 20pt; }
</STYLE>
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

Помимо этого можно создавать несколько таблиц стилей в одном Web-документе. Например, если необходимо, чтобы все заголовки первого

уровня отображались шрифтом Arial полужирного начертания размером 16 пунктов зеленого цвета, а все заголовки второго уровня - шрифтом Helvetica размером 14 пунктов полужирного курсивного начертания желто-зеленого цвета необходимо в заголовке страницы (в любом месте между тегами <head> и </head>) указать таблицу стилей, в которой нужно прописать несколько правил сразу.

Для этого создается тег-контейнер таблицы стилей, начинающийся открывающим тегом <style> и заканчивающийся закрывающим тегом </style>. Внутри этого тега-контейнера можно задать любое количество правил CSS, состоящих из селектора (названия тега HTML, к которому будет применяться правило) и его определения (непосредственно набора средств форматирования), заключенного в фигурные скобки. Синтаксис для приведенного выше примера такой:

```
<HTML>
<HEAD>
<STYLE>
body { font-family: 'Times New Roman', serif;
font-size: 12pt;
color: darkgreen;}
h1 {font-family: Arial, sans-serif;
font-size: 16pt;
color: green;
font-weight: bold;}
h2 {font-family: Arial, sans-serif;
font-size: 14pt;
color: greenyellow;
font-weight: bold;
font-style: italic;}
</STYLE>
</HEAD>
<BODY>
…
</BODY>
</HTML>
```

Одной из особенностей языка CSS является то, что с его помощью можно создавать различные эффекты на HTML-странице без использования графики. Рассмотрим такой пример.

```
<HTML>
<HEAD>
<TITLE>Пример CSS
</TITLE>
<STYLE>
BODY {color: “black”; font-size: 16px; font-family: Arial}
.тень {color: #dbdbdb; text-align: right; weight: medium; margin-top: 0px;
font-size: 240px; line-height: 240px; font-family: Times}
```

```
.основа {color: red; weight: 900; margin-top: -270px; font-size:220px; line-height: 220 px; font-family: Times}
.слой1 {color: black; margin-top: -100 px; weight: medium; font-size: 45 px; line-height: 45 px; font-family: Arial}
.слой2 {color: green; margin-top: 0 px; weight: medium; font-size: 55 px; line-height: 25 px; font-family: Arial}
</STYLE>
</HEAD>
<BODY>
<P> Пример использования каскадного стиля вместо графики </P>
<CENTER>
<TABLE WIDTH=500 CELLPADING=0 CELLSPACING=0 BORDER=0>
<TR>
<TD ALIGN=CENTER VALIGN=TOP>
<DIV CLASS= “тень”>Это</DIV>
<DIV CLASS= “основа”>Это</DIV>
<DIV CLASS= “слой1”>сделано без всякой графики</DIV>
<DIV CLASS= “слой2”>используя только стили текста</DIV>
</TD>
</TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

В броузере эта HTML-страница будет выглядеть следующим образом:

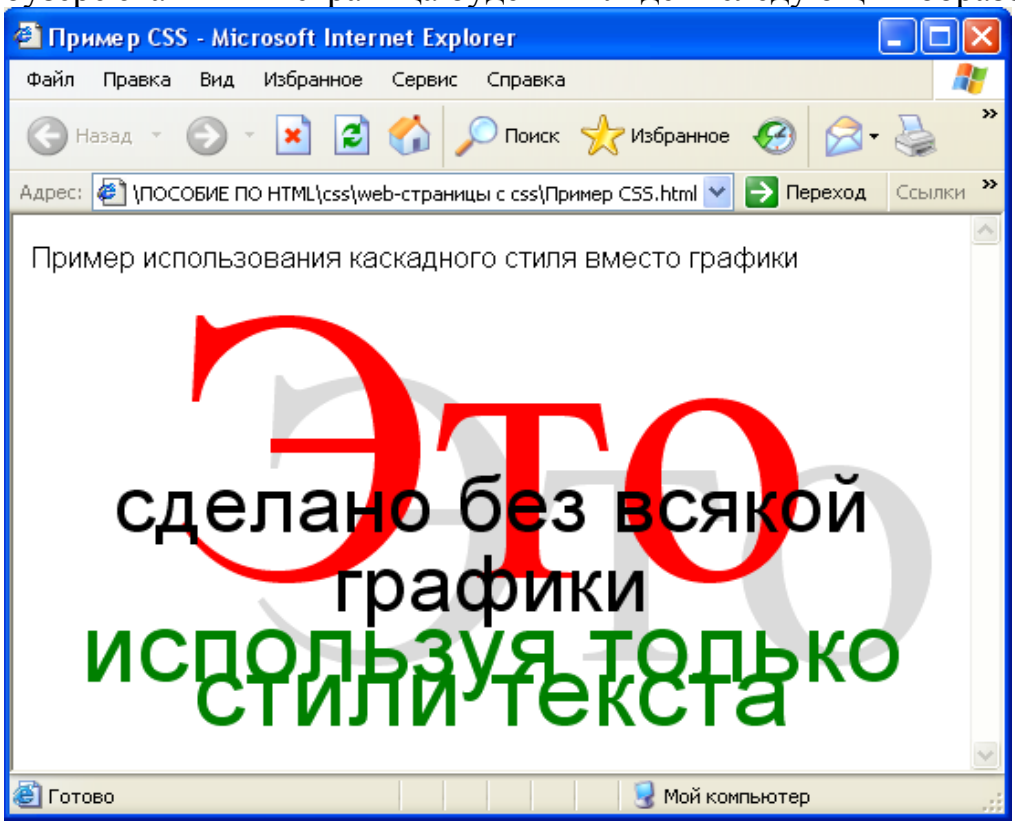

Рисунок 8 – Использование стилей текста место графики

В приведенном выше примере использован тег <DIV> и атрибут CLASS. Тег <DIV> применяется для задания части страницы (фрагмента документа). Он ничего не форматирует, а лишь помечает фрагмент текста, который рассматривается как единый объект. Атрибут CLASS позволяет сослаться на таблицу стилей и тем самым задать стиль представления текста, расположенного между тегами <DIV CLASS…> и </DIV>. При этом тексты определяются как бы на слоях, которые накладываются друг на друга. Сначала выводится слой “тень”, затем на него накладывается слой “основа”, а потом “слой1” и “слой2”. Очередность, в которой слои накладываются друг на друга, задает порядок следования фрагментов текста, помеченных тегом <DIV>. Также перекрытие обеспечивается применением отрицательных значений свойства margin-top (отступ сверху).

Рассмотрим еще один атрибут элемента <STYLE>, задающий тип устройства, для которого предназначена таблица стилей. Это атрибут media. В качестве его значений допустимы следующие:

- screen – для экранов компьютеров, не разделенных на страницы (в отличие от вывода на печать, где деление на страницы происходит обязательно).
- tty – для носителей с фиксированным размером символов, таких как телетайпы, терминалы и т.д.
- tv – для устройства типа телевизора (т.е. с низким разрешением и ограниченными возможностями прокрутки).
- projector – для устройства типа проекторов.
- handheld – для устройств типа карманных компьютеров или мобильных телефонов (небольшой экран, монохромный дисплей, растровая графика).
- print – для устройства вывода на печать, работающих постранично, или для документов, просматриваемых на экране в режиме предварительного просмотра печати.
- aural – для синтезаторов речи.
- all – для всех устройств.

По умолчанию используется screen, т.е изначально таблицы стилей ориентируются на экран компьютера. В качестве значения могут быть указаны одно устройство или их список, разделенный запятыми.

3.3 Внешние таблицы стилей

Помимо встраивания и внедрения для связи CSS и HTML используются способы импортирования и связывания таблиц стилей. Это, безусловно, наилучшие способы для придания единого стилевого оформления нескольким (или даже всем) страницам одного сайта. При этом вся таблица стилей хранится в одном файле (расширение файла должно быть стандартным - *.css).

Для подключения к документу внешней таблицы стилей (т. е. таблицы стилей, хранящейся в отдельном файле) следует поместить в заголовок документа элемент LINK, например:

<LINK rel="stylesheet" href="style.css" type="text/css">

В этой строке указывается, что связываемый файл является таблицей стилей (rel="stylesheet"), формат этого файла - *.css (type="text/css"), и находится он в той же директории, что и файл .html, либо имеет другой URL-адрес (href="style.css").

В элементе LINK можно дополнительно указать типы устройств, на которые распространяется данная таблица стилей, например:

<LINK rel="stylesheet" href= "style.css" type= "text/css" media="screen, print">

Внешние таблицы стилей следует использовать в том случае, когда несколько HTML-документов пользуются единой таблицей стилей.

Например, следующая таблица стилей CSS (хранящаяся в файле "special.css") устанавливает зеленый цвет текста абзаца и окружает его сплошной красной рамкой:

```
P.special {
color : green;
border: solid red;}
```

При этом теги <style> и </style> внутри файла таблицы стилей не используются - расширение .css явно указывает броузеру на то, что файл является таблицей стилей. Один такой файл может быть связан сразу с несколькими страницами (или импортирован сразу в несколько страниц).

Вызов в исходном HTML-документе таблицы стилей, расположенной в файле special.css:

```
<HTML>
<HEAD>
<LINK href="special.css" rel= "stylesheet" type= "text/css">
</HEAD>
<BODY>
<P class="special">В этом абзаце текст должен быть зеленым</P>
</BODY>
</HTML>
```

Приведем пример содержимого более сложного файла CSS (например, my.css):

```
body {
font-family: 'Times New Roman', serif;
font-size: 12 pt;
color: darkgreen;}
h1 {
font-family: Arial, sans-serif;
font-size: 16 pt;
color: green;
font-weight: bold;}
```

h2 {
font-family: Arial, sans-serif;
font-size: 14 pt;
color: greenyellow;
font-weight: bold;
font-style: italic;}

При этом необходимо отметить, что каскадные языки таблиц стилей, такие как CSS, позволяют использовать информацию о стиле из нескольких источников. Однако не все языки таблиц стилей поддерживают каскады. Чтобы определить каскад, авторы указывают последовательность элементов LINK и/или STYLE. Каскад информации таблиц стилей производится в порядке указания элементов в разделе HEAD.

В следующем примере определяются две альтернативные таблицы стилей с именем “compact”. Если пользователь выбирает стиль “compact”, агент пользователя должен применять обе внешние таблицы, а также постоянную таблицу “common.css”. Если пользователь выбирает стиль “bigprint”, применяется только альтернативная таблица ”bigprint.css” и постоянная таблица “common.css”.

<LINK rel= “alternate stylesheet” title= “compact” href= “small-base.css” type= “text/css”>

<LINK rel= “alternate stylesheet” title= “compact” href= “small-extras.css” type= “text/css”>

<LINK rel= “alternate stylesheet” title= “big print” href= “bigprint.css” type= “text/css”>

<LINK rel= “stylesheet” href=”common.css” type= “text/css”>

В заключении этого раздела, для более наглядного отображения отличий внутренних и внешних таблиц стилей рассмотрим следующий пример.

```
<HTML>
<HEAD>
<TITLE>Моя домашняя страница
</TITLE>
<STYLE type=“text/css”>
BODY {background : “00AADD”}
H1 {color: red}
P {color: “000088”}
</STYLE>
</HEAD>
<BODY>
<H1>Моя домашняя страница</H1>
<P>Добро пожаловать!</P>
</BODY>
</HTML>
```

Этот документ содержит элемент STYLE, содержащий два правила CSS: первое из них определяет, что цвет отображения элемента BODY должен быть красным, а второе - что цвет отображения элемента H1 должен

быть синим. Поскольку стиль отображения элемента P не задан, он унаследует стиль своего родительского элемента, в данном случае элемента BODY. В результате данный HTML-документ в броузере будет отображаться следующим образом.

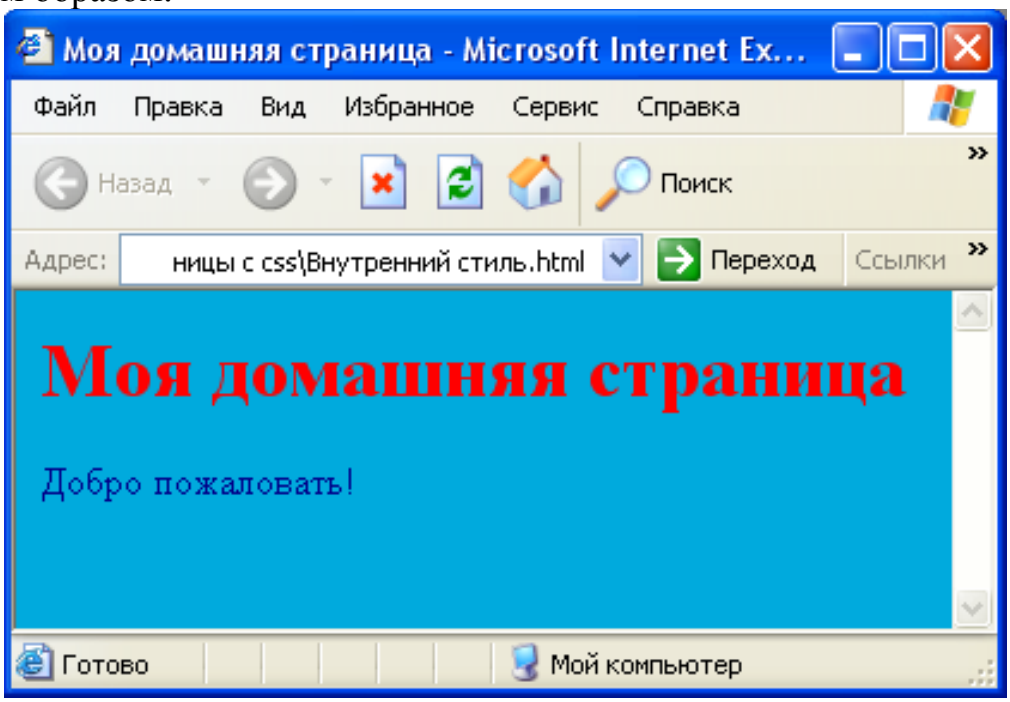

Рисунок 9 – Пример простого внутреннего стиля

Теперь для изменения стиля отображения этого документа достаточно менять только содержимое элемента STYLE, не внося никаких модификаций в остальной HTML-документ. Для большей гибкости можно создать текстовый файл STYLE.CSS и перенести в него описание стилей:

```
BODY {
background :“00AADD”}
H1 {color: red}
P {color: “000088”
}
```

HTML-документ после этого следует изменить следующим образом:

```
<HTML>
<HEAD>
<TITLE>Моя домашняя страница</TITLE>
<LINK rel= “stylesheet” href= “style.css” type= “text/css”>
</HEAD>
<BODY>
<H1>Моя домашняя страница</H1>
<P>Добро пожаловать!</P>
</BODY>
</HTML>
```

В дальнейшем для изменения стилей можно изменять только файл STYLE.CSS, не касаясь головного HTML-документа.

3.4 Наследование стилей

Термин “наследование” перекочевал в CSS из объектно-ориентированных языков. Некоторые свойства CSS могут наследоваться от элементов-предков. Предком в данном случае называется элемент, стоящий в дереве документа уровнем выше, чем исходный элемент. Непосредственным предком называется элемент, стоящим ровно на один уровень выше, чем в дереве. Аналогично определяется и непосредственный потомок – элемент, стоящий на уровень ниже в дереве элементов (рисунок 10).

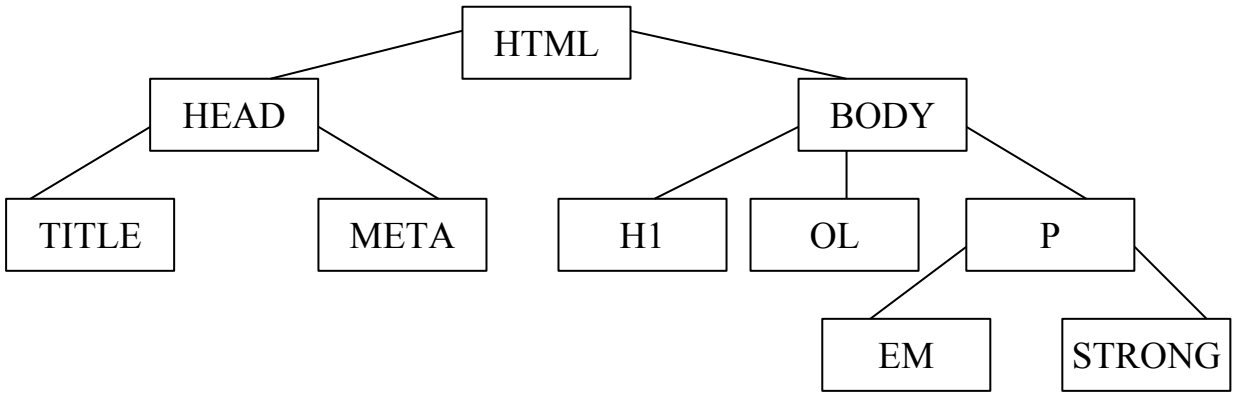

Рисунок 10 – Иерархия элементов HTML-документа

Например, в дереве, показанном на рисунке 10, элемент <LI> является потомком элементов <OL> и <BODY>, но непосредственным потомком только элемента <OL>. Элемент <OL>, в этом случае, называется непосредственным предком элемента <LI>. У элемента P два непосредственных потомка: EM и STRONG.

Для каждого свойства CSS указывается, является ли оно наследуемым или нет. Например, к наследуемым свойствам относится цвет, задаваемый ключевым словом color.

Воспользовавшись свойством наследования, можно задать общие для всего документа параметры оформления, указав их в самом старшем элементе документа, которыми могут быть или HTML или BODY.

BODY { color: brown;
font-size: 12 pt}

Тем самым был задан коричневый цвет для всего текста на странице и размер шрифта, равный 12 пунктам. Эти значения могут быть переопределены по более специфичным правилам, например на основе классов или идентификаторов.

Необходимо заметить, что правило с селектором BODY будет работать, даже если в коде HTML-страницы элемент BODY явно не указан, что вполне допустимо. В этом случае синтаксический анализатор сам его сгенерирует и все равно включит этот элемент в дерево документа.

Кроме этого, для каждого свойства может быть задано значение inherit, которое означает, что это свойство должно наследоваться.

P {color: inherit}

В результате цвет текста будет наследоваться от предка.

3.5 Каскадирование стилей

Любая таблица в HTML может иметь три источника возникновения. Во-первых, ее может создать автор HTML-документа, во-вторых, любой броузер содержит собственную таблицу стилей, которая задает значения по умолчанию для оформления всех элементов. Наконец, в-третьих, пользователь может сам создать таблицу стилей для представления страницы и использовать ее в том случае, если броузер предоставляет ему эту возможность.

Области их действия могут пересекаться, например, в каждой таблице может существовать правило, определяющее внешний вид текста и абзаца. В этом случае выбор приоритетного CSS-правила осуществляется в соответствии с принципами каскадирования. При наличии нескольких правил для одного и того же элемента преимущество имеет правило с наибольшим приоритетом.

По умолчанию правила в таблицах стилей, созданных разработчиками страницы, имеют больший приоритет, чем правила, созданные пользователем. И все правила разработчика и пользователя имеют больший приоритет, чем правила в таблице стилей броузера, используемой по умолчанию. Таким образом, наименее приоритетна – таблица стилей броузера, затем идет таблица стилей пользователя, и самой приоритетной является таблица стилей, созданная разработчиком.

Этот порядок можно просто изменить или же, меняя его, указать директиву **! important** для какого-то отдельного правила – тогда правило с такой директивой становится более приоритетным. Если, например, в таблице пользователя написано **P {color: blue ! important},** а в таблице разработчика написано **P{color: red},** то более приоритетным буде первое правило, т.е. цвет текста абзацев будет синим. Если же правило с директивой ! important будет существовать и в таблице пользователя, и в таблице разработчика, то приоритетнее будет таблица разработчика.

Таблицы стилей, подключаемые из внешних источников, также каскадируются, и их приоритетность определяется порядком подключения таблиц. Сначала применяется таблица стилей по умолчанию, затем ее правила переопределяются первой подключенной таблицей, потом подключается следующая, которая переопределяет правила предыдущей, и т.д. Поэтому, необходимо подключать таблицы стилей в порядке возрастания их важности, поскольку порядок здесь имеет значение /8/.

Рассмотрим каскадирование стилей на конкретном примере. В файле prog.css (внешняя таблица) заголовки H1 заданы красного цвета и выровнены по центру, гарнитура Times New Roman, значит любой документ, имеющий ссылку на style.css будет иметь заголовки такого же цвета, выравнивания и гарнитуры, если только в одном из этих HTML-документов в разделе STYLE не будет указан другой стиль (внутренняя таблица).

Например, для того же тега H1 будет задан синий цвет, гарнитура Arial, начертание полужирный, выравнивание по правому краю, размер шрифта 12 pt. Тогда все теги H1 этой HTML-страницы будут иметь заданный стиль.

Однако если, в этой же HTML-странице присвоить внедренный стиль для конкретного HTML-тега, например для H1 задать бордовый цвет, гарнитура Courier, начертание курсив, выравнивание по левому краю, размер 20 pt, то этому тегу присвоится именно этот стиль, так как приоритет у тега выше, а все остальные теги H1 останутся заданы прежним стилем.

Таким образом, чем ближе к документу и даже к тегу, тем больший приоритет имеют таблицы стилей. Самый высокий приоритет - у тега. Ниже представлен пример этой HTML-страницы.

```
<HTML>
<HEAD><TITLE>Пример каскадирования стилей</TITLE>
<LINK href="prog.css" rel= "stylesheet" type= "text/css">
<STYLE TYPE= "text/css">
H1 {color: blue;      font-family: Arial; font-weight: bold; font-size:12 pt;
text-align: right;}</STYLE>
</HEAD>
<BODY>
<H1>Этот заголовок будет синего цвета, гарнитуры Arial… </H1>
<H1 style="color: maroon; font-family: Courier; font-style: italic; font-size:
20 pt; text-align: left;">А этот заголовок будет бордового цвета… </H1>
<H1>Этот заголовок будет синего цвета, гарнитуры Arial… </H1 >
</BODY>
</HTML>
```

В броузере эта HTML-страница будет выглядеть следующим образом.

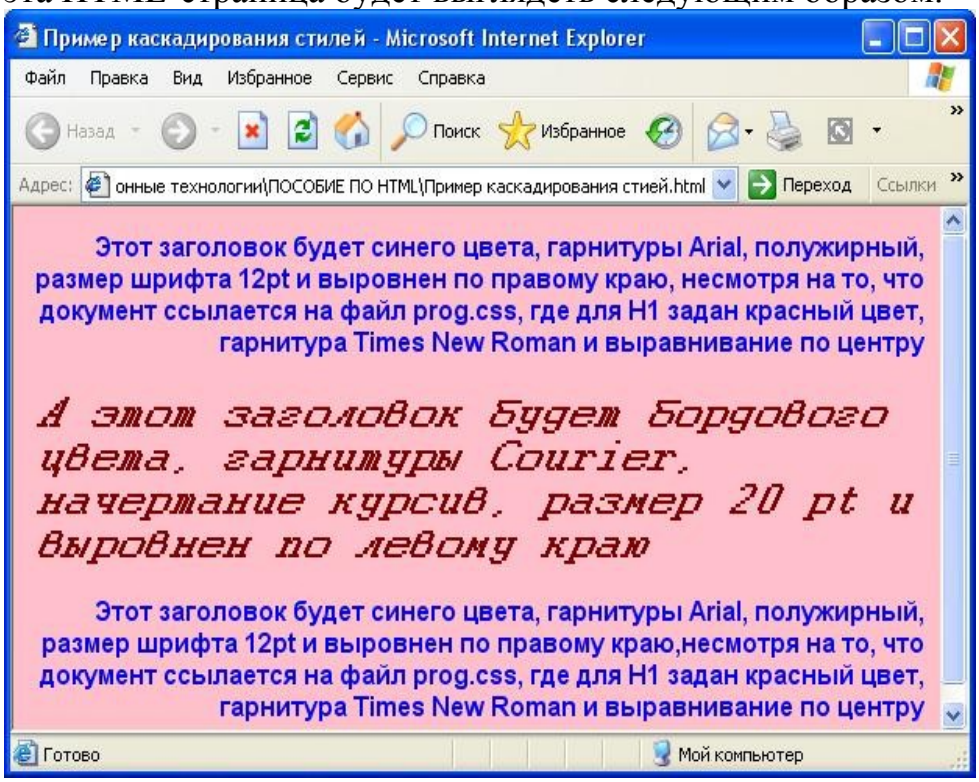

Рисунок 11 – Пример каскадирование стилей

4 Свойства CSS

Как уже было сказано выше, описание стиля включает в себя селектор тега, а также пару вида “свойство_стиля:значение”, то есть **селектор_тега { свойство_стиля:значение}.**

Все свойства CSS можно разделить на несколько групп:

- свойства шрифта;
- свойства цвета и фона;
- свойства текста;
- свойства списков;
- расположение элементов;
- свойства границ;
- свойства блоков /9/.

Рассмотрим наиболее распространенные свойства стилей и принимаемые ими значения.

4.1 Свойства шрифта

Font - задает параметры шрифта элемента страницы.

Заменяет атрибуты *font-family, font-height, font-size, font-style, font-variant* и *font-weight*. Значения этих атрибутов могут располагаться в любом порядке.

Синтаксис: font: {*font-family*} [{*font-height*}] [{*font-size*}] [{*font-style*}] [{*font-variant*}] [{*font-weight*}];

Значение по умолчанию - *normal normal normal medium normal “Times New Roman"*.

Альтернативный формат:

font: *caption*|*icon*|*menu*|*message-box*|*small-caption*|*status-bar*;

В этом случае доступны шесть предопределенных значений, задающие один из стандартных шрифтов, используемых в элементах интерфейса Windows:

font: *caption;* - шрифт заголовка кнопок, текстовых меток и т.п.;

font: *icon;* - шрифт подписей под пиктограммами;

font: *menu;* - шрифт пунктов меню;

font: *message-box;* - шрифт содержимого стандартных окон-предупреждений;

font: *small-caption;* - мелкий шрифт заголовков;

font: *status-bar;* - шрифт содержимого строки состояния.

Font-family - указывает имя шрифта или шрифтового семейства, используемого в элементе страницы.

Синтаксис: font-family: *{Имя шрифта}*|*serif*|*san-serif*|*cursive*|*fantasy*|*monospace*;

В качестве значения этого атрибута задается либо непосредственно имя нужного шрифта, либо одно из пяти предопределенных значений, задающих имя шрифтового семейства. Можно задавать одновременно несколько

шрифтов, разделив их имена запятыми; в этом случае Web-обозреватель сможет выбрать из них тот, который установлен на компьютере клиента. Если имя шрифта содержит пробелы, его следует взять в кавычки.

Пример, font-family: “Times New Roman”, sans-serif.

Font-weight - задает “жирность” шрифта, используемого в элементе страницы.

Синтаксис: font-weight: *normal|bold|bolder|lighter|100..900*;

“Жирность” может быть задана тремя способами. Во-первых, предопределенными значениями *normal* и *bold*, задающими обычное и жирное начертание соответственно. Во-вторых, относительными значениями *bolder* и *lighter*, делающими шрифт элемента страницы жирнее и светлее шрифта родителя.

И, в-третьих, одним из девяти значений от 100 до 900; здесь нормальному начертанию соответствует значение 400, а жирному - 700.

font-weight: *normal;* - обычный;

font-weight: *lighter;* - светлее;

font-weight: *bold;* - жирный;

font-weight: *bolder;* - жирнее;

font-weight: *от 100 до 900* - любое значение, кратное 100 (200,700).

Значение по умолчанию *normal*.

Font-size - задает размер шрифта, используемого в элементе страницы.

Синтаксис: font-size: *xx-small|x-small|small|medium|large|x-large|xx-large|larger|smaller|{Абсолютный размер}|{Относительный размер}%*;

Возможно задание либо абсолютного размера шрифта в одной из поддерживаемых CSS единиц измерения, либо как процент от размера шрифта родителя. Также доступны девять определенных значений.

font-size: *200%* - относительная величина (проценты);

font-size: *150px* - размер в пикселях;

font-size: *300pt* - размер в пунктах;

font-size: *{xx-small,small,medium,large,x-large,xx-large}* - задают один из семи размеров шрифтов, поддерживаемых HTML;

font-size: *{smaller,larger}* - задают размер шрифта, который на размер либо больше, либо меньше родительского шрифта соответственно.

Font-style - задает начертание шрифта.

Синтаксис: font-style: *normal|italic|oblique*;

font-style: *normal;* - задает обычный вид шрифта (используется по умолчанию);

font-style: *italic* - курсивное начертание;

font-style: *obligue* - наклонное начертание (легкий наклон нормального шрифта).

Font-variant - задает вид малых букв шрифта, используемого в элементе страницы.

Синтаксис: font-variant: *normal|small-caps*;

font-variant: *normal;* - задает обычный вид малых букв шрифта (используется по умолчанию);

font-variant: *small-caps;* - делает их такими же, как большие буквы, только меньшего размера /9/.

4.2 Свойства цвета и фона

Color - определяет цвет элемента.

Синтаксис: color: {Цвет};

Пример: color: red; “#00ff00”; либо color rgb (128,0,0).

Background - задает все свойства фона элемента страницы в один прием. Заменяет собой атрибуты *background-attachment, background-color, background-image, background-position* и *background-repeat*.

Синтаксис: background: [{*background-color*}] [{*background-image*}] [{*background-repeat*}] [{*background-attachment*}] [{background-position}];

Значения этих свойств могут располагаться в любом порядке.

Значение по умолчанию *transparent none repeat scroll 0% 0%.*

Background-color - задает фоновый цвет Web-страницы или ее элемента.

Синтаксис: background-color: {Цвет}|transparent;

Предопределенное значение transparent задает “прозрачный” фон. Оно же является значением по умолчанию.

Background-image - задает фоновый рисунок Web-страницы или ее элемента.

Синтаксис: background-image: *url ({URL-адрес файла рисунка})*|*none*;
Предопределенное значение none отключает фоновый рисунок. Оно же является значением по умолчанию.

Background-attachment - данный атрибут позволяет “зафиксировать” фоновый рисунок, чтобы он не прокручивался вместе с содержимым Web-страницы.

Синтаксис: background-attachment: *scroll*|*fixed*;

background-attachment: *scroll;* - фоновое изображение прокручивается вместе с содержимым страницы (значение по умолчанию);

background-attachment: *fixed;* - фоновое изображение фиксируется на одном месте и не прокручивается с содержимым страницы.

Применяется только для тега <BODY>.

Background-repeat - устанавливает порядок повторения фонового рисунка на Web-странице или ее элементе.

Иногда этот атрибут необходим, чтобы занять все свободное пространство, если фоновый рисунок слишком мал, чтобы покрыть пространство без повторения.

Синтаксис: background-repeat: *repeat*|*no-repeat*|*repeat-x*|*repeat-y*;

background-repeat: *repeat;* - размножает фоновое изображение во всех направлениях (значение по умолчанию);

background-repeat: *no-repeat;* - запрещает фоновому изображению повторяться;

background-repeat: *repeat-x;* - размножает фоновое изображение только по горизонтали;

background-repeat: *repeat-y;* - размножает фоновое изображение только по вертикали.

Background-position - задает местонахождение фонового рисунка. Это комбинированный атрибут, заменяющий атрибуты *background-position-x* и *background-position-y*.

Синтаксис: background-position: [{*background-position-x*}] [{*background-position-y*}];

Если задана только одна координата, то она считается горизонтальной, а для вертикальной принимается значение 50%. Значение по умолчанию 0% 0%.

Background-position-x - задает горизонтальную координату фонового рисунка.

Синтаксис: background-position-x: *{X}|{X}%|left|center|right*;

Координата может быть задана целым числом (абсолютная координата), процентом от соответствующего размера фонового рисунка (относительная координата) или предопределенным значением. Доступны три предопределенных значения: *left, center, right*, которые задают выравнивание фонового рисунка на странице по левому краю, по центру и по правому краю соответственно. Значение по умолчанию 0%.

Background-position-y - задает вертикальную координату фонового рисунка.

Синтаксис: background-position-y: *{Y}|{Y}%|top|center|bottom*;

Координата может быть задана целым числом (абсолютная координата), процентом от соответствующего размера фонового рисунка (относительная координата) или предопределенным значением. Доступны три предопределенных значения: *top, center, bottom*, которые задают выравнивание фонового рисунка на странице по верху, по центру и по низу страницы соответственно. Значение по умолчанию 0%.

Scrollbar-3d-light-color - задает цвет верхней и левой границ полосы прокрутки, ее бегунка и стрелок.

Синтаксис: scrollbar-3d-light-color: {*Цвет*};

Scrollbar-arrow-color - задает цвет стрелок на кнопках полосы прокрутки.

Синтаксис: scrollbar-arrow-color: {*Цвет*};

Scrollbar-base-color - задает цвет бегунка и кнопок-стрелок полосы прокрутки.

Синтаксис: scrollbar-base-color: {*Цвет*};

Scrollbar-darkshadow-color - задает цвет “тени”, отбрасываемой бегунком и кнопками прокрутки полосы прокрутки (цвет правых и нижних границей).

Синтаксис: scrollbar-darkshadow-color: {*Цвет*};

Scrollbar-face-color - задает цвет бегунка и кнопок прокрутки полосы прокрутки.

Синтаксис: scrollbar-face-color: {*Цвет*};

Scrollbar-highlight-color - задает цвет “освещенной” части бегунка и кнопок прокрутки полосы прокрутки (цвет левых и верхних их граней).

Синтаксис: scrollbar-highlight-color: {*Цвет*};

Scrollbar-shadow-color - задает цвет "неосвещенной" части бегунка и кнопок прокрутки полосы прокрутки (цвет правых и нижних их граней). Не путать с цветом "тени", задаваемым атрибутом *scroll-darkshadow-color*.

Синтаксис: scrollbar-shadow-color: {*Цвет*};

Scrollbar-track-color - задает цвет рабочей части полосы прокрутки, т.е. той ее части, по которой перемещается бегунок.

Синтаксис: scrollbar-track-color: {*Цвет*};

4.3 Свойства текста

Text-decoration - задает специальное оформление текста: подчеркнутый, зачеркнутый и т.п.

Синтаксис: text-decoration: *none*|*underline*|*overline*|*line-through*|*blink*;

text-decoration: *none;* - отменяет любое специальное оформление (значение по умолчанию для большинства тегов);

text-decoration: *underline;* - подчеркивает текст (значение по умолчанию для тегов <A>, <INS> и <U>);

text-decoration: *overline;* - черта сверху текста;

text-decoration: *line-through;* - зачеркивает текст (значение по умолчанию для тегов <DEL>, <S> и <STRIKE>);

text-decoration: *blink;* - мигание текста.

text-decoration: *uppercase* - все буквы становятся заглавными;

text-decoration: *lowercase* - все буквы становятся маленькими (прописными);

text-decoration: *capitalize* - каждое слово в строке начинается с большой буквы.

Text-underline-position - задает местонахождение линии подчеркивания: выше или ниже текста. Имеет смысл, если атрибут *text-decoration* равен *underline* или *overline*.

Синтаксис: text-underline-position: *below*|*above*;

text-underline-position: *below;* - помещает линию подчеркивания под текстом (значение по умолчанию);

text-underline-position: *above;* - над текстом (“надчеркивание”).

Text-align - определяет горизонтальное выравнивание текста.

Синтаксис: text-align: *left*|*right*|*center*|*justify*;

text-align: *left;* - выравнивание текста по левому краю (значение по умолчанию);

text-align: *right;* - выравнивание по правому краю;

text-align: *center;* - выравнивание по центру;

text-align: *justify;* - выравнивание по обоим краям (по ширине).

Text-align-last - задает горизонтальное выравнивание последней строки абзаца.

Синтаксис: text-align-last: *auto*|*inherit*|*left*|*right*|*center*|*justify*;

text-align: *auto;* - выравнивает последнюю строку абзаца так же, как и остальные строки (основываясь на значении атрибута *text-align*) (значение по умолчанию);

text-align: *inherit;* - так же, как выровнен текст родителя;

text-align: *left;* - выравнивание текста по левому краю;

text-align: *right;* - выравнивание по правому краю;

text-align: *center;* - выравнивание по центру;

text-align: *justify;* - выравнивание по обоим краям (по ширине).

Text-indent - устанавливает отступ красной строки.

Синтаксис: text-ident: *{Отступ}*|*{Отступ}%*;

Отступ может быть задан как абсолютной величиной, так и процентом от ширины родителя. Значение по умолчанию 0.

Text-height - интервал между строками текста.

Синтаксис: text-height: *{Интервал}*|*{Интервал}%*;

Интервал может быть задан как абсолютной величиной, так и процентом от ширины родителя.

Text-transform - задает преобразование регистра символов текста.

Синтаксис: text-transform: *none*|*capitalize*|*uppercase*|*lowercase*;

text-transform: *none;* - отключает любые преобразования регистра символов (значение по умолчанию);

text-transform: *capitalize;* - преобразует первую букву каждого слова текста в верхний регистр;

text-transform: *uppercase;* - преобразует все символы текста в верхний регистр;

text-transform: *lowercase;* - в нижний регистр.

Text-justify - задает тип текста по ширине. Значение атрибута *text-align* при этом должно быть равно *justify*.

Синтаксис: text-justify: *auto*|*newspaper*|*distribute*|*distribute-all-lines*|*distribute-center-last*|*inter-word*|*inter-ideograph*|*inter-cluster*|*kashida*;

text-justify: *auto;* - отдает управление выравниванием по ширине на усмотрение Web-обозревателя (используется по умолчанию);

text-justify: *newspaper;* - выравнивает строки, изменяя расстояние между словами и между символами;

text-justify: *distribute;* - аналогично *newspaper* и предназначено для азиатских языков (тайский и пр.);

text-justify: *distribute-all-lines;* - аналогично *distribute* за тем исключением, что последняя строка абзаца подвергается полному выравниванию. Предназначено для иероглифических языков;

text-justify: *distribute-center-last;* - не реализовано;

text-justify: *inter-word;* - выравнивает строки, изменяя только расстояние между словами;

text-justify: *inter-ideograph;* - выравнивает строки иероглифического текста, изменяя расстояния между словами и между иероглифами;

text-justify: *inter-cluster;* - выравнивает строки текста на азиатских языках, не содержащих пробелов между словами;

text-justify: *kashida;* - выравнивает строки текста на арабском языке, изменяя ширину самих символов.

Word-spacing - определяет дополнительное расстояние между словами в тексте.

Синтаксис: word-spacing: *normal|{величина}*;

Значение этого атрибута может быть задано либо абсолютной величиной в одной из поддерживаемых CSS единиц измерения, либо предопределенным значением *normal*, задающим стандартную величину расстояния между символами.

Word-wrap - устанавливает, будет ли строка, выходящая за границы элемента страницы и не содержащая пробелов, переноситься по словам.

Синтаксис: word-wrap: *normal|break-word*;

word-wrap: *normal;* - запрещает переносить строки, по словам (значение по умолчанию);

word-wrap: *break-word;* - разрешает.

Word-mode - задает направление строк текста: горизонтальное или вертикальное.

Синтаксис: word-mode: *lr-tb|tb-rl*;

word-mode: *lr-tb;* - задает обычное горизонтальное расположение строк текста; текст пишется слева направо и сверху вниз (значение по умолчанию);

word-mode: *tb-rl;* - поворачивает текст на 90° по часовой стрелке; при этом он будет писаться сверху вниз и справа налево.

Letter-spacing - определяет расстояние между символами в тексте.

Синтаксис: letter-spacing: *normal|{Величина}*;

Значение этого атрибута может быть задано либо абсолютной величиной в одной из поддерживаемых CSS единиц измерения, либо предопределенным значением *normal*, задающим стандартную величину расстояния между символами. Значение по умолчанию *normal*.

Line-height - задает вертикальное расстояние между строками текста.

Синтаксис: line-height: *normal|{Y}|{Y}%*;

Высота может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение *normal* задает стандартное расстояние. Значение по умолчанию *normal*.

Direction - задает порядок чтения текста: слева направо или справа налево.

Синтаксис: direction: *ltr|rtl|inherit*;

direction: *ltr;* - задает порядок чтения слева направо (значение по умолчанию);

direction: *rtl;* - справа налево;

direction: *inherit;* - заставляет наследовать порядок чтения у родителя.

Для документов, составленных на европейских языках, порядок чтения всегда слева направо (*ltr*).

White-space - задает, будут ли строки текста, содержащегося в элементе страницы, автоматически переноситься, если они не помещаются в нем по ширине.

Синтаксис: white-space: *normal*|*nowrap*|*pre*;

white-space: *normal;* - (значение по умолчанию) включает автоматический перенос длинных строк;

white-space: *nowrap;* - отключает автоматический перенос строк /9/.

4.4 Свойства списков

List-style - задает параметры маркера или номера позиции списка.

Заменяет атрибуты *list-style-image, list-style-position* и *list-style-type*. Значения этих атрибутов могут располагаться в любом порядке.

Синтаксис: list-style: *[{list-style-image}] [{list-style-position}] [{list-style-type}]*;

Значение по умолчанию *disk outside none*.

List-style-image - задает графическое изображение, отображаемое в качестве маркера позиции списка. Имеет приоритет над атрибутом *list-style-type*.

Синтаксис: list-style-image: *none*|*url({URL-адрес файла изображения})*;

Если задано предопределенное значение *none*, то стиль маркера берется из установок атрибута *list-style-type*, если он задан, или отображается маркер по умолчанию. Если задан Интернет-адрес файла изображения, то оно отображается в качестве маркера, перекрывая установки атрибута *list-style-type*. Значение по умолчанию *none*.

List-style-position - задает местонахождение маркера позиции списка: в тексте позиции или вне его.

Синтаксис: list-style-position: *outside*|*inside*;

Доступны два значения. Значение *outside* (по умолчанию) задает отображение маркера позиции списка вне текста позиции. Значение же *inside* заставляет Web-обозреватель отобразить маркер позиции в ее тексте в качестве первого символа.

List-style-type - задает тип маркера или номер позиции списка.

Синтаксис: list-style-type: *disc*|*circle*|*square*|*decimal*|*lower*|*roman*|*upper-roman*|*lower-alpha*|*upper-alpha*|*none*;

list-style-type: *disc;* - (значение по умолчанию) отображает сплошной кружок;

list-style-type: *circle;* - окружность;

list-style-type: *square;* - сплошной квадрат;

list-style-type: *decimal;* - нумерует позиции арабскими цифрами;

list-style-type: *lower-roman;* - малыми римскими;

list-style-type: *upper-roman;* - большими римскими;

list-style-type: *lower-alpha;* - помечает позиции малыми латинскими буквами;

list-style-type: *upper-alpha;* - большими латинскими;

list-style-type: *none;* - вообще убирает маркер или нумерацию.

Page-break-after - устанавливает, будет ли после текущего элемента при печати Web-страницы производиться прогон листа.

Синтаксис: page-break-after: *auto*|*always*|*empty string*;

page-break-after: *auto;* - передает управление размещением информации на листе операционной системе (значение по умолчанию);

page-break-after: *always;* - заставляет принтер прогнать лист после печати текущего элемента страницы;

page-break-after: *empty string;* - запрещает принтеру делать это в любом случае.

Page-break-before - устанавливает, будет ли перед текущим элементом при печати Web-страницы производиться прогон листа.

Синтаксис: page-break-before: *auto*|*always*|*empty string*;

page-break-before: *auto;* - передает управление размещением информации на листе операционной системе (значение по умолчанию);

page-break-before: *always;* - заставляет принтер прогнать лист перед печатью текущего элемента страницы;

Page-break-before: *empty string;* - запрещает принтеру делать это в любом случае.

4.5 Расположение элементов

Margin - задает ширину полей между элементами страницы и его соседями. Заменяет атрибуты *margin-top, margin-right, margin-bottom* и *margin-left*.

Синтаксис: margin: *{margin-top} [{margin-right}] [{margin-bottom}] [{margin-left}]*;

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем полям. Если задано два значения, первое относится к верхнему и нижнему полю, а второе - к левому и правому. Если задано три значения, то первое применяется к верхнему полю, второе - к левому и правому, третье - к нижнему.

Margin-top - задает верхнее поле между элементом страницы и его соседями сверху.

Синтаксис: margin-top: *auto*|*{Y}*|*{Y}%*;

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение *auto* заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задан атрибут *margin-bottom*, значение верхнего поля устанавливается равным значению нижнего поля. Значение по умолчанию *auto*.

Margin-right - задает правое поле между элементом страницы и его соседями справа.

Синтаксис: margin-right: *auto|{X}|{X}%*;

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение *auto* заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задан атрибут *margin-left*, значение правого поля устанавливается равным значению левого поля. Значение по умолчанию *auto*.

Margin-bottom - задает нижнее поле между элементом страницы и его соседями снизу.

Синтаксис: margin-bottom: *auto|{Y}|{Y}%*;

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение *auto* заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задан атрибут *margin-top*, значение нижнего поля устанавливается равным значению верхнего поля. Значение по умолчанию *auto*.

Margin-left - задает левое поле между элементом страницы и его соседями слева.

Синтаксис: margin-left: *auto|{X}|{X}%*;

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение *auto* заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задан атрибут *margin-right*, значение левого поля устанавливается равным значению правого поля. Значение по умолчанию *auto*.

Padding - задает отступ между элементом страницы и различными границами. Заменяет атрибуты *padding-top, padding-right, padding-bottom, padding-left*.

Padding-top - задает расстояние между элементом страницы и верхней границей.

Синтаксис: padding-top: *{Y}|{Y}%*;

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя.

Значение по умолчанию 0, для тега <TD> 1.

Padding-right - задает расстояние между элементом страницы и правой границей.

Синтаксис: padding-right: *{X}|{X}%*;

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя.

Значение по умолчанию 0, для тега <TD> 1.

Padding-bottom - задает отступ между элементом страницы и нижней границей.

Синтаксис: padding-bottom: *{Y}|{Y}%*;

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя.

Значение по умолчанию 0, для тега <TD> 1.

Padding-left - задает расстояние между элементом страницы и левой границей.

Синтаксис: padding-left: *{X}|{X}%*;

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Значение по умолчанию 0, для тега <TD> 1.

Width - задает ширину свободно позиционирования элемента.

Синтаксис: width: *auto|{X}|{X}%*;

Ширина может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение *auto* заставляет Web-обозреватель устанавливать ширину элемента самостоятельно.

Значение по умолчанию *auto*.

Height - задает высоту свободно позиционированного элемента.

Синтаксис: height: *auto|{X}|{X}%*;

Высота может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение *auto* заставляет Web-обозреватель устанавливать высоту элемента самостоятельно.

Значение по умолчанию *auto*.

Top - задает вертикальную позицию верхней границы свободно позиционированного элемента относительно родителя.

Синтаксис: top: *auto|{Y}|{Y}%*;

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение *auto* заставляет Web-обозреватель позиционировать элемент самостоятельно.

Значение по умолчанию *auto*.

Bottom - задает вертикальную позицию нижней границы свободно позиционированного элемента относительно родителя.

Синтаксис: bottom: *auto|{Y}|{Y}%*;

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение auto заставляет Web-обозреватель позиционировать элемент самостоятельно.

Значение по умолчанию *auto*.

Left - задает горизонтальную позицию левой границы свободно позиционированного элемента относительно родителя.

Синтаксис: left: *auto|{X}|{X}%*;

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение *auto* заставляет Web-обозреватель позиционировать элемент самостоятельно.

Значение по умолчанию *auto*.

Right - задает горизонтальную позицию правой границы свободно позиционированного элемента относительно родителя.

Синтаксис: right: *auto|{X}|{X}%*;

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение *auto* заставляет Web-обозреватель позиционировать элемент самостоятельно.

Значение по умолчанию *auto*.

Vertical-align - задает вертикальное выравнивание элемента страницы внутри родителя.

Синтаксис: vertical-align: *auto* | *baseline* | *sub* | *super* | *top* | *text-top* | *middle* | *bottom* | *text-bottom*;

vertical-align: *auto;* - выравнивает элемент страницы согласно значению атрибута *layout-flow*;

vertical-align: *baseline;* - задает выравнивание базовой линии элемента страницы по базовой линии родителя (используется по умолчанию);

vertical-align: *sub;* - превращает текст в нижний индекс;

vertical-align: *super;* - превращает текст в верхний индекс;

vertical-align: *top;* - выравнивает верх элемента страницы по самому верху родителя;

vertical-align: *text-top;* - выравнивает верх текста элемента страницы по верху текста родителя;

vertical-align: *middle;* - выравнивает центр элемента страницы по центру родителя;

vertical-align: *bottom;* - выравнивает низ элемента страницы по низу родителя;

vertical-align: *text-bottom;* - выравнивает низ текста элемента страницы по низу текста родителя.

Пример: IMG {vertical-align: top;}.

Overflow - задает поведение элемента страницы, если содержимое в нем не помещается.

Синтаксис: overflow: *visible* | *scroll* | *hidden* | *auto*;

overflow: *visible;* - заставляет элемент страницы расшириться так, чтобы все его содержимое было видно (значение по умолчанию);

overflow: *scroll;* - заставляет Web-обозреватель отобразить в элементе страницы полосы прокрутки, пользуясь которыми можно прокручивать его содержимое;

overflow: *hidden;* - скрывает все то, что выходит за пределы элемента страницы;

overflow: *auto;* - аналогично *scroll* за тем исключением, что полосы прокрутки отображаются только тогда, когда они реально необходимы (значение по умолчанию для <TEXTAREA>).

Overflow-x - задает поведение элемента страницы, если его ширина меньше ширины содержимого.

Синтаксис: overflow-x: *visible* | *scroll* | *hidden* | *auto*;

overflow-x: *visible;* - заставляет элемент страницы расшириться по горизонтали так, чтобы все его содержимое было видно (значение по умолчанию);

overflow-x: *scroll;* - заставляет Web-обозреватель отобразить в элементе страницы полосы прокрутки, пользуясь которыми можно прокручивать его содержимое;

overflow-x: *hidden;* - скрывает все то, что выходит за пределы элемента страницы;

overflow-x: *auto;* - аналогично *scroll* за тем исключением, что полосы прокрутки отображаются только тогда, когда они реально необходимы /9/.

Overflow-y - задает поведение элемента страницы, если его высота меньше высоты его содержимого.

Синтаксис: overflow-y: *visible* | *scroll* | *hidden* | *auto*;

overflow-y: *visible;* - заставляет элемент страницы расшириться по вертикали так, чтобы все его содержимое было видно (значение по умолчанию);

overflow-y: *scroll;* - заставляет Web-обозреватель отобразить в элементе страницы полосы прокрутки, пользуясь которыми можно прокручивать его содержимое;

overflow-y: *hidden;* - скрывает все то, что выходит за пределы элемента страницы;

overflow-y: *auto;* - аналогично *scroll* за тем исключением, что полосы прокрутки отобаржаются только тогда, когда они реально необходимы (значение по умолчанию для <TEXTAREA>).

Zoom - задает масштаб отображения элемента страницы.

Синтаксис: zoom: *normal|{Масштаб}|{Масштаб}%*;

Масштаб может быть задан как числом с плавающей точкой, обозначающим степень увеличения или уменьшения, так и процентной величиной. Предопределенное значение *normal* задает масштаб 1.0 или 100%.

Значение по умолчанию *normal*.

Table-layout - позволяет “зафиксировать” значения ширины ячеек таблицы.

Синтаксис: table-layout: *auto|fixed*;

table-layout: *auto;* - устанавливает ширину ячейки по ширине ее содержимого (значение по умолчанию);

table-layout: *fixed;* - использует для установки ширин ячеек значения атрибутов *WIDTH* или, если они не заданы, просто дает ячейкам равную ширину.

Другими словами, задание значения *auto* позволит точно “подогнать” значения ширин ячеек, но таблица при этом будет выводиться очень долго. Значение *fixed* этого атрибута позволит Web-обозревателю вывести таблицу значительно быстрее, но Web-дизайнер должен будет сам задать значения ширины ячеек. Применяется только для тега <TABLE>.

4.6 Свойства границ

Border - задает все свойства границ элемента страницы. Заменяет атрибуты *border-color, border-style* и *border-width*. Значения этих атрибутов могут располагаться в любом порядке.

Синтаксис: border: *[{border-color}] [{border-style}] [{border-width}]*;

Значение по умолчанию *medium none*.

Border-color - задает цвет всех границ элемента страницы. Заменяет атрибуты *border-top-color, border-right-color, border-bottom-color* и *border-left-color*.

Синтаксис: border-color: {*border-top-color*} [{*border-right-color*}] [{*border-bottom-color*}] [{*border-left-color*}];

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам. Если задано два значения, первое относится к верхней и нижней границам, а второе - к левой и правой. Если задано три значения, то первое применяется к верхней границе, второе - к левой и правой, третье - к нижней.

Border-color - задает цвет границ элемента страницы.

Синтаксис: border-color: *none*|*{Цвет}*;

Может быть задано либо цветное значение, либо *none*, обозначающее отсутствие границ.

Border-top-color - задает цвет верхней границы элемента страницы.

Синтаксис: border-top-color: *{Цвет}*;

Border-bottom-color - задает цвет нижней границы элемента страницы.

Синтаксис: border-bottom-color: *{Цвет}*;

Border-left-color - задает цвет левой границы элемента страницы.

Синтаксис: border-left-color: *{Цвет}*;

Border-right-color - задает цвет правой границы элемента страницы.

Синтаксис: border-right-color: *{Цвет}*;

Border-style - задает тип сразу всех границ элемента страницы в один прием. Заменяет атрибуты *border-top-style, border-right-style, border-bottom-style* и *border-left-style*.

Синтаксис: border-style: *none* | *dotted* | *dashed* | *solid* | *double* | *groove* | *ridge* | *inset* | *outset*;

border-style: *none;* - запрещает рисование границы (значение по умолчанию);

border-style: *dotted;* - рисует точечную линию;

border-style: *dashed;* - рисует штриховую линию;

border-style: *solid;* - рисует сплошную линию;

border-style: *double;* - рисует двойную сплошную линию;

border-style: *groove;* - рисует трехмерную вдавленную границу;

border-style: *ridge;* - рисует трехмерную выпуклую границу;

border-style: *inset;* - рисует трехмерную "ступеньку вверх";

border-style: *outset;* - рисует трехмерную "ступеньку вниз".

Border-top-style - задает тип верхней границы элемента страницы.

Синтаксис: border-top-style: *none* | *dotted* | *dashed* | *solid* | *double* | *groove* | *ridge* | *inset* | *outset*;

Border-top-style: *none;* - запрещает рисование границы (значение по умолчанию);

Border-top-style: *dotted;* - рисует точечную линию;

Border-top-style: *dashed;* - рисует штриховую линию;

Border-top-style: *solid;* - рисует сплошную линию;

Border-top-style: *double;* - рисует двойную сплошную линию;

Border-top-style: *groove;* - рисует трехмерную вдавленную границу;

Border-top-style: *ridge;* - рисует трехмерную выпуклую границу;

Border-top-style: *inset;* - рисует трехмерную "ступеньку вверх";

Border-top-style: *outset;* - рисует трехмерную "ступеньку вниз".

Border-bottom-style - задает тип нижней границы элемента страницы.

Синтаксис: border-bottom-style: *none* | *dotted* | *dashed* | *solid* | *double* | *groove* | *ridge* | *inset* | *outset*;

Border-bottom-style: *none;* - запрещает рисование границы (значение по умолчанию);

Border-bottom-style: *dotted;* - рисует точечную;

Border-bottom-style: *dashed;* - рисует штриховую линию;

Border-bottom-style: *solid;* - рисует сплошную линию;

Border-bottom-style: *double;* - рисует двойную сплошную линию;

Border-bottom-style: *groove;* - рисует трехмерную вдавленную границу;

Border-bottom-style: *ridge;* - рисует трехмерную выпуклую границу;

Border-bottom-style: *inset;* - рисует трехмерную "ступеньку вверх";

Border-bottom-style: *outset;* - рисует трехмерную "ступеньку вниз".

Border-left-style - задает тип левой границы элемента страницы.

Синтаксис: Border-left-style: none | dotted |dashed | solid | double | groove | ridge | inset | outset;

Border-left-style: none; - запрещает рисование границы (значение по умолчанию);

Border-left-style: dotted; - рисует точечную линию;

Border-left-style: dashed; - рисует штриховую линию;

Border-left-style: solid; - рисует сплошную линию;

Border-left-style: double; - рисует двойную сплошную линию;

Border-left-style: groove; - рисует трехмерную вдавленную границу;

Border-left-style: ridge; - рисует трехмерную выпуклую границу;

Border-left-style: inset; - рисует трехмерную "ступеньку вверх";

Border-left-style: outset; - рисует трехмерную "ступеньку вниз".

Border-right-style - задает тип правой границы элемента страницы.

Синтаксис: border-right-style: *none* | *dotted* | *dashed* | *solid* | *double* | *groove* | *ridge* | *inset*| *outset*;

Border-right-style: *none;* - запрещает рисование границы (значение по умолчанию);

Border-right-style: *dotted;* - рисует точечную линию;

Border-right-style: *dashed;* - рисует штриховую линию;

Border-right-style: *solid;* - рисует сплошную линию;

Border-right-style: *double;* - рисует двойную сплошную линию;

Border-right-style: *groove;* - рисует трехмерную вдавленную границу;

Border-right-style: *ridge;* - рисует трехмерную выпуклую границу;

Border-right-style: *inset;* - рисует трехмерную "ступеньку вверх";

Border-right-style: *outset;* - рисует трехмерную “ступеньку вниз”.

Border-width - задает толщину всех границ элемента страницы. Заменяет атрибуты *border-top-width, border-right-width, border-bottom-width* и *border-left-width*.

Синтаксис: border-width: {*border-top-width*} [{*border-right-width*}] [{*border-bottom-width*}] [{*border-left-width*}];

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам. Если задано два значения, первое относится в верхней и нижней границам, а второе - к левой и правой. Если задано три значения, то первое применяется к верхней границе, второе - к левой и правой, третье - к нижней.

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: *thin, medium, thick*, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию *medium*.

Border-top-width - задает толщину верхней границы элемента страницы.

Синтаксис: Border-top-width: *medium* | *thin* | *thick* | *{Толщина}*;

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: *thin, medium, thick*, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию *medium*.

Border-bottom-width - задает толщину нижней границы элемента страницы.

Синтаксис: Border-bottom-width: *medium* | *thin* | *thick* | *{Толщина}*;

Толщина может быть задана числовым значением. Также доступны три предопределенных значения*: thin, medium, thick*, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию *medium*.

Border-left-width - задает толщину левой границы элемента страницы.

Синтаксис: border-left-width: *medium* | *thin* | *thick* | *{Толщина}*;

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: *thin, medium, thick*, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию *medium*.

Border-right-width - задает толщину правой границы элемента страницы.

Синтаксис: border-right-width: *medium* | *thin* | *thick* | *{Толщина}*;

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: *thin, medium, thick*, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию *medium*.

Border-top - задает все свойства верхней границы элемента страницы за один прием. Заменяет атрибуты *border-top-color, border-top-style* и *border-top-width*. Значения этих атрибутов могут располагаться в любом порядке.

Синтаксис: border-top: [{*border-top-color*}] [{*border-top-style*}] [{*border-top-width*}]; Значение по умолчанию *medium none*.

Border-bottom - задает все свойства нижней границы элемента страницы за один прием. Заменяет атрибуты *border-bottom-color, border-bottom-style* и *border-bottom-width*. Значения этих атрибутов могут распологаться в любом порядке.

Синтаксис: border-bottom: [{*border-bottom-color*}] [{*border-bottom-style*}] [{*border-bottom-width*}]. Значение по умолчанию *medium none*.

Border-left - задает все свойства левой границы элемента страницы за один прием. Заменяет атрибуты *border-left-color, border-left-style* и *border-left-width*. Значения этих атрибутов могут располагаться в любом порядке.

Синтаксис: border-left: [{*border-left-color*}] [{*border-left-style*}] [{*border-left-width*}];

Значение по умолчанию *medium none*.

Border-right - задает все свойства правой границы элемента страницы за один прием. Заменяет атрибуты *border-right-color, border-right-style* и *border-right-width*. Значения этих атрибутов могут располагаться в любом порядке.

Синтаксис: border-right: [{*border-right-color*}] [{*border-right-style*}] [{*border-right-width*}]. Значение по умолчанию *medium none*.

Border-collapse - задает, будут ли границы ячеек и общая граница таблицы сливаться в одну или нет. Применяется только для тега <TABLE>.

Синтаксис: border-collapse: *separate*|*collapse*;

border-collapse: *separate;* - разделяет границу таблицы и границы ее ячеек (значение по умолчанию);

border-collapse: *collapse;* - объединяет их /9/.

4.7 Свойства блоков

Position - устанавливает, каким образом вычисляется положение элемента на плоскости экрана.

Синтаксис: position: *static*|*absolute*|*relative*;

position: *static;* - (значение по умолчанию) задает статическое позиционирование, при котором элемент страницы отображается внутри общего “потока” текста, т.е. не свободно. Значения атрибутов *bottom, left, right* и *top* при этом не принимаются Web-обозревателем во внимание;

position: *absolute;* - задает абсолютное свободное позиционирование. Значения атрибутов *bottom, left, right* и *top* при этом задают абсолютные координаты элемента страницы относительно родителя;

position: *relative;* - задает относительное свободное позиционирование. Значения атрибутов *bottom, left, right* и *top* при этом задают смещение координат элемента страницы от точки, в которой он был отображен, будь атрибут *position* установлен в *static*.

Z-index - задает порядок перекрытия свободно позиционированными объектами друг друга.

Синтаксис: z-index: *auto*|*{Порядок перекрытия}*;

Порядок перекрытия задается положительными или отрицательным целым числом. При этом элементы с большим значением этого атрибута будут перекрывать элементы с меньшим значением. Предопределенное значение *auto* задает порядок перекрытия по умолчанию, когда элементы, определенные в HTML-коде раньше, перекрываются заданными позже. Значение по умолчанию *auto*.

Visibility - позволяет элементу быть видимым или невидимым на странице.

Синтаксис: visibility: *inherit*|*visible*|*hidden*;

visibility: *inherit* - (значение по умолчанию) элемент виден, если его родительский элемент является видимым (наследует видимость);

visibility: *visible* - делает элемент страницы видимым;

visibility: *hidden* - невидимым.

Clip - задает прямоугольный фрагмент элемента страницы, который будет видим.

Синтаксис: clip: *auto*|*rect ({Верхняя граница} {Правая граница} {Нижняя граница} {Левая граница})*;

Предопределенное значение *auto* задает видимость всего элемента страницы. Оно же является значением по умолчанию. Для того чтобы ограничить видимую часть элемента страницы прямоугольным фрагментом, задаются четыре координаты границ этого прямоугольника, разделенные пробелами.

Float - определяет обтекание элемента другими слева или справа вместо помещения под ним.

Синтаксис: float: *none*|*left*|*right*;

float: *none;* - заставляет элемент страницы появляться там, где он задан (используется по умолчанию);

float: *left;* - принудительно выравнивает элемент страницы по левому краю;

float: *right;* - по правому краю.

Clear - задает поведение текста при "обтекании" им некоторых элементов страницы, таких как изображения.

Синтаксис: clear: *none*|*left*|*right*|*all*;

Атрибут задается для текста, а не для элемента страницы, который он будет “обтекать”.

clear: *none;* - разрешает тексту "обтекать" элемент страницы (значение по умолчанию);

clear: *left;* - запрещает тексту "обтекать" элемент страницы с левой стороны;

clear: *right;* - с правой стороны;

clear: *all;* - с обеих сторон.

Cursor - определяет форму курсора мыши, которую он принимает при наведении на элемент страницы.

Синтаксис: cursor: *auto* | *crosshair* | *default* | *hand* | *move* | * - *resize* | *text* | *wait* | *help*;

cursor: *auto* - (значение по умолчанию) заставляет Web-обозреватель самому определять нужную форму курсора мыши;

cursor: *crosshair* - крестообразный курсор;

cursor: *default* - курсор по умолчанию, обычно стрелка;

cursor: *hand* – “указывающий перст”;

cursor: *move* - стрелка, указывающая “на все четыре стороны”;

cursor: *e-resize*
cursor: *ne-resize*
cursor: *nw-resize*
cursor: *n-resize*
cursor: *se-resize*
cursor: *sw-resize*
cursor: *s-resize*
cursor: *w-resize*
cursor: *text* - текстовой курсор;
cursor: *wait* - "песочные часы", курсор ожидания;
cursor: *help* - стрелка с вопросительным знаком.

Display - определяет, как будет отображаться элемент.

Синтаксис: display: *inline* | *block* | *none* | *inline-block* | *list-item* | *table-header-group* | *table-footer-group*;

display: *inline;* - (значение по умолчанию) заставляет элемент страницы вести себя как встроенный. При этом он располагается внутри текста, его позиция и размеры рассчитываются согласно позиции и размерам текста.

display: *block;* - делает элемент страницы блочным. При этом его можно свободно позиционировать.

display: *none;* - делает элемент страницы невидимым. При этом страница отображается так, будто этого элемента вообще не существует;

display: *inline-block;* - аналогично *inline*, но содержимое страницы ведет себя как блочный элемент;

display: *list-item;* - аналогично *block*, но при этом элемент страницы считается позицией списка (ставится маркер);

display: *table-header-group;* - заставляет элемент страницы отображаться после верхнего заголовка таблицы и перед всеми строками данных (аналогично тегу <THEAD>);

display: *table-footer-group;* - заставляет элемент страницы отображаться перед основанием таблицы и после всех строк данных (аналогично тегу <TFOOT>);

4.8 Единицы измерения

Что касается параметров, то наиболее часто используемые параметры в CSS это значение размера и цвета. Задание цвета (color) в CSS, также как и в тегах HTML, возможно двумя способами: наименованием и численным значением.

При задании цвета численным значением используют либо шестнадцатеричное значение, либо тройку вида rgb (red, green, blue), где red, green и blue являются числами в диапазоне от 0 до 255, либо процентными величинами, определяющими яркость каждой составляющей цвета в RGB. Значения 255 или 100% указывают, что соответствующая составляющая цвета имеет максимальную яркость; значения 0 или 0% -- соответствующая

составляющая полностью отсутствует. Приведем простейший пример задания цвета в CSS.

H1 {color: #0000FF}
H1 {color: #00F}
H1 {color: rgb(27, 119, 207)}
H1 {color: rgb(50%,75%, 0%)}

При этом, во втором примере используется трехзначный синтаксис, при использовании которого три цифры преобразуются в шесть путем дублирования каждой из них.

При задании единиц длины (length) используют число, за которым без пробелов следует двухсимвольное обозначение единицы измерения.

Единицы измерения подразделяются на *абсолютные* и *относительные*. Абсолютные единицы измерения задают точный физический размер, а относительные - указывают размер относительно другого размера.

К абсолютным относят дюймы (in), сантиметры (cm), миллиметры (mm), точки (pt), пики (pc).

Однако, чаще используют относительные величины, такие как размер (em), высота строчных букв соответствующего шрифта (ex), пиксели (px).

Единицы em и ex основываются на размере шрифта того элемента, к которому относится данная декларация. При этом em задает размер шрифта, т. е. размер его наибольшей буквы (обычно это буква 'M', отсюда аббревиатура em), а ex - высоту строчных букв шрифта (обычно это высота буквы 'x', отсюда английское название x-height и аббревиатура ex). Единица px основана на размере пикселя устройства отображения (обычно это дисплей). Пиксель - это точка дисплея и ее размер зависит как от физических размеров экрана, так и от его разрешения /9/.

H1 { margin: 0.5 em; }
H1 { text-indent: 1ex; }
H3 { font-size: 12 px; }
H1 { margin: 0.5in }
H2 { line-height: 3cm }
H3 { word-spacing: 4mm }
H4 { font-size: 12pt }
H4 { font-size: 1pc }
P {font-size: 12pt}
H3 {text-indent: 20em}

Помимо перечисленных значений некоторые значения могут быть заданы в процентах (*percent*). При этом они вычисляются как процент от другого свойства элемента, обычно его размера, например,

P {line-height: 120%;}.
H4 { font-size: 100% }
H2 { font-size: 1% }
P {font-size: 12%}
H3 {text-indent: 20%}

5 Правила форматирования блоковых элементов страницы

Как известно, все элементы страниц языка HTML можно подразделить на две группы - блоковые и строчные. Блочные элементы создают визуально самостоятельную структурную единицу – блок. Примером блочных элементов могут быть элементы H1-H6, P, DIV, они отделяются от других элементов абзацными отступами. Строчные элементы не создают визуальной самостоятельной структурной единицы и выводятся линейной строкой, например: I, B, U, S.

Большую часть свойств CSS можно применить и к тем, и к другим элементам. Это такие свойства, как цвет текста и начертание шрифта, цвет фона и т.п. Строчные элементы, так же, как и блоковые, поддерживают классы и идентификаторы. Но есть отдельная группа свойств, применимых только к блоковым элементам.

Каждый блок обязательно имеет информационную часть, или содержимое, которым может быть текст, изображение или другая информация. Эта часть блока называется его контентом или содержимым.

Вокруг области контента могут быть пустые, не занятые содержимым области, называемые полями (padding). При наличии полей определенного размера текст не примыкает вплотную к границам блока, имеются не занятые текстом области вдоль краев документа /10/.

Непосредственно за полями проходит граница блока (border), которая может иметь определенную толщину и стиль линий. Ширина блока может быть произвольной – от нулевой до произвольно заданной в единицах измерения длины. Стиль может быть различным, от простой линии до объемных вариантов. Кроме того, граница может иметь произвольный цвет.

Также блок может иметь отступы (margin), это дополнительное свободное пространство вокруг границы блока. Границы и отступы не входят в ширину блока. Графическая модель блокового элемента страницы выглядит следующим образом:

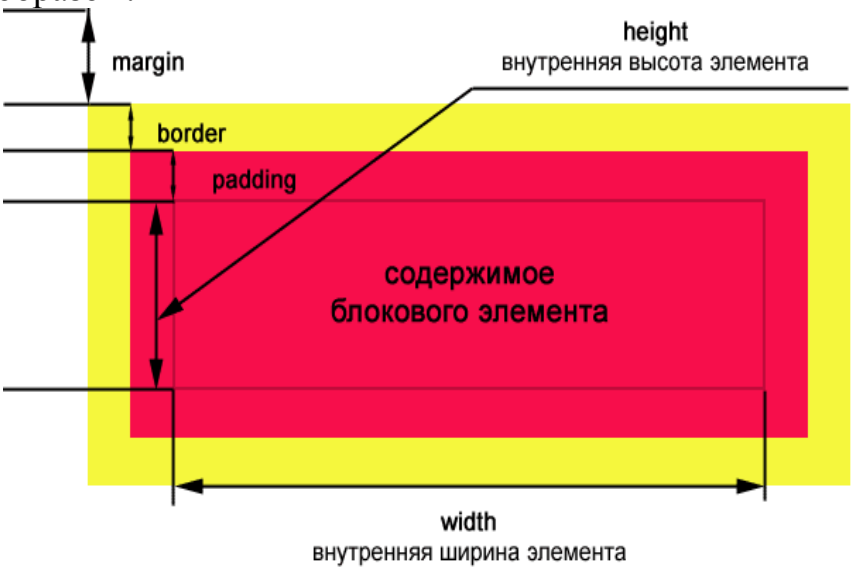

Рисунок 12 - Графическая модель блокового элемента страницы

Все поля и границы делятся соответственно на верхние, нижние, правые и левые. Для каждого из них можно задать значения независимо от других.

Фон для различных областей блока задает следующим образом:

- Область содержимого: фон задается свойством background элемента, порождающего данный.
- Поля: фон задается свойством background, порождающего блок.
- Границы: вид границ определяется свойствами границ элемента, порождающего блок. Фона они не имеют, чаще всего это линии определенного стиля, цвета и толщины.
- Отступы: область всегда прозрачна, соответственно сквозь нее может просвечивать фон родительских элементов. Таким образом, отступы, как бы наследуют фон от родительского элемента.

5.1 Поля

Поля - это специальные области, позволяющие задать некое свободное, никем не занятое пространство вокруг границы блока. Регулируя величину полей, можно управлять расстоянием между блоками, например, уменьшить или увеличить абзацный отступ между абзацами.

Свойства **margin-top, margin-right, margin-bottom, margin-left** определяют величину полей блока:

- margin-top – задает ширину верхнего поля;
- margin-right – задает ширину правого поля;
- margin-bottom – задает ширину нижнего поля;
- margin-left – задает ширину левого поля.

Например,

```
BODY {
margin-top: 0 px;
margin-right: 10%;
margin-bottom: 0 px;
margin-left 10%}
```

Свойство **margin** – стенографическое свойство, используется для краткой записи значений свойств margin-top, margin-right, margin-bottom, margin-left. Свойство Margin может иметь одно, два, три или четыре значения.

Если задано только одно значение, это означает, что величины полей со всех сторон одинаковые:

```
P {
Margin: 1 em}
```

В данном случае для текста абзаца будет установлено одинаковое значение для всех четырех полей, равное высоте шрифта.

Если заданы два значения, то первое относится к верхнему и нижнему полям, второе - к правому и левому, при этом значения каждой пары будут равны. Например,

DIV {

Margin: 10 px 5 px}

В данном примере для контейнеров DIV будут установлены поля по 10 пикселей сверху и по 5 пикселей слева и справа.

Если задано три значения, то первое применяется к верхнему полю, второе – к левому и правому, а третье – к нижнему. Например,

BODY {

Margin: 0 px 15 px 5px}

В этом примере страница не имеет верхнего поля, слева и справа поля одинаковы – по 15 пикселей, поле снизу равно 5 пикселей.

Если заданы все четыре значения, то они соответственно будут отвечать верхнему, правому, нижнему и левому полям. Иначе говоря, задание размеров начинается с верхнего поля и далее по часовой стрелке, заканчивая левым полем. Таким образом, следующие две записи будут эквивалентными:

P {margin-top: 0.5 em;
Margin-right: 0.5 em;
Margin-bottom: 0.5 em;
Margin-left: 1 em;
}

и

P{
Margin: 0.5 em 0.5 em 0.5em 1 em}

5.2 Отступы

Отступы позволяют отделить основное содержимое блока от границы таким образом, чтобы граница располагалась на некотором расстоянии от содержимого.

Свойства padding-top, padding-right, padding-bottom, padding-left – индивидуальные свойства, позволяющие задать размеры отступов для каждой стороны в отдельности:

- padding-top – ширина верхнего поля;
- padding-right - ширина правого поля;
- padding-bottom – ширина нижнего поля;
- padding-left – ширина левого поля.

Свойство **padding –** стенографическое свойство, используется для краткой записи значения свойств padding-top, padding-right, padding-bottom и padding-left. Свойство padding может иметь одно, два, три или четыре значения. Значения присваиваются также как для полей.

Рассмотрим пример. В стиле элемента P укажим цвет фона блока светло-желтый, вокруг блока граница толщиной 1 пиксель, размеры отступов

по 0.5 em сверху и снизу и по 3 em слева и справа. Первый абзац оформляется таким стилем, для второго абзаца значение свойства padding переопределяется и устанавливается равным нулю. В результате, согласно правилам каскадирования, отступы второго абзаца должны отсутствовать.

```
<HTML>
<HEAD>
<TITLE> Отступы в блоках</TITLE>
</HEAD>
<STYLE>
P {
background-color: #FFC;
border-width: 2 px;
border-style: ridge;
padding: 0.5 em 3 em;}
</STYLE>
<BODY>
<P>На примере этого абзаца рассмотрим, как работают отступы</P>
<P style="padding: 0 px">А вот пример абзаца без отступов</P>
</BODY>
</HTML>
```

В окне броузера этот HTML-страница будет выглядеть следующим образом.

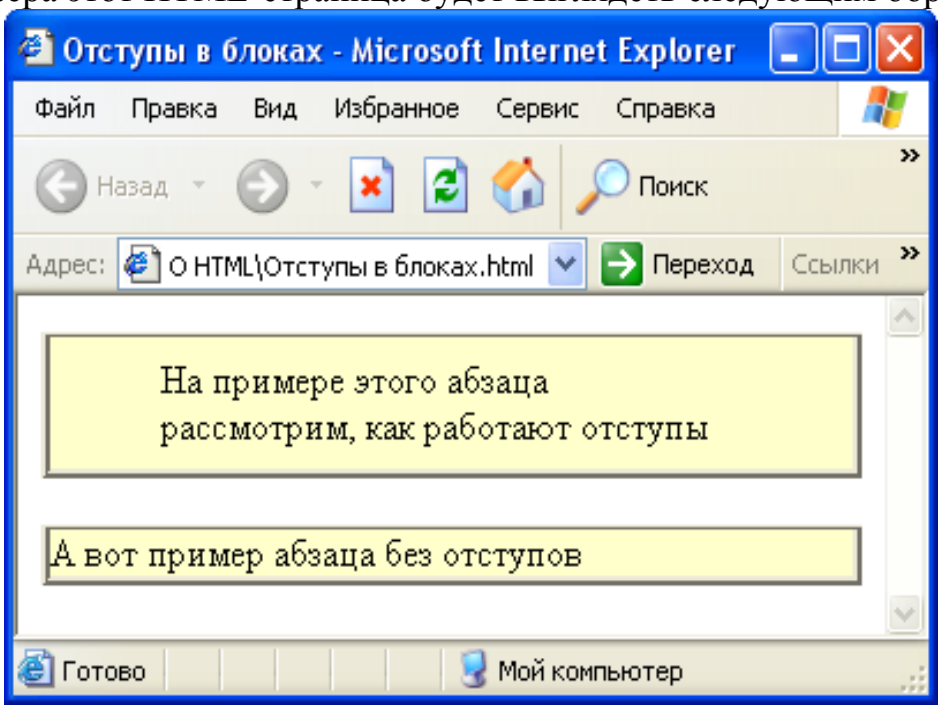

Рисунок 13 – Отступы в блоках

Различные отступы для отдельных сторон блока можно задавать при помощи именных свойств. Например,

```
P {
Padding-top: 0.5 em;
Padding-right: 10 px;
Padding-bottom: 3%;
```

padding-left: 1 em;}

Задавать различные размеры для отступов можно и при помощи стенографического свойства padding. Например,

P {

Padding: 0.5 em 10 px 3% 1 em}

Порядок задания значений отступов такой же, как и для полей: начиная с верхнего отступа, далее по часовой стрелке правый и нижний, последним задается левый отступ. Таки образом, два приведенных выше правила будут эквивалентны, но второе существенно короче по записи.

Рассмотрим пример отличия между отступами и полями, т.е между свойствами padding и margin. Для этого рассмотрим два класса:

P_mar {
Background-color: #FFC;
border-width: 2 px;
border-style: ridge;
margin: 2 em }
P_pad {
Background-color: #FFC;
border-width: 2 px;
border-style: ridge;
padding: 2 em }

Оформим два различных абзаца при помощи этих классов:

<P class= “p_mar”>Для этого абзаца используется класс “p_mar”, для него заданы поля размером 2 em</P>

<P class= “p_pad”>Для этого абзаца используется класс “p_pad”, для него заданы отступы размером 2 em ></P>

В первом случае, когда используются поля, “пустые” места расположены за пределами границы блока, внутри блока отступов нет. Во втором случае, когда используются отступы, “пустоты” расположены внутри границы блока.

Полный код HTML-страницы представлен ниже.

```
<HTML>
<HEAD>
<TITLE> Отступы в блоках</TITLE></HEAD>
<STYLE>
P_mar {
Background-color: #FFC;
border-width: 2 px;
border-style: ridge;
margin: 2 em }
P_pad {
Background-color: #FFC;
border-width: 2 px;
border-style: ridge;
padding: 2 em }
```

</STYLE>
<BODY>
<P class= “p_mar”>Для этого абзаца используется класс “p_mar”, для него заданы поля размером 2 em</P>
<P class= “p_pad”>Для этого абзаца используется класс “p_pad”, для него заданы отступы размером 2 em </P>
</BODY>
</HTML>

В броузере эта страница будет выглядеть следующим образом.

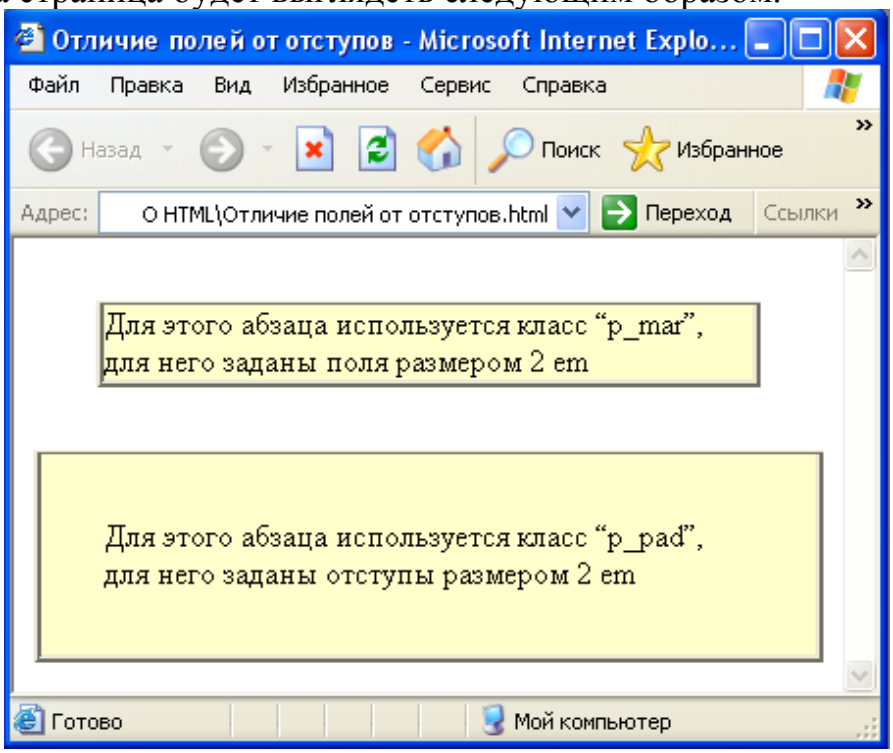

Рисунок 14 – Отличие полей от отступов

На рисунке хорошо видно и второе различие между отступами и полями. Как видно из рисунка, сквозь прозрачный фон полей просвечивает белый фон основной страницы. Фон отступов такой же, как и у всего блока - светло-желтый.

Помимо рассмотренных выше двух различий между отступами и полями имеется третье различие. Размеры полей могут занимать отрицательные значения, а размеры отступов только положительные.

Рассмотрим пример использования заголовка первого уровня, следом за ним идет текст некоего абзаца:

<HTML>
<HEAD><TITLE>Отрицательные значения полей</TITLE>
</HEAD>
<BODY>
<H1>Заголовок</H1>
<P>Абзац текста, на который с помощью задания отрицательных значений для полей, попробуем “надвинуть заголовок.”</P>
</BODY>

</HTML>

Изначально эта HTML-страница в броузере будет выглядеть следующим образом.

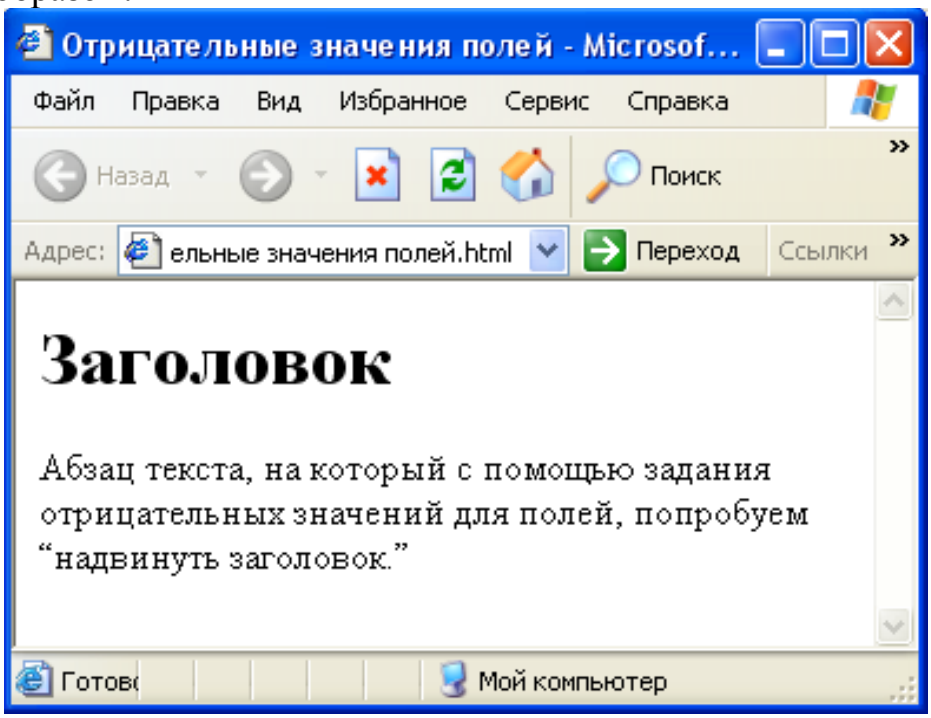

Рисунок 15 – Начальный вид заголовка и абзаца

Теперь, если для заголовка прописать стиль с использованием свойства margin и задать ему отрицательное значение, то можно надвинуть заголовок на текст, сократив расстояние между блоками. Используем, например, отрицательное значение свойства margin-bottom.

H1 { margin-bottom: - 1em}

После добавления этой строчки вид окна броузера изменится, как показано на рисунке 16.

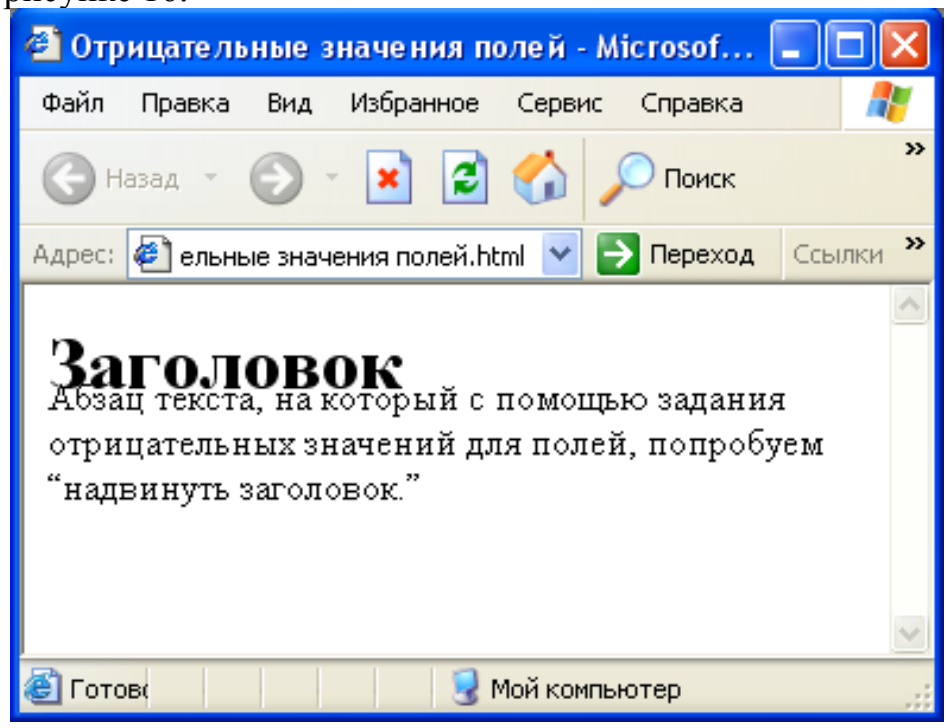

Рисунок 16 – Задано отрицательное значение для поля абзаца

За счет отрицательного значения нижнего поля {margin-bottom: -1 em} добились сокращения расстояния между блоками H1 и P на 1 em, в результате чего и произошло наслаивание заголовка на текст абзаца.

5.3 Границы

Границы блоков, так же как отступы и поля, делятся на четыре сегмента: верхний, правый, нижний и левый. Каждый отдельный сегмент границы блока может иметь свои характеристики: толщину (border-width), цвет (border-color), тип линий (border-style). Для задания характеристик могут использоваться отдельные свойства или стенографические свойства, позволяющие записывать правила в более краткой форме.

Свойства **border-top-width, border-right-width, border-bottom-width, border-left-width** определяют толщину границ блоков для верхнего, правого, нижнего и левого сегментов соответственно. Значениями могут быть произвольные единицы измерения либо ключевые слова:

- thin – тонкая граница;
- medium – граница средней толщины;
- thick – толстая граница.

Например, **P {**

Border-top-width: 1 px;

Border-right-width: medium;

Border-bottom-width: 1 px;

Border-left-width: thin;}

Свойство **border-width** стенографического типа задает значения свойств border-top-width, border-right-width, border-bottom-width, border-left-width. Свойство border-width может иметь одно, два, три или четыре значения. Если задать одно значение, то это означает, что толщина всех сегментов границы вокруг блока одинакова. Например,

P {border-width: 1 px }

Текст абзаца будет обрамлен рамкой толщиной в 1 пиксель.

Если заданы два значения, то первое относится к верхнему и нижнему сегментам границы, второе – к правому и левому, при этом толщина сегмента в каждой из пар будет одинаковой. Например,

DIV {Border-width: thin medium}

В этом примере для контейнеров DIV устанавливаются тонкие (thin) сегменты границы сверху и снизу контейнера и средние (medium) слева и справа.

Если задано три значения, то первое применяется к толщине верхнего сегмента границы, второе – левого и правого, а третье – нижнего сегмента. Например, **H1{ Border-width: 0 px 1 px 2 px}**

В этом примере верхний сегмент границы заголовка отсутствует, слева и справа границы будут одинаковы, по 1 пикселю, а толщина нижнего сегмента будет равна 2 пикселям.

Если заданы все четыре значения, то они будут отвечать толщине верхнего, правого, нижнего и левого сегментов границы соответственно, т.е. задание толщины сегментов границы начинается с верхнего и далее по часовой стрелке, заканчивая левым сегментом. При этом запись

P {
Border-top-width: 1 px;
Border-right-width: medium;
Border-bottom-width: 2 px;
Border-left-width: thin}

будет эквивалентно правилу:

P {Border-width: 1 px medium 2px thin}.

Свойство **border-color** это свойство графического типа и задает цвет границы блока. Если требуется задать различные цвета для четырех составляющих границ, то можно пользоваться индивидуальными свойствами:

- Border-top-color - цвет верхнего сегмента границы;
- Border-right-color - цвет правого сегмента границы;
- Border-bottom-color- цвет нижнего сегмента границы;
- Border-left-color- цвет левого сегмента границы.

Значением индивидуальных свойств может быть значение цвета, указанное любым доступным в CSS способом:

- ключевым словом (red, yellow, silver и т.д);
- шестнадцатеричным кодом цвета в полной форме (“#00FFCC”, “#EEDDAA”) или сокращенно (“#0FC”, “#EDA”);
- десятичным кодом в модели RGB(25, 205, 172).

Пример, запись

P {
Border-top-color: red;
Border-right-color: #EDA;
Border-bottom-color: #41A41C;
Border-left-color: rgb (25,205,172)}

эквивалентна записи:

P { Border-color: red #EDA #41A41C rgb (25,205,172)}.

Свойство **border-style** стенографическое свойство, задает тип линий, которыми будут отображаться границы блока. Если требуется задать различные типы линий для четырех составляющих границ, то можно пользоваться индивидуальными свойствами:

- Border-top-style – тип линии верхней границы;
- Border-right-style – тип линии правой границ;
- Border-bottom-style – тип линии нижней границы;
- Border-left-style – тип линии левой границы.

Значением, как индивидуальных свойств, так и стенографического может быть тип линии границы, определяемый одним из приведенных ключевых слов:

- None – граница отсутствует, в результате значение свойства border-width также равно нулю. Это значение используется по умолчанию.
- Hidden – подобно значению none, граница отсутствует, за исключением тех случаев, когда такое значение будет применено для таблиц.
- Dotted – граница отображается линией, составленной из точек.
- Dashed – граница отображается пунктирной линией.
- Solid – граница отображается сплошной линией.
- Double – граница отображается двойной сплошной линией. Сумма значений двух линий и расстояния между ними должна быть равна значению свойства border-width.
- Groove – граница отображается “вдавленной” линией с имитацией объема.
- Ridge – противоположно значению groove. Граница отображается “выпуклой” линией с имитацией объема.
- Inset – граница отображается так, что весь блок выглядит как бы вдавленным (похоже на нажатую кнопку).
- Outset – противоположно значению inset. Весь блок выглядит выпуклым.

Стенографическое свойство border-style по аналогии с border-width позволяет сокращенно записать значения стилей для всех четырех сегментов границы. Оно допускает задание одного, двух, трех или четырех значений. Применение этих значений аналогично свойству border-width.

P { Border-style: solid dotted}

Свойства **Border-top, border-right, border-bottom и border-left** – стенографические свойства, представляющие собой обобщения для задания свойств border-width, border-style и border-color соответствующих сегментов границы блока. Например,

P {Border-top: 1 px solid red;
Border-right: 2 px solid red;
Border-bottom: 1 px solid red;
Border-left: 2 px solid red;
}

Рассмотрим пример HTML-страницы, на которой использованы различные значения свойства Border-style.

```
<HTML>
<HEAD>
<TITLE>Стили границ блоков</TITLE>
</HEAD>
<BODY>
<div style="border-style:dashed"><font size="4">Стиль границы Dashed - пунктирный</font></div><br>
<div style="border-style:dotted"><font size="4">Стиль границы Dotted - точечный</font></div><br>
```

** Стиль границы Double - двойной
**
** Стиль границы Groove - с бороздкой
**
** Стиль границы Inset - внутренний
**
** Стиль границы Outset - внешний
**
** Стиль границы Ridge - гребень
**
** Стиль границы Solid - сплошной **
</BODY>
</HTML>

В броузере эта HTML-страница будет выглядеть следующим образом.

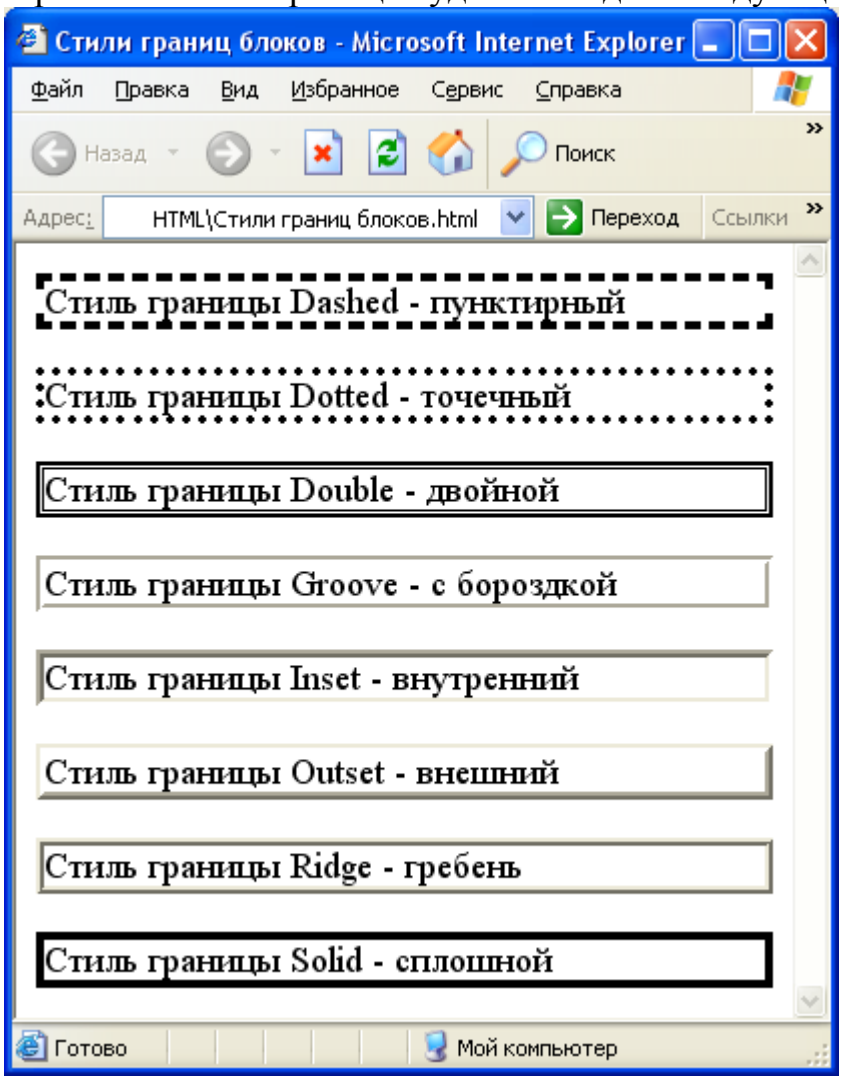

Рисунок 17 – Стили границ блоков

Свойство **border** используется для одновременного задания параметров всех четырех сегментов границы: ширины, цвета и стиля. В отличие от стенографических свойств margin и padding, свойство border не

может устанавливать различные значения для отдельных элементов границы. Чтобы это сделать, необходимо использовать другие свойства границы, а именно border-top, border-right, border-bottom и border-left.

Приведем пример форматирования блоковых элементов с использованием вышеперечисленных свойств.

```
<HTML>
<HEAD><TITLE>Пример форматирования блоков</TITLE>
<STYLE>
BODY {margin: 0;}
div {margin-left: 15px;
margin-top: 10px;
padding: 5px;
border-width: 8px;
border-style: outset;
border-color: pink;
width: 300px;
font-family: Arial, sans-serif;
font-size: 10pt;
color: red;}
p {margin-left: 40px;
margin-right: 20px;
width: 100%;
padding-left: 15px;
padding-right: 5px;
padding-top: 10px;
padding-bottom: 4px;
border-width: 18px;
border-style: groove;
border-color: darkgreen;
background-color: yellow;
color: green;
font-family: 'Times New Roman', serif;
font-size: 14pt;}
</STYLE>
</HEAD>
<BODY>
<div>Это - внутреннее содержимое блокового элемента.
Это - внутреннее содержимое блокового элемента.
Это - внутреннее содержимое блокового элемента.
Это - внутреннее содержимое блокового элемента.</div>
<p>Это - внутреннее содержимое блокового элемента.
Это - внутреннее содержимое блокового элемента.
Это - внутреннее содержимое блокового элемента. </p>
</BODY>
</HTML>
```

В броузере эта Web-страница будет выглядеть следующим образом:

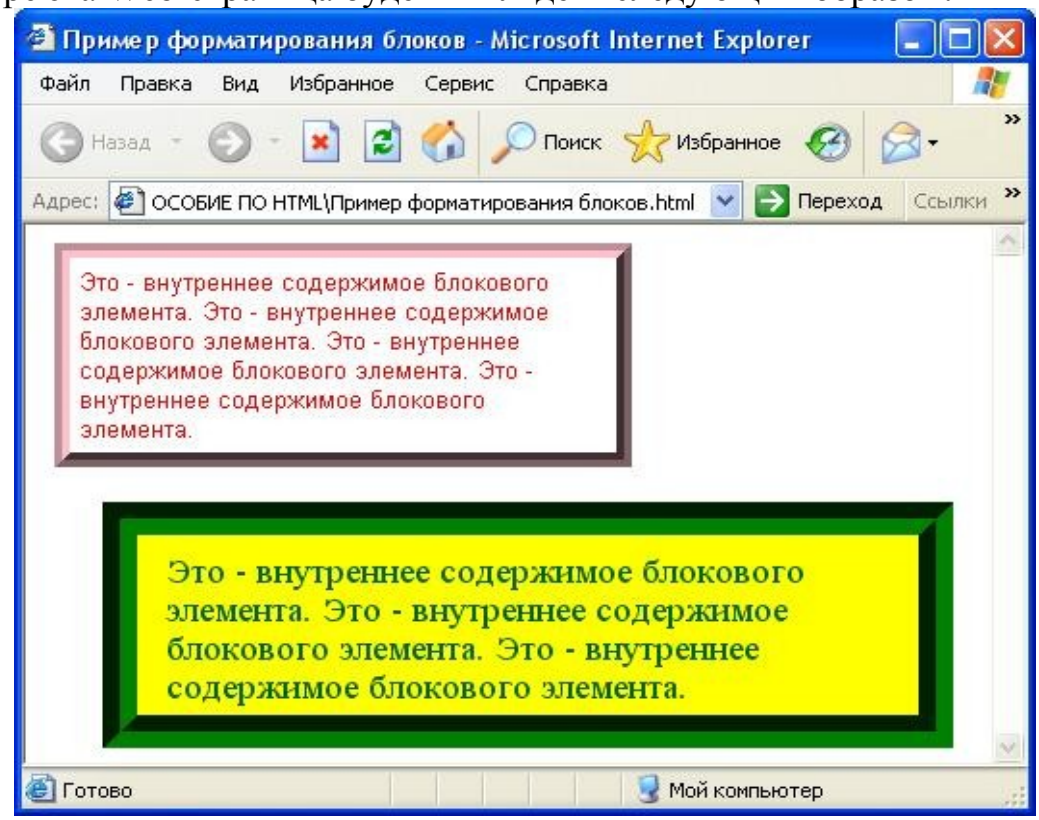

Рисунок 18 – Пример форматирования блоков

Блочные элементы можно вкладывать друг в друга, но они не должны пересекаться. Обобщением блочных элементов, с точки зрения стилей, является элемент DIV. DIV играет роль универсального блока. Блочный элемент всегда отделен от прочих элементов страницы пустой строкой. DIV не несет никакой смысловой нагрузки. Часто говорят, что DIV - это раздел страницы. Но на самом деле его применение имеет смысл только в контексте CSS. Никаких правил по умолчанию для отображения DIV не существует. Это просто новая строка текста.

DIV позволяет применить атрибуты стиля, связанные с границей блока и отступами блока от границ старшего элемента, а также “набивку”, т.е. отступ от границы блока до границы вложенного элемента

5.4 Позиционирование элементов

Суть позиционирования очень проста: любой бокс можно расположить в любом месте страницы, задав ему точные координаты. Среди параметров стиля имеются специальные свойства для позиционирования:

- left - для задания расстояния в пикселах от левого края окна (x-координата);

- top - для задания расстояния в пикселах от верхнего края окна (y-координата) (рисунок 19);

- z-index- для указания порядка, в котором элементы будут перекрывать друг друга, элементы с большим z-индексом будут появляться над элементами с меньшим z-индексом /10/.

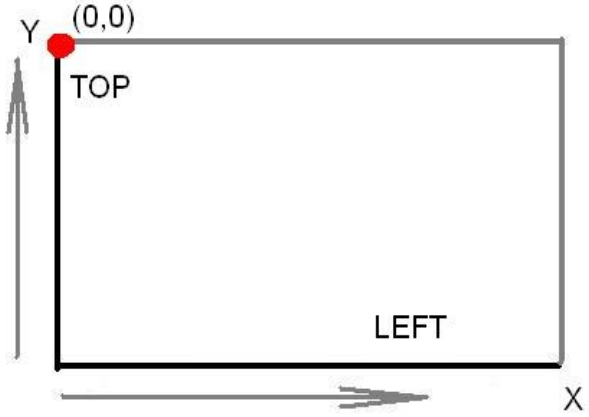

Рисунок 19 - Позиционирование элементов

Помимо свойств координат используется свойство position, которое в сочетании со свойствами left и top позволяет устанавливать элементы в определенные позиции окна. Свойство position может принимать следующие значения:

- *Static.* Элемент устанавливается в фиксированное положение относительно фона. Это способ по умолчанию, отсутствие какого бы то ни было специального позиционирования, а просто выкладывание боксов одного за другим сверху вниз. Этот порядок также называют прямой поток (рисунок 20).

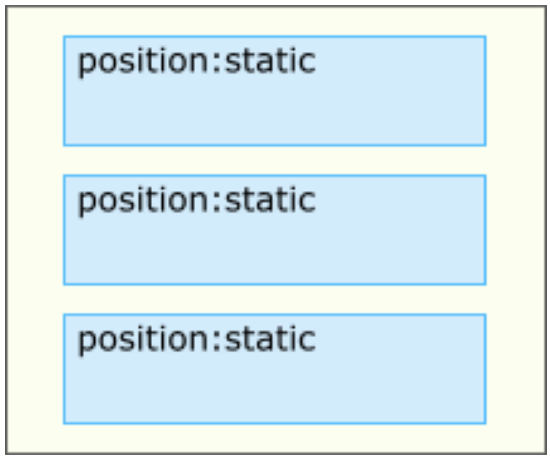

Рисунок 20 – Позиционирование элементов. Static

- *Fixed.* Ведет себя так же, как absolute, но при этом он не скролится вместе с остальной страницей.
- *Absolute.* Заданные свойства left и top устанавливают элемент в место с координатами x и y относительно верхнего левого угла контейнера (объекта, содержащего данный элемент). Если они определены для элемента вне контейнера, то началом отсчета координат будет верхний левый угол страницы. Положение элемента не зависит от положения его тега внутри HTML-страницы. Бокс с абсолютным позиционированием

располагается по заданным координатам, а из того места, где он должен был бы быть, он удаляется, и в этом месте сразу начинают раскладываться следующие боксы. Говорят, что он “исключается из потока“ (рисунок 21).

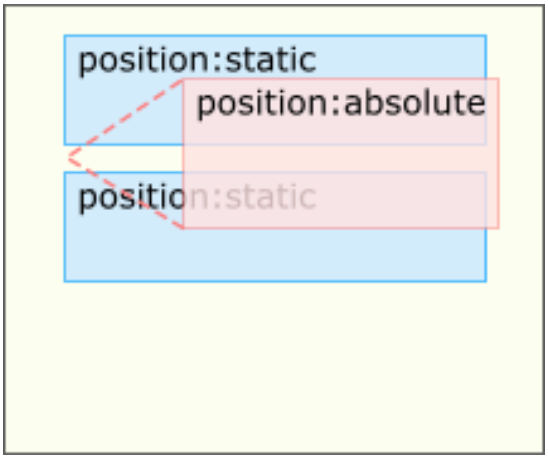

Рисунок 21 – Позиционирование элементов. Absolute

Relative. Элемент будет установлен в соответствии с тем, в каком месте исходного текста он находится, это значение установлено по умолчанию. Такой бокс можно сдвинуть относительно того места, где он был бы в потоке, но при этом из потока он не исключается, а продолжает занимать там свое место. То есть сдвигается со своего места он только визуально, а положение всех боксов вокруг него никак не меняется (рисунок 22).

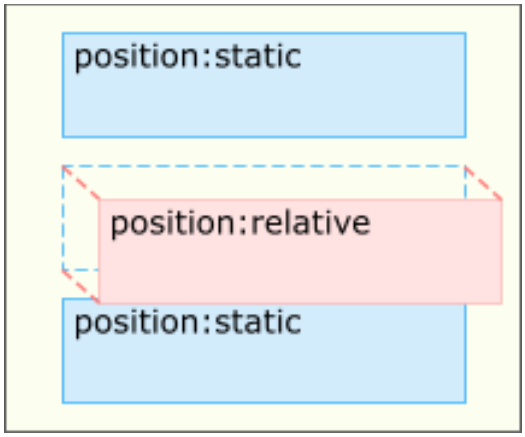

Рисунок 22 – Позиционирование элементов. Relative

5.4.1 Абсолютное позиционирование

Позиционирование блоков. Как уже было сказано, абсолютное позиционирование подразумевает назначение точных координат расположения элемента. Координаты задаются посредством свойств CSS top и left. Если блок находится непосредственно в теле страницы, точкой отсчета (началом координат) является левый верхний угол страницы. Рассмотрим пример трех блоковых элементов, позиционированных абсолютно и частично перекрывающих друг друга:

```
<HTML>
<HEAD>
<TITLE>Пример абсолютного позиционирования</TITLE>
<STYLE>
BODY { margin: 0; }
div.1 { background-color: #FF6666; width: 200px; height: 100px;
position: absolute; top: 25px; left: 30px; }
div.2 { background-color: #FF0000; width: 200px; height: 100px;
position: absolute; top: 55px; left: 60px; }
div.3 { background-color: #CC0000; width: 200px; height: 100px;
position: absolute; top: 85px; left: 90px; }
</STYLE>
</HEAD>
<BODY>
<div class=1>Элемент 1</div>
<div class=2>Элемент 2</div>
<div class=3>Элемент 3</div>
</BODY>
</HTML>
```

В броузере это будет выглядеть следующим образом:

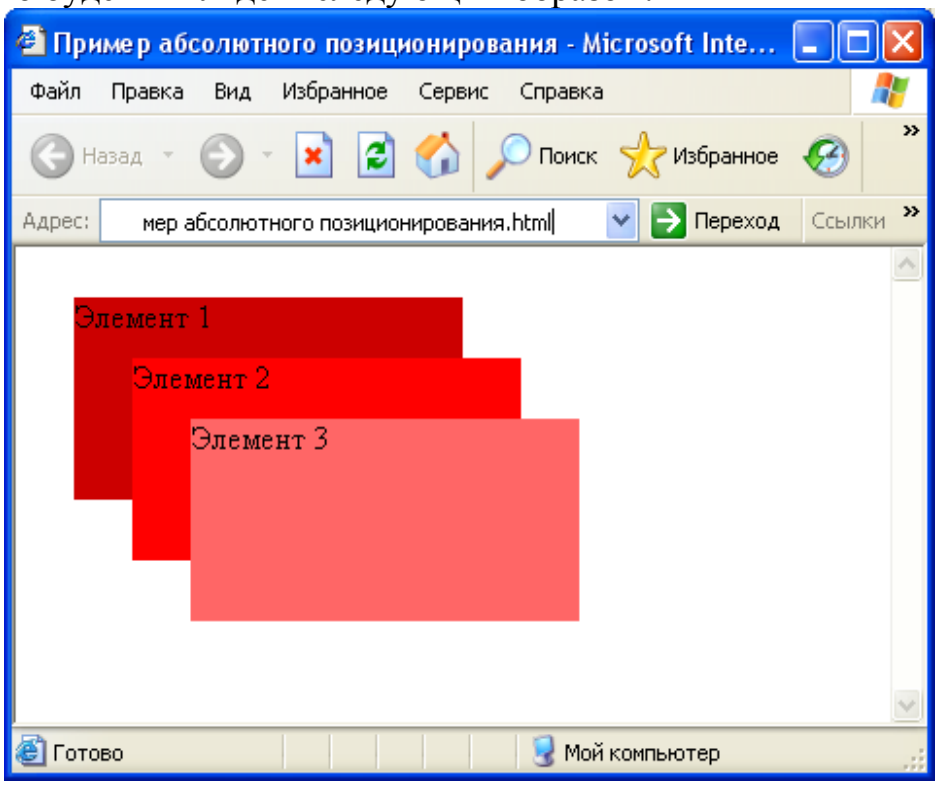

Рисунок 23 - Пример абсолютного позиционирования

Таким образом, можно создавать Web-страницы с любым количеством абсолютно позиционированных элементов, управляя положением элементов на странице с точностью до одного пиксела.

Рассмотрим еще одно интересное свойство CSS - z-index или “третье измерение”, о котором говорилось выше. Это свойство позволяет управлять

порядком наслоения абсолютно позиционированных элементов друг на друга, присваивая каждому из них координату “третьего измерения” - z-координату.

По умолчанию свойство z-index принимает значение auto, то есть “слои” перекрывают друг друга в том порядке, в каком они описаны в коде страницы. При этом тот блок, у которого значение свойства z-index больше выводится на первый план, далее следует блок, у которого значение z-index меньше, и, наконец, блок с наименьшим значением z-index располагается ниже всех блоков. В этом мы могли убедиться в предыдущем примере - в самом низу находится слой, описанный первым, а самым верхним оказался слой, описанный последним.

Используем свойство z-index для предыдущего примера. Для этого необходимо прописать свойство каждому из слоев так, чтобы они выстроились в обратном порядке:

```
<HTML>
<HEAD>
<TITLE>Пример третьего измерения</TITLE>
<STYLE>
BODY { margin: 0; }
div.1 {background-color: #FF6666; width: 200px; height: 100px;
position: absolute; top: 15px; left: 15px; z-index: 3; }
div.2 {background-color: #FF0000; width: 200px; height: 100px;
position: absolute; top: 30px; left: 30px; z-index: 2; }
div.3 {background-color: #CC0000; width: 200px; height: 100px;
position: absolute; top: 45px; left: 45px; z-index: 1; }
</STYLE>
</HEAD>
<BODY>
<div class=1>Элемент 1</div>
<div class=2>Элемент 2</div>
<div class=3>Элемент 3</div>
</BODY>
</HTML>
```

В сочетании с простейшими обработчиками событий JavaScript это позволит создавать эффектные меню для сайта. При этом, необходимо немного вытянуть каждый элемент и добавить ему рамку, например:

```
<HTML>
<HEAD>
<TITLE>Пример третьего измерения</TITLE>
<STYLE>
BODY {margin: 0;
}
div.1 { background-color: #FF6666; width: 200px; height: 60px;
position: absolute; top: 15px; left: 15px; z-index: 3; padding: 5px;
border-width:6px; border-color: silver; border-style: outset; }
```

```
div.2 { background-color: #FF0000; width: 200px; height: 60px;
position: absolute; top: 45px; left: 45px; z-index: 2; padding: 5px;
border-width:6px; border-color: silver; border-style: outset; }
div.3 { background-color: #CC0000; width: 200px; height: 60px;
position: absolute; top: 75px; left: 75px; z-index: 1; padding: 5px;
border-width:6px; border-color: silver; border-style: outset; }
</STYLE>
</HEAD>
<BODY>
<div class=1
onmousedown=“this.style.borderStyle='inset';”
onmouseup= “this.style.borderStyle='outset';”>
Элемент 1
</div>
<div class=2
onmouseover=“this.style.zIndex='4';”
onmouseout=“this.style.zIndex='2';”
onmousedown=“this.style.borderStyle='inset';”
onmouseup=“this.style.borderStyle='outset';”>
Элемент 2</div>
<div class=3
onmouseover=“this.style.zIndex='4';”
onmouseout= “this.style.zIndex='1';”
onmousedown=“this.style.borderStyle='inset';”
onmouseup= “this.style.borderStyle='outset';”>
Элемент 3</div>
</BODY>
</HTML>
```

В результате получилось динамическое меню, при наведении мышки на элементы находящиеся на нижних слоях они активизируются (рисунок 24, рисунок 25, рисунок 26). Таким образом, средствами CSS можно создавать динамические меню, не используя громоздкие графические кнопки.

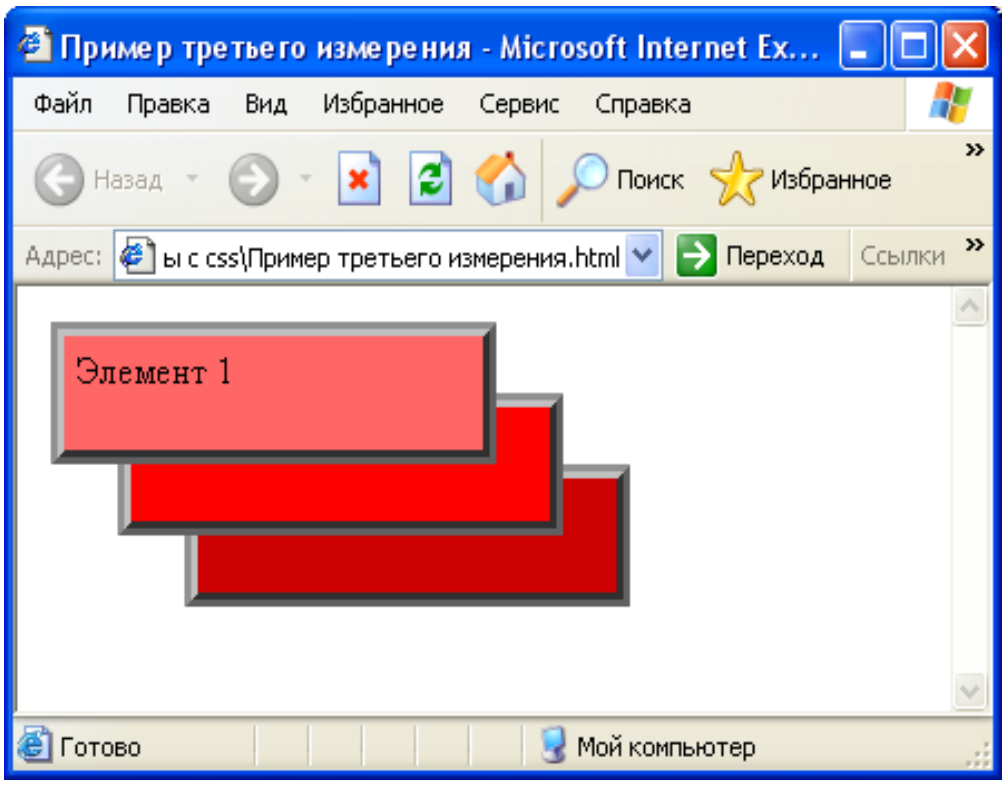

Рисунок 24 – Пример третьего измерения. Элемент 1

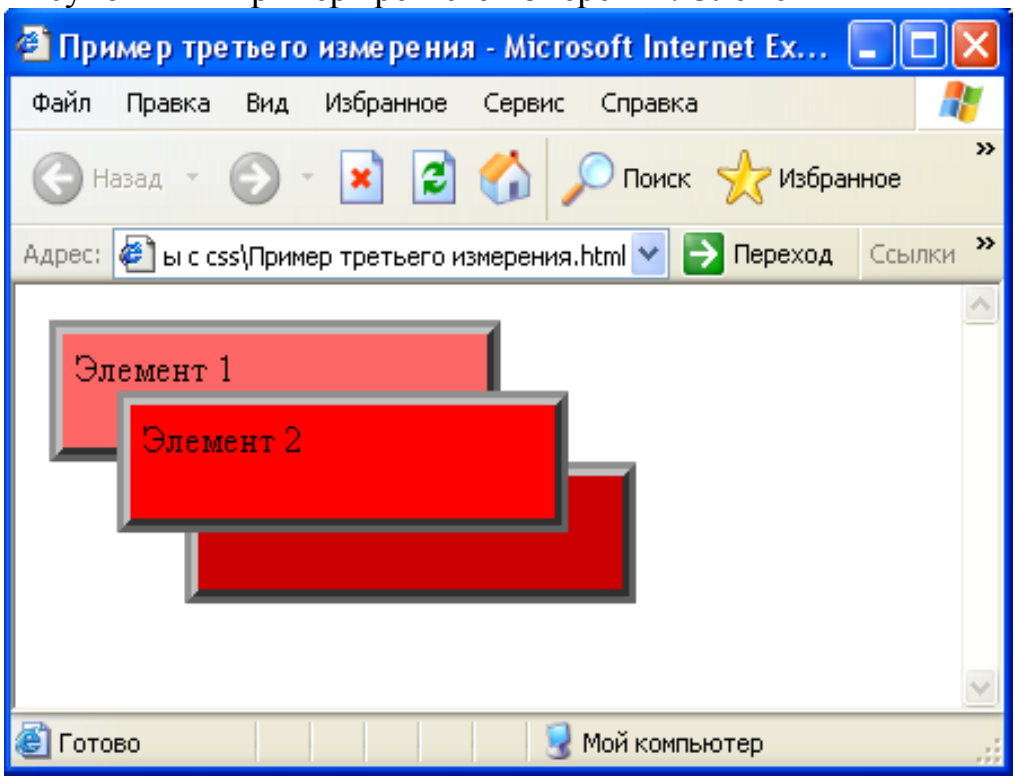

Рисунок 25 – Пример третьего измерения. Элемент 2

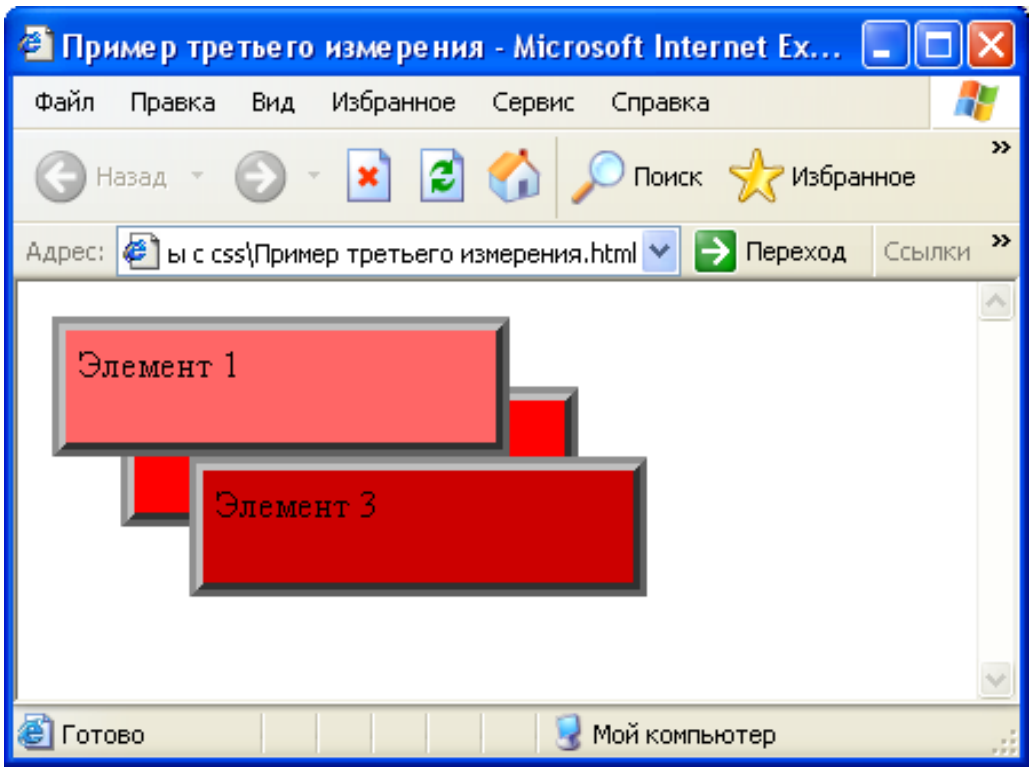

Рисунок 26 – Пример третьего измерения. Элемент 3

Рассмотрим еще один пример абсолютного позиционирования блоков. В данном случае для создания объемных заголовков. Создается HTML-документ, в котором объемный заголовок создается с помощью трех наложенных друг на друга надписей. Одна надпись предназначена для эффекта тени (задний план), вторая располагается над первой (передний план), еще одна используется для создания эффекта подсветки. Позиционирование надписей производится параметрами атрибута STYLE контейнерных тегов <DIV>, в которые заключены теги абзаца.

```
<HTML>
<HEAD>
<TITLE> 3D Эффект
</TITLE>
</HEAD>
<STYLE>
P {font-family:sans-serif; font-size:72; font-weight:800; color: 00aaaa}
P.highlight {color:silver}
P.shadow {color:darkred}
</STYLE>
<BODY BGCOLOR = aeb98c>
<! Тень >
<DIV STYLE = “position: absolute; top:5; left:5”>
<P CLASS = shadow> Объемный заголовок </P>
</DIV>
<! Подсветка>
<DIV STYLE = “position: absolute; top: 0; left: 0”>
```

<P CLASS = highlight> Объемный заголовок </P>
<! Передний план>
<DIV STYLE = “position: absolute; top: 2; left: 2”>
<P> Объемный заголовок </P>
</DIV>
</BODY>
</HTML>

В броузере эта Web-страница будет выглядеть следующим образом:

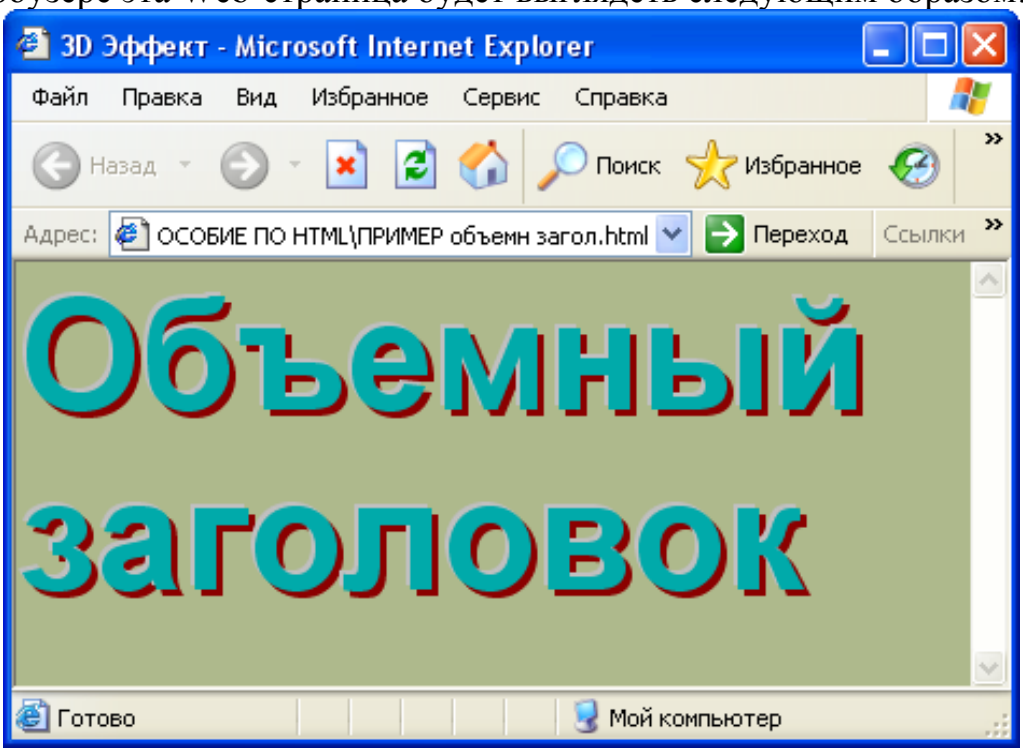

Рисунок 27 – Пример объемного заголовка

Таким образом, CSS позволяют создавать различные объемные объекты без использования графики.

Позиционирование текста. В CSS помимо позиционирования блоков имеется возможность позиционирования текста. Отличие позиционирования текста заключается в том, что используются не привычные теги DIV, P, H1-H6, а теги работы с текстом, такие как B, U, S и т.д.

Для задания абсолютного позиционирования текста используется также свойство position: absolute. При этом текст будет отображаться, начиная от левого верхнего края окна броузера, независимо оттого, что расположено до и после него. При этом может быть наложение текста. Рассмотрим пример абсолютного позиционирования текста. Имеются две одинаковые надписи, однако одна накладывается на другую.

<HTML>
<HEAD>
<TITLE> Абсолютное позиционирование текста </TITLE>
</HEAD>
<BODY STYLE=“Background-color: #FFC;”>
<B STYLE= “position: absolute; top:10; left: 20; font-size:30;”>Абсолютное позиционирование текста</B>

<B STYLE= “position: absolute; top:20; left: 30; font-size: 30;”>Абсолютное позиционирование текста</B>
</BODY>
<HTML>
В броузере эта Web-страница будет выглядеть следующим образом.

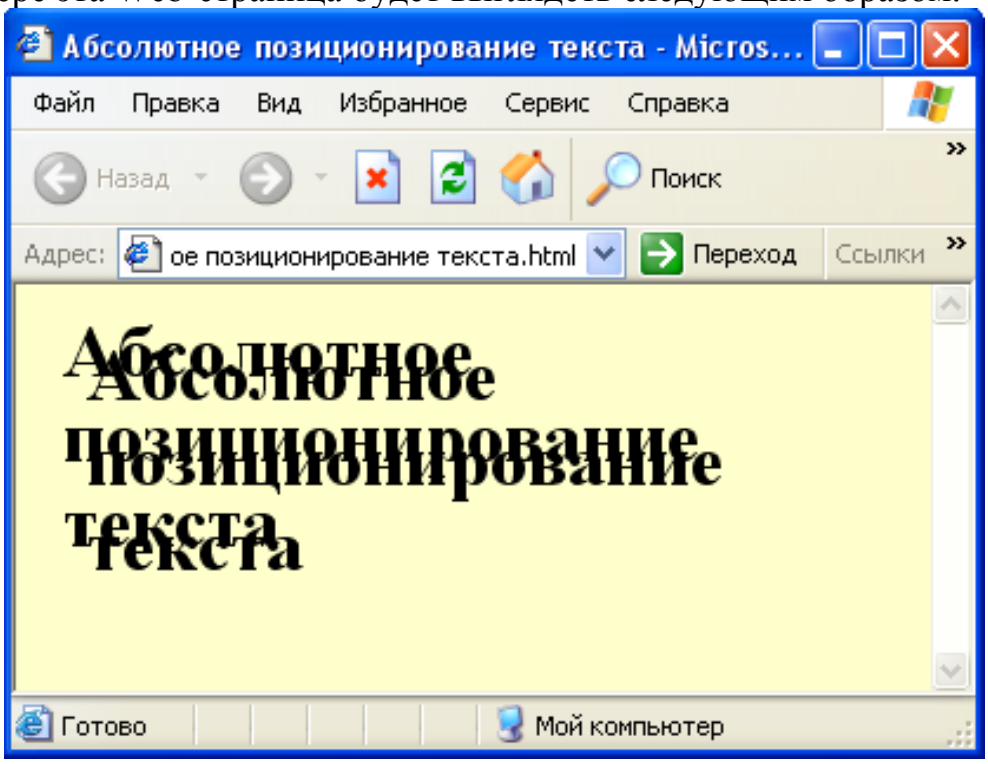

Рисунок 28 – Пример абсолютного позиционирование текста

На рисунке видно, что обе надписи начинают свой отчет с левого верхнего угла окна броузера, в результате чего получилось наложение текста.

При этом, позицию блокового элемента можно задать в процентном соотношении от величины окна броузера. Рассмотрим пример использования трех элементов, каждый из которых позиционирован абсолютно, позиционирование задано в процентах.

```
<HTML>
<TITLE>Задание позиционирования блоков в процентах</TITLE>
<HEAD>
<STYLE type=text/css>
BODY {FONT-WEIGHT: bold; FONT-SIZE: 100%; FONT-FAMILY: Verdana}
.d1 {FONT-SIZE: 900%; LEFT: 10%; WIDTH: 70%; COLOR: #ffccff;
POSITION: absolute; TOP: 10%}
.d2 {FONT-SIZE: 350%; LEFT: 20%; WIDTH: 60%; COLOR: #aaccff;
POSITION: absolute; TOP: 150px}
.d3 {FONT-SIZE: 80%; LEFT: 35%; WIDTH: 35%; COLOR: navy; POSITION:
absolute; TOP: 10%}
B {FONT-SIZE: 100%; COLOR: #0088bb}
I {COLOR: #0088bb}
</STYLE>
</HEAD>
<BODY>
<DIV class=d1>HTML</DIV>
<DIV class=d2>Учебник</DIV>
<DIV class=d3>
<P>Данный <B>учебник HTML</B> и <B>CSS</B> поможет начинающему
пользователю, шаг за шагом, от начала и до конца создать свою собственную
страницу в интернете,
<B>WEB-узел</B> или их локальную версию на <B>CD</B>,
а также может послужить <B>on-line</B> <B>справочником</B> для более
продвинутых <I>юзеров</I>.
</P>
</DIV>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 29. Как видно из рисунка 29, все три надписи наложились друг на друга, однако, в отличии от предыдущего примера, это дало красивый эффект, который можно использовать при создании различных гипертекстовых учебников или на демонстрационных сайтах.

Помимо этого, с помощью позиционирования можно создать имитацию таблицы. С помощью элемента DIV создаются блоки, причем каждый позиционирован абсолютно, позиционирование задано в процентах. Преимущества задания позиции блока в процентах проявляется при изменении размеров окна броузера. На рисунке 30 приведен пример такого позиционирования.

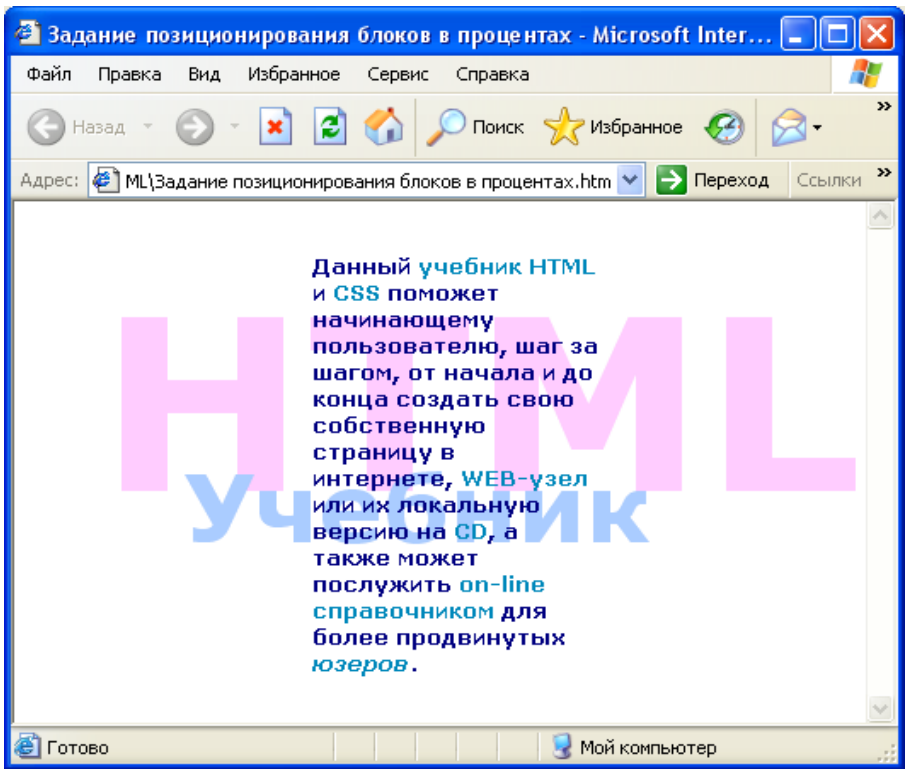

Рисунок 29 - Задание позиционирования текста в процентах

```
<HTML>
<HEAD>
<TITLE>Имитация создания таблиц средствами
позиционирования</TITLE>
<STYLE type=text/css>
BODY {FONT-SIZE: 100%; COLOR: navy; FONT-FAMILY: Verdana}
.d1 {LEFT: 5%; POSITION: absolute; TOP: 5%; font-weight: bold;}
.d2 {LEFT: 45%; POSITION: absolute; TOP: 5%;  font-weight: bold;}
.d3 {LEFT: 85%; POSITION: absolute; TOP: 5%;  font-weight: bold;}
.d4 {LEFT: 5%; POSITION: absolute; TOP: 45%;  font-weight: bold;}
.d5 {LEFT: 45%; POSITION: absolute; TOP: 45%; font-weight: bold;}
.d6 {LEFT: 85%; POSITION: absolute; TOP: 45%;  font-weight: bold;}
.d7 {LEFT: 5%; POSITION: absolute; TOP: 85%;  font-weight: bold;}
.d8 {LEFT: 45%; POSITION: absolute; TOP: 85%;  font-weight: bold;}
.d9 {LEFT: 85%; POSITION: absolute; TOP: 85%;  font-weight: bold;}
</STYLE>
</HEAD>
<BODY>
<DIV class=d1>Ячейка1</DIV>
<DIV class=d2>Ячейка2</DIV>
<DIV class=d3>Ячейка3</DIV>
<DIV class=d4>Ячейка4</DIV>
<DIV class=d5>Ячейка5</DIV>
<DIV class=d6>Ячейка6</DIV>
<DIV class=d7>Ячейка7</DIV>
```

<DIV class=d8>Ячейка8</DIV>
<DIV class=d9>Ячейка9</DIV>
</BODY>
</HTML>

Реализация данного HTML-кода представлена на рисунке 30.

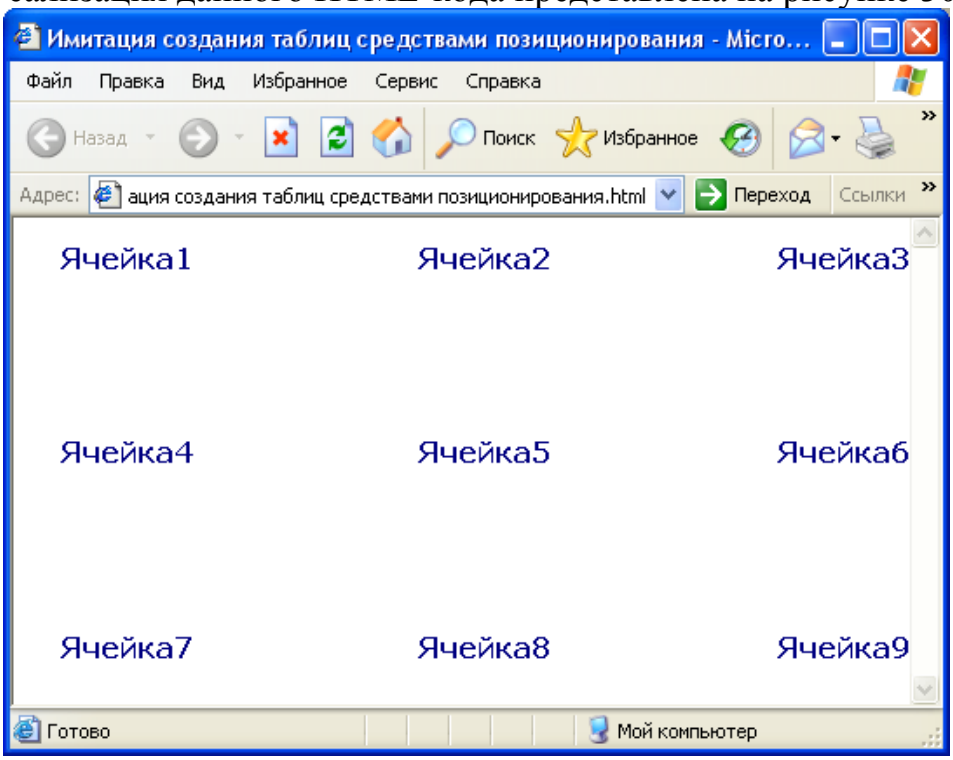

Рисунок 30 – Имитация создания таблиц средствами позиционирования

5.4.2 Относительное позиционирование

Позиционирование блоков. Относительное позиционирование позволяет разместить блоки на странице в координатах охватывающего их блока. Преимущества относительной системы координат очевидны: она позволяет сохранять взаимное расположение элементов разметки при любом размере окна броузера и его настройках по умолчанию /11/.

В качестве точки отсчета в этой системе координат выбрана точка размещения текущего родительского блока. Чтобы задать координаты блока, в этой системе применяют запись типа:

<HTML>
<HEAD>
<TITLE>Относительное позиционирование</TITLE></HEAD>
<BODY>
<DIV STYLE="border-width:2px; border-style:solid;
width: 100%; height:100px;">
<DIV STYLE= "position: relative; top:0px; left:0px; width:300 px;
border-width: 1px; border-style:solid;">
Этот блок находится в точке отсчета относительных координат
</DIV>
<DIV STYLE= "position: relative; top:0px; left:50px; width:300 px;

border-width: 1px; border-style:solid;”>
А этот блок смещен вправо на 50 px
</DIV>
</DIV>
</BODY>
</HTML>

В броузере это будет выглядеть следующим образом:

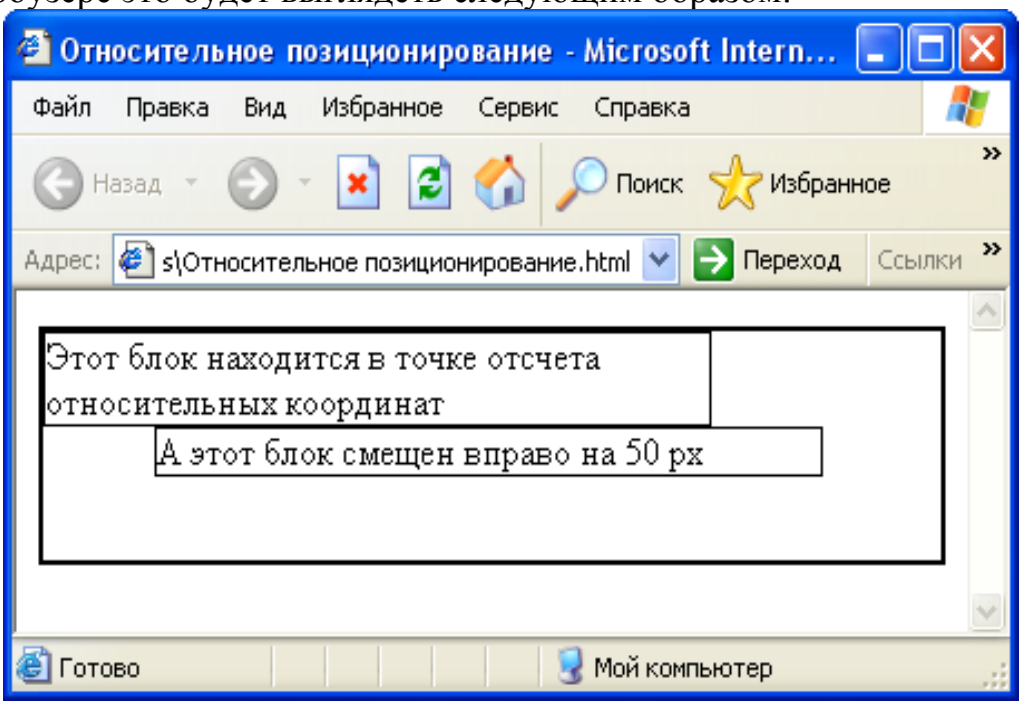

Рисунок 31 – Относительное позиционирование

При этом, для работы с относительной системой координат лучше пользоваться универсальными блоками DIV.

Позиционирование текста. Для задания относительного позиционирования текста используется также свойство position: relative. Позиционирование аналогично позиционированию блоков, однако используются другие теги: B, I, U, S и др.

Рассмотрим пример позиционирования текста на примере тега B. В данном примере рассматривается HTML-страница, представленная на рисунке 28, однако вместо свойства position: absolute используется свойство position: relative.

<HTML>
<HEAD>
<TITLE> Относительное позиционирование текста</TITLE>
</HEAD>
<BODY STYLE= “Background-color: #FFC;”>
<B STYLE= “position: relative; top:10; left: 20; font-size: 30;”>
Относительное позиционирование текста</B>
<B STYLE=“position: relative; top:20; left: 30; font-size: 30;”>
Относительное позиционирование текста</B>
</BODY>

</HTML>

Пример реализации данной Web-страницы представлен на рисунке 32.

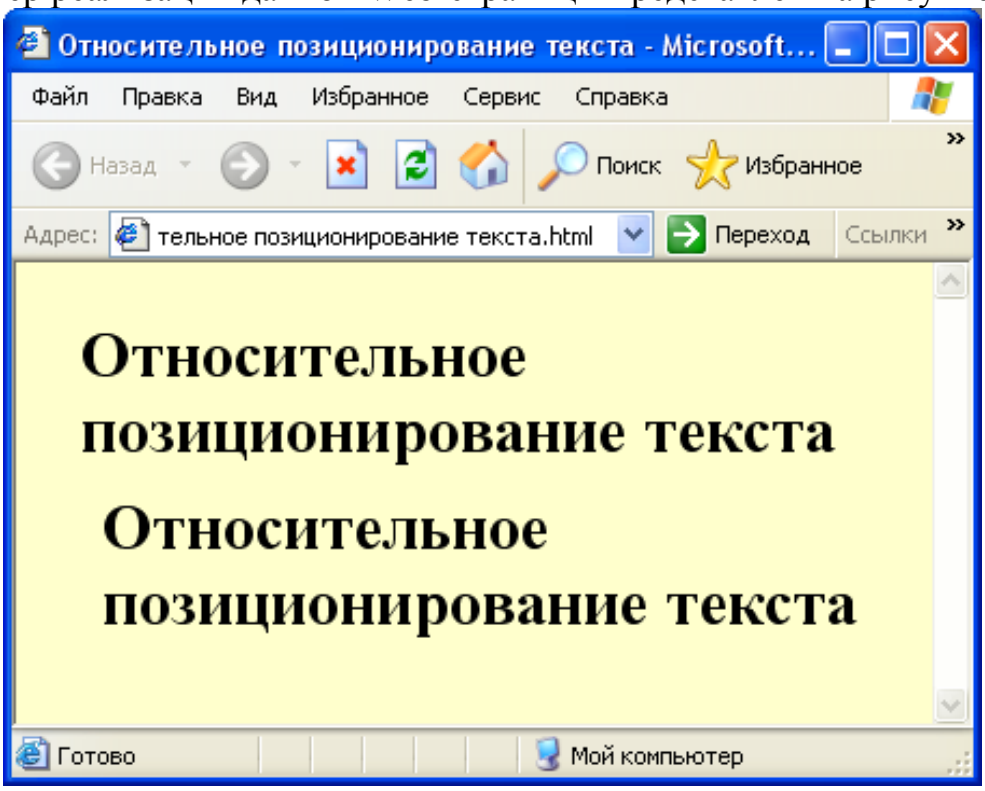

Рисунок 32 – Пример относительного позиционирования текста

На рисунке видно, что второй текст сдвинут относительно первого на 20 пикселей сверху, 30 пикселей слева, наложения не происходит, вторая надпись начинает свой отсчет от первой надписи.

5.4.3 Фиксированные блоки

Рассмотрим особенность свойства position: fixed. Согласно спецификации CSS 2, блок с описанием position: fixed зафиксирован и находится всегда в области просмотра страницы, т.е. не должен прокручиваться.

Рассмотрим конкретный пример. Используются два блока, один зафиксирован и располагается в левой верхней части окна броузера, ширина 100 пикселей. Второй прокручивается, как обычный, позиционирован абсолютно, со сдвигом вправо на 100 пикселей, чтобы он не “налезал” на фиксированный.

```
<HTML>
<HEAD> <TITLE>Фиксированные блоки</TITLE>
<STYLE>
#fix {
position: fixed;
left: 0px;
top: 0px;
width: 100 px}
#move {
```

```
position: absolute;
left: 100 px;
top: 0 px; }
</STYLE>
</HEAD>
<BODY>
<DIV id="fix">
Пункт 1<BR>
Пункт 2<BR>
Пункт 3<BR>
Пункт 4<BR>
Пункт 5<BR>
</DIV>
<DIV id="move">
Основной текст страницы прокручивается <BR>
…
Основной текст страницы прокручивается <BR>
</DIV>
</BODY>
</HTML>
```

Необходимо отметить, свойство fixed не поддерживается броузером Internet Explorer. Однако броузеры Opera, Netscape и Mozilla его понимают, тогда как Internet Explorer воспринимает подобные блоки так, будто они расположены в нормальном потоке, что, естественно, неверно.

Ниже представлен пример реализации данной страницы в окне броузера Opera.

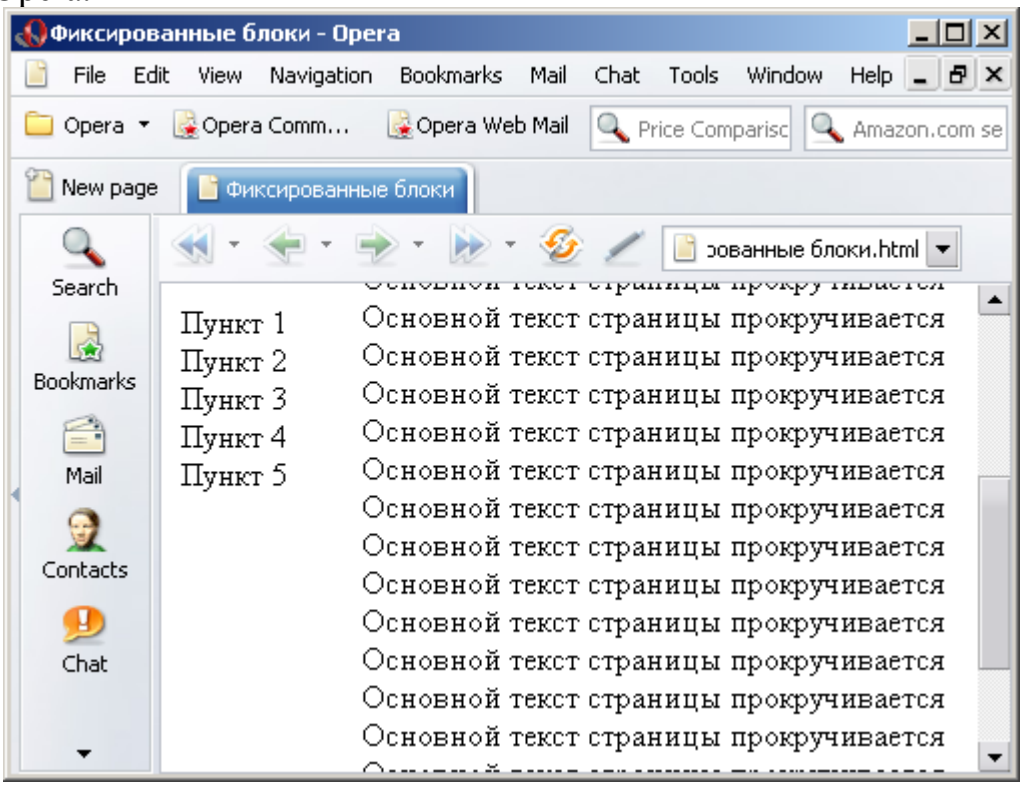

Рисунок 33 – Использование фиксированных блоков в броузере Opera

В некотором смысле фиксированные блоки похожи на фреймы. Отличие в том, что внутри самого фрейма доступна прокрутка и допустимо прокручивание содержимого отдельных фреймов независимо друг от друга.

5.4.4 Статическое позиционирование

В стандартном потоке статически позиционированные элементы ведут себя аналогично относительно позиционированным: они отображаются непосредственно сразу после предыдущего элемента в потоке. Единственное их отличие от относительно позиционируемых заключается в том, что для них нельзя установить значения свойств top и left, и тем самым сместить их, например, со строки абзаца вверх или вниз.

Для того, чтобы в CSS реализовать “резиновую” верстку, были придуманы плавающие блоки. Их нельзя позиционировать с точностью до пикселя, как, например, абсолютно позиционированные. Они могут свободно перемещаться и прижиматься к краю своего контейнера.

Свойство float. Плавающие блоки в CSS определяются свойством float. Данное свойство определяет, будет ли блок плавающим, и в какую сторону он будет перемещаться. Свойство может принимать следующие значения:

- Left – структурный блок перемещается влево. Остальное содержимое документа будет выводиться вдоль правой стороны блока, начиная с самого верха.
- Right – структурный блок перемещается вправо. Остальное содержимое документа выводится вдоль левой стороны блока, начиная самого верха.
- None – блок не перемещается (значение по умолчанию), т.е. позиционируется согласно алгоритму, заданному свойству position. Если свойство position не задано, то предполагается нормальный поток.

Рассмотрим пример простого плавающего блока. Создадим блок и зададим ему свойство float со значением right.

```
<HTML>
<HEAD>
<TITLE>Пример использования плавающих блоков </TITLE>
<STYLE>
.comment {
Background: #FFC;
Border: 1 px solid;
Padding: 5 px;
Width: 150 px;
Float: right;}
</STYLE>
</HEAD>
<BODY>
<DIV class= “comment”>
```

Плавающий блок может прилипать к левой или правой стороне контейнера, сторона задается свойством <EM>float</EM>.
</DIV>
<P>… некий фрагмент текста…
</P>
</BODY>
</HTML>

Реализация данной HTML-страницы в броузере представлена на рисунке 34.

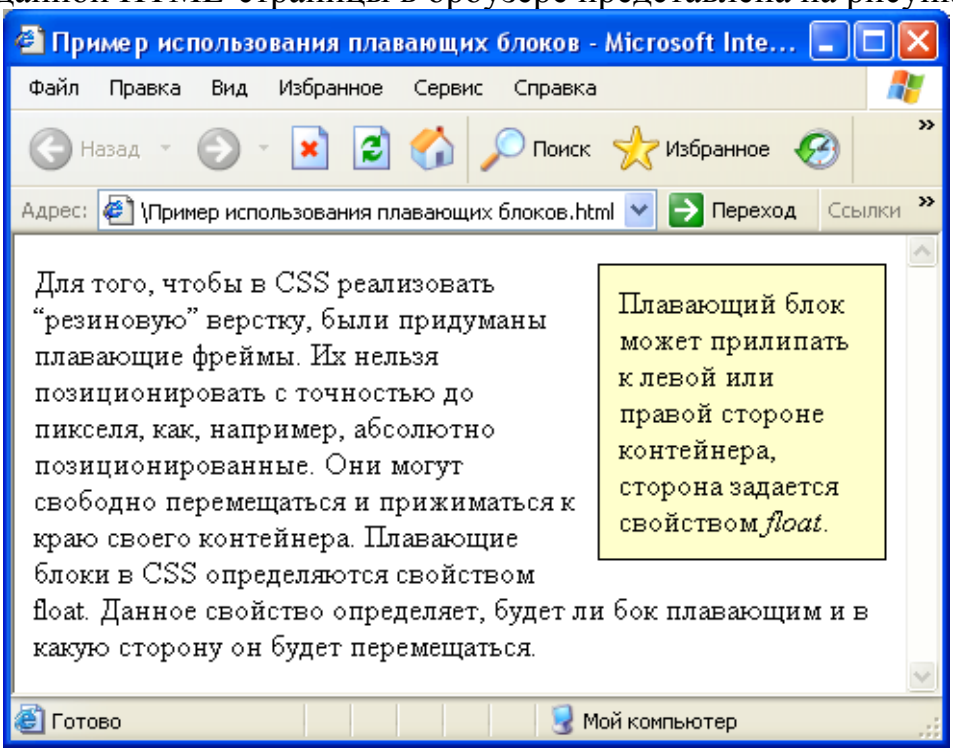

Рисунок 34 – Использование плавающих блоков, float: right

Как видно из этого рисунка, текст может обтекать не только рисунки, но и другие текстовые блоки.

Если изменить значение свойства float с right на left, то блок переместится к левому краю контейнера, которым является окно броузера (рисунок 35).

Если необходимо вставить несколько одинаковых плавающих блоков, то необходимо прописать в существующий следующий HTML-код:

<DIV class= "comment">
Первый плавающий блок должен прилипать к левому краю контейнера.
</DIV>
<DIV class= "comment">
Второй плавающий блок должен прилипать к левому краю контейнера.
</DIV>
<P>
… некий фрагмент текста….
</P>

Результат представлен на рисунке 36. Как видно из рисунка второй блок располагается левее первого, несмотря на то, что также является

левосторонним. Управлять же положением перемещаемых блоков, можно с помощью свойства clear.

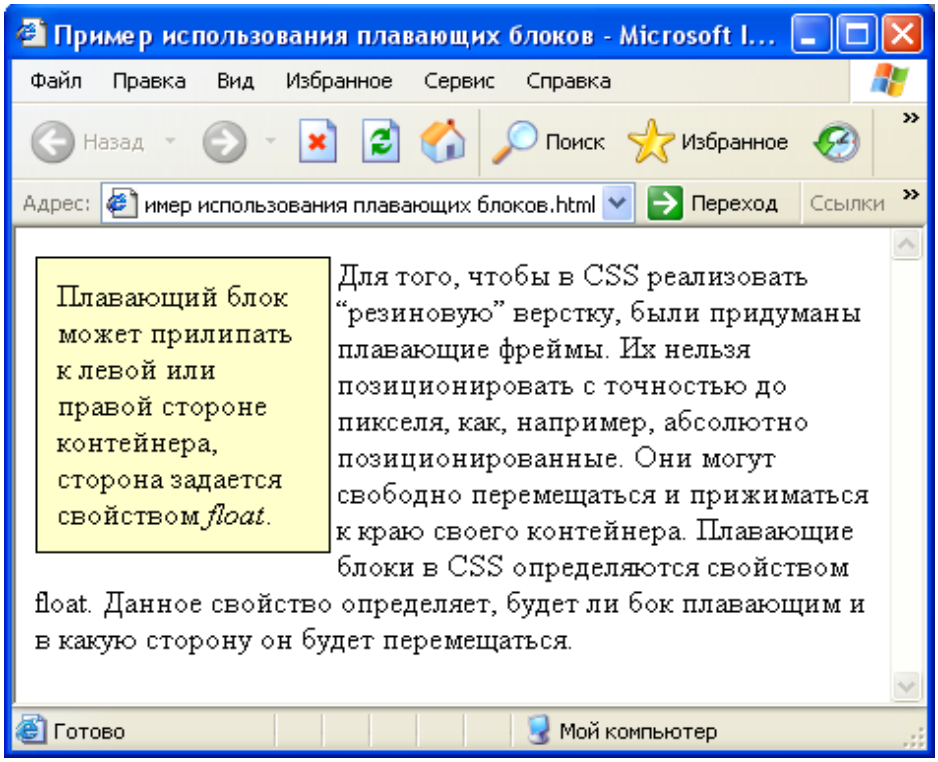

Рисунок 35 – Использование плавающих блоков, float: left

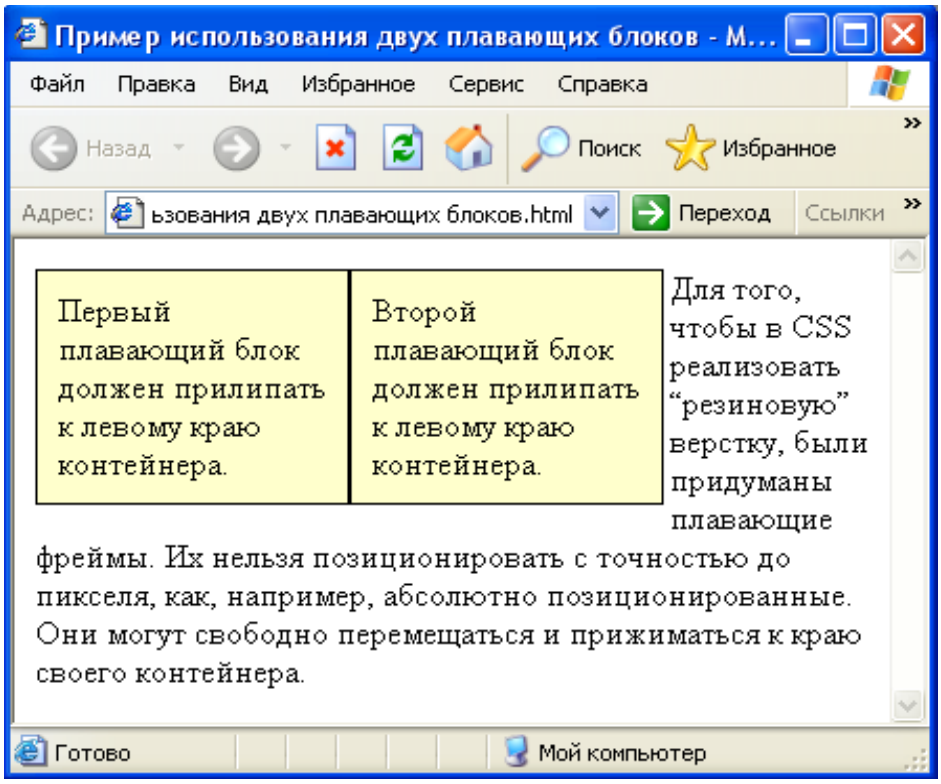

Рисунок 36 – Использование двух плавающих блоков

Свойство clear. Свойство clear определяет, какие стороны перемещаемых блоков не могут соседствовать с другими перемещаемыми блоками. Свойство применяется только для элементов блокового уровня и может принимать следующие значения:

- Left – блок должен размещаться ниже всех левосторонних плавающих блоков.

- Right – блок должен размещаться ниже всех правосторонних плавающих блоков.

- Both – блок должен размещаться ниже всех плавающих блоков.

- None – никаких ограничений на положение блока относительно перемещаемых объектов не накладывается.

Если у некоторого элемента существуют перемещаемые потомки, свойство clear на них не распространяется.

Таким образом, свойство clear можно применить и к плавающему блоку, и к любому структурному блоку. Приведем пример HTML-кода с двумя блоками, зададим для второго блока свойство clear: left.

```
<DIV class= “comment”>
Первый плавающий блок должен прилипать к левому краю контейнера.
</DIV>
<DIV class= “comment” style= “clear:left”>
Второй плавающий блок должен прилипать к левому краю контейнера.
</DIV>
<P>
…
некий фрагмент текста
…
</P>
```

В результате вид страницы изменится так, как показано на рисунке 37. Как видно из рисунка 37 второй блок прикрепился к левому плавающему блоку снизу. Получилось два блока: один сверху, другой снизу.

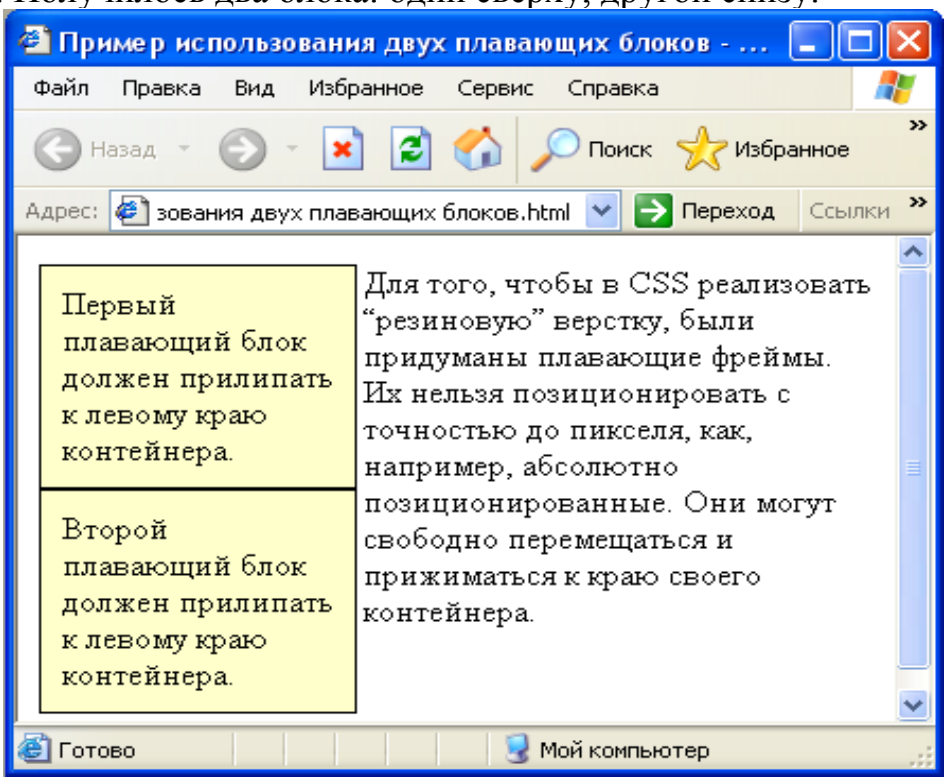

Рисунок 37 – Действие свойства clear: left на плавающий блок

Из всего сказанного выше можно сделать вывод, что несколько плавающих блоков будут располагаться на одной линии, если их суммарная ширина будет меньше ширины контейнера или будет равна ей. Если в качестве контейнера рассмотреть окно броузера, то таким образом можно легко реализовать верстку в несколько колонок. Ширину блоков при этом можно указать в процентах, так как при указании ширины в фиксированных единицах не удается заранее предсказать ширину окна броузера у конкретного пользователя.

Создадим три плавающих левосторонних блока так, чтобы их суммарная ширина не превышала 100% ширины окна.

```
<HTML>
<HEAD>
<TITLE>Три плавающих блока </TITLE>
<STYLE>
#block1 {background: #FFC;
border: 1px solid;
padding: 5 px;
width: 30%;
float: left;
}
#block2 {background: #FFC;
border: 1px solid;
padding: 5 px;
width: 30%;
float: left;
}
#block3 {background: #FFC;
border: 1px solid;
padding: 5 px;
width: 30%;
float: left;
}
</STYLE>
</HEAD>
<BODY>
<DIV id="block1">Первый плавающий блок должен прилипать к левому краю контейнера.</DIV>
<DIV id="block2">Второй плавающий блок должен выстроиться с левым плавающим блоком в одну линию.</DIV>
<DIV id="block3">Третий  плавающий блок должен выстроиться с двумя предыдущими в одну линию независимо от размера окна броузера.</DIV>
</BODY>
</HTML>
```

Реализация данного кода представлена на рисунке 38.

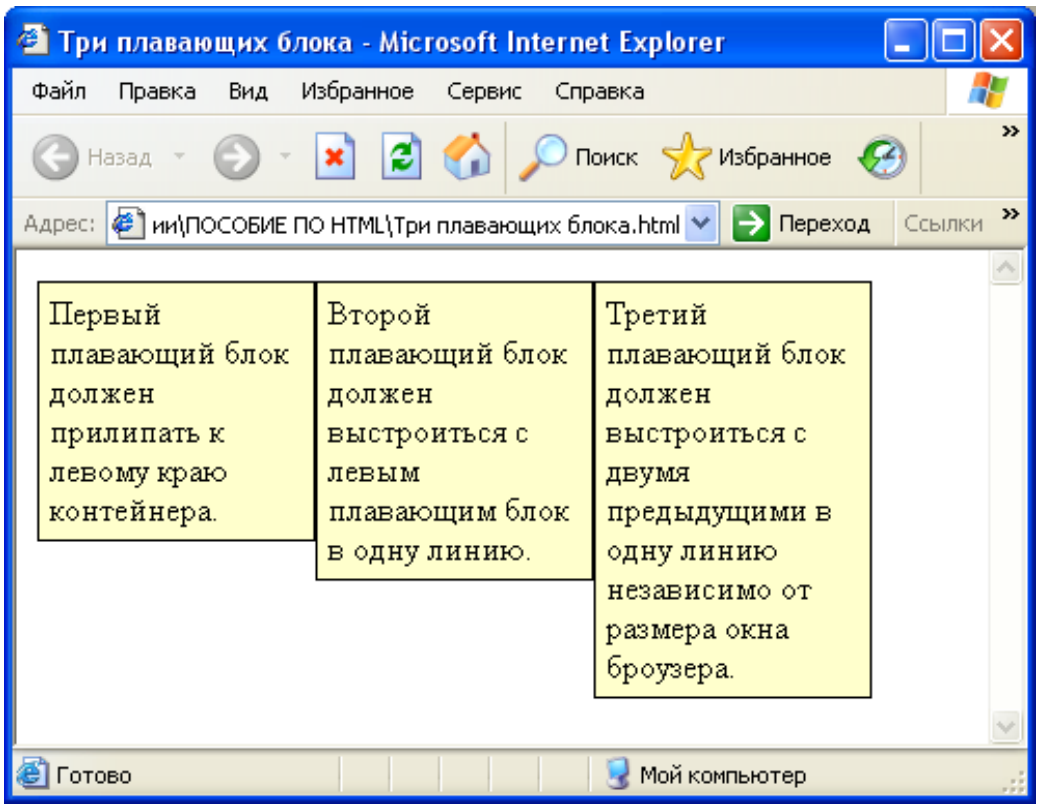

Рисунок 38 – Три блока – высота каждого зависит от содержимого

Как видно из рисунка 38, высота колонок будет зависеть от содержимого блоков. Можно легко подровнять колонки по высоте, указав в CSS-правилах для блоков свойство height:100%. Это механизм позволяет реализовать “резиновую” верстку с нужным количеством колонок (рисунок 39).

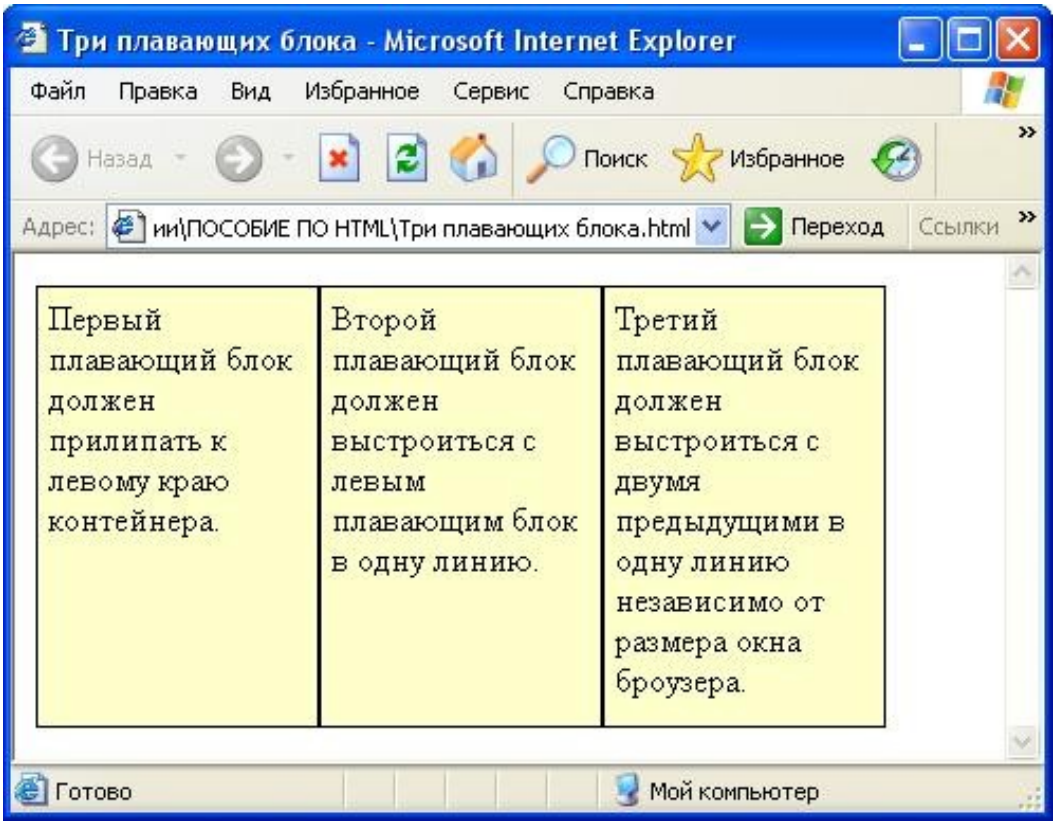

Рисунок 39 - Высота колонок выровнена

Свойство display. Свойство display задает тип конкретного блока и может принимать четыре значения: none, inline, block, list-item.

Значение display:none отключает отображение блока в окне броузера так, как будто его никогда не было в коде HTML-страницы. Блок со свойством display:none будет удален со страницы совсем и поэтому не будет занимать места.

Значение display:block определяет некий элемент как структурный блок. Это может пригодиться для восстановления видимости блока после работы свойства none. Также можно воспользоваться этим свойством для создания навигационного меню из ссылок, каждая из которых будет отдельным структурным блоком.

Значение display:inline определяет строчный блок. По умолчанию многие элементы HTML имеют такое значение свойства display – например, I, EM, B и.т.д. Это свойство полезно, в тех случаях, когда необходимо преобразовать структурный блок в строчный.

С помощью значения display:list-item элемент становится частью списка. При использовании данного значения в список можно превратить что угодно, любые структурные блоки.

Свойство visibility. Как следует из названия, этого свойство управляет видимостью блока. Оно может принимать только два значения:

- visible – обычное состояние блока (по умолчанию);
- hidden – блок становится прозрачным, т.е. невидимым.

Свойство visibility:hidden очень похоже на свойство display:none. Однако свойство visibility:hidden делает элемент прозрачным, но при этом продолжает присутствовать на странице, а свойство display:none отображается так, как будто его никогда не было. Рассмотрим конкретный пример.

```
<HTML>
<HEAD>
<TITLE>Отличия свойств visibility и display</TITLE>
<BODY>
<HR>
<P>Первый параграф. Свойство display установлено по умолчанию </P>
<P STYLE="display:none"> Второй параграф. Установлено свойство display:none </P>
<P>Третий параграф. Свойство display установлено по умолчанию </P>
<HR>
<P>Первый параграф. Свойство display установлено по умолчанию </P>
<P STYLE="visibility:hidden"> Второй параграф. Установлено свойство visibility:hidden </P>
<P>Третий параграф. Свойство display установлено по умолчанию </P>
<HR></BODY>
</HTML>
```

Реализация данной HTML-страницы в броузере представлена на рисунке 40.

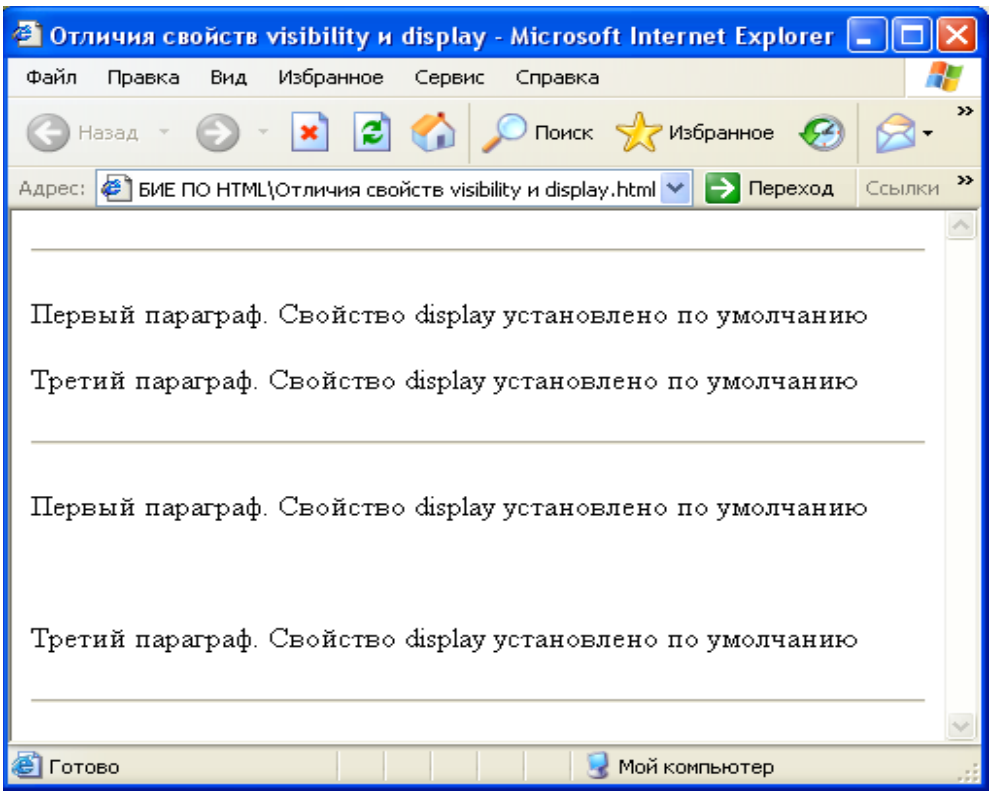

Рисунок 40 – Визуальное отличие свойств visibility:hidden и display:none

Как видно из рисунка, в первой группе второй абзац с установленным свойством display:none просто отсутствует на экране, поэтому сразу после первого абзаца идет третий. Во второй группе абзац с установленным свойством visibility:hidden также отсутствует на экране, но его место остается незанятым. Иначе говоря, он также остается на своем месте, но является прозрачным, вследствие чего невидимым.

С помощью свойства visibility можно организовывать раскрывающиеся меню. В начальном состоянии видимость всех пунктов этого меню устанавливается как visibility:hidden. Когда пользователь щелкает по команде главного меню, то включается видимость нужного пункта, с помощью объявления visibility:visible.

Свойство overflow. Свойство overflow определяет поведение элемента, когда размеры его содержимого не соответствуют размерам блока отображения, установленного свойствами top, left, width и height. Существует четыре значения этого свойства: visible, hidden, auto, scroll. Рассмотрим подробнее использование значений этого свойства.

При использовании значения overflow:visible содержимое, превышающее размеры блока, должно перекрывать блок и оставаться видимым на экране, оно заставляет элемент сжиматься или увеличиваться, чтобы полностью отобразиться в заданном блоке (для графического содержимого элемента) или увеличивает размеры блока отображения (для текстового содержимого) /11/.

Рассмотрим конкретный пример. Создадим небольшой по размерам блок и поместим внутрь него фрагмент текста, превышающий размеры созданного блока.

```
<HTML>
<HEAD>
<TITLE>Использование overflow:visible</TITLE>
<STYLE>
#small {
border: 1px dotted #660033;
padding: 0.5 em;
height: 60 px;
width: 150 px;
overflow: visible}
</STYLE>
<BODY>
<DIV id="small">
Ширина данного блока 150 пикселей, а высота 60 пикселей. Проверим, удастся ли контролировать ситуацию с переполнением.
</DIV></BODY>
</HTML>
```

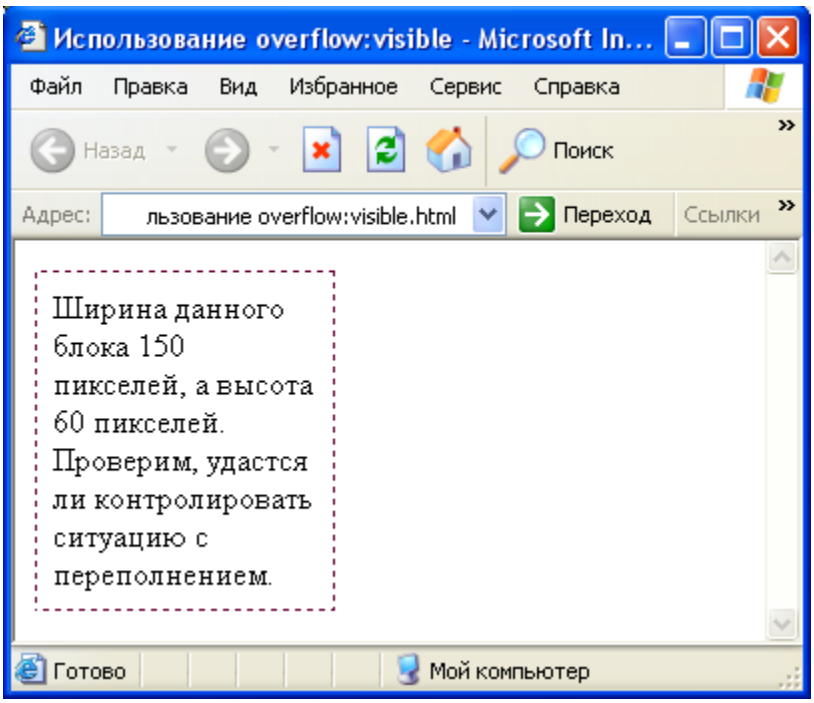

Рисунок 41 – Действие свойства overflow:visible

Как видно из рисунка 41, броузер Internet Explorer при вертикальном переполнении поступает так же, как и поступал до этого, т.е. не обращает внимания на объявление overflow:visible. Он просто растягивает сам блок, чтобы содержимое поместилось полностью. Броузеры Opera и Mozilla ведут себя в соответствии с объявлением overflow: visible, потому что это значение является значением по умолчанию.

Значение overflow:hidden предписывает броузеру обрезать содержимое, которое превышает размеры блока, при этом никакого механизма доступа к скрытой информации (например, полосы прокрутки) пользователю не предоставляет (рисунок 42).

<HTML>
<HEAD><TITLE>Использование overflow:hidden</TITLE>
<STYLE>
#small {border: 1px dotted #660033;
padding: 0.5 em;
height: 60 px;
width: 150 px;
overflow: hidden}
</STYLE>
<BODY>
<DIV id="small">
Ширина данного блока 150 пикселей, а высота 60 пикселей. Проверим, удастся ли контролировать ситуацию с переполнением.
</DIV></BODY>
</HTML>

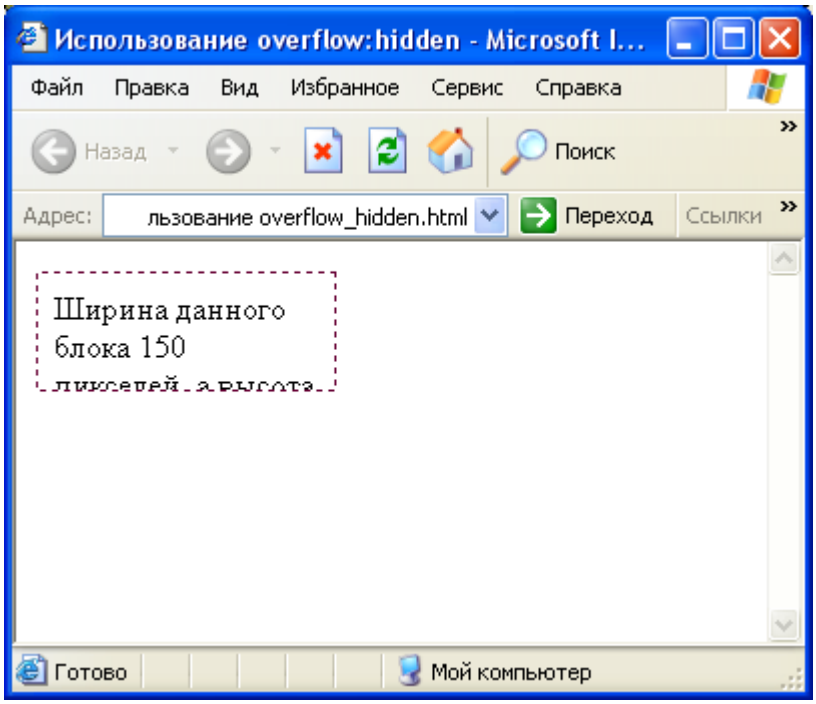

Рисунок 42 – Действие свойства overflow:hidden

Значение overflow:scroll добавляет полосы прокрутки к блоку отображения. Пользователь сможет просмотреть скрытое содержимое, прокрутив его от начала до конца (рисунок 43).

<HTML>
<HEAD><TITLE>Использование overflow:hidden</TITLE>
<STYLE>
#small {border: 1px dotted #660033;
padding: 0.5 em;
height: 60 px;
width: 150 px;
overflow: scroll}
</STYLE>

<BODY>
<DIV id="small">
Ширина данного блока 150 пикселей, а высота 60 пикселей. Проверим, удастся ли контролировать ситуацию с переполнением.
</DIV></BODY>
</HTML>

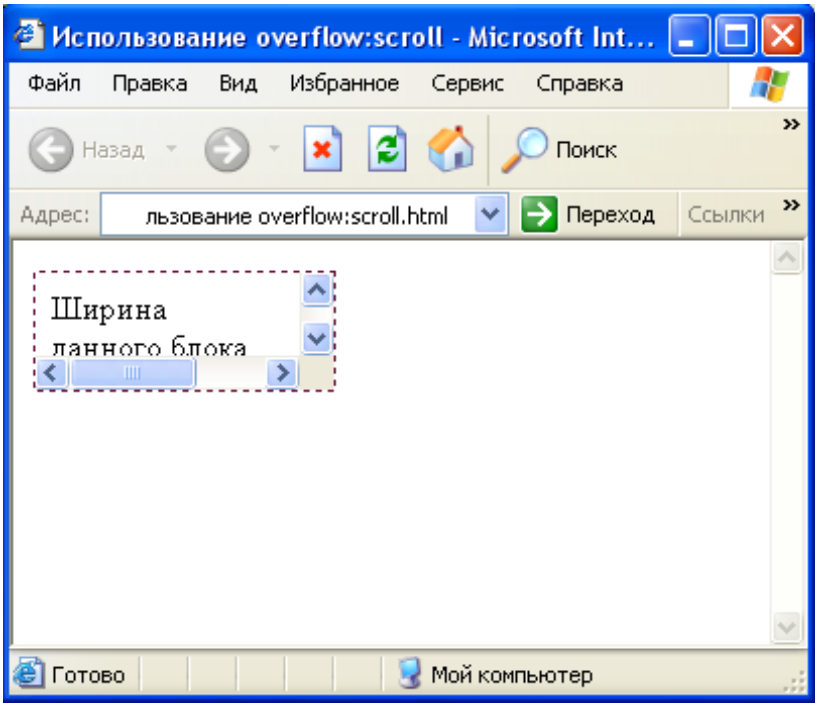

Рисунок 43 – Действие свойства overflow:scroll

Значение overflow:auto добавляет полосы прокрутки к блоку отображения в случае, если размеры содержимого элемента превосходят размеры блока отображения, т.е. добавляет к блоку только те полосы прокрутки, которые действительно необходимы (рисунок 44).

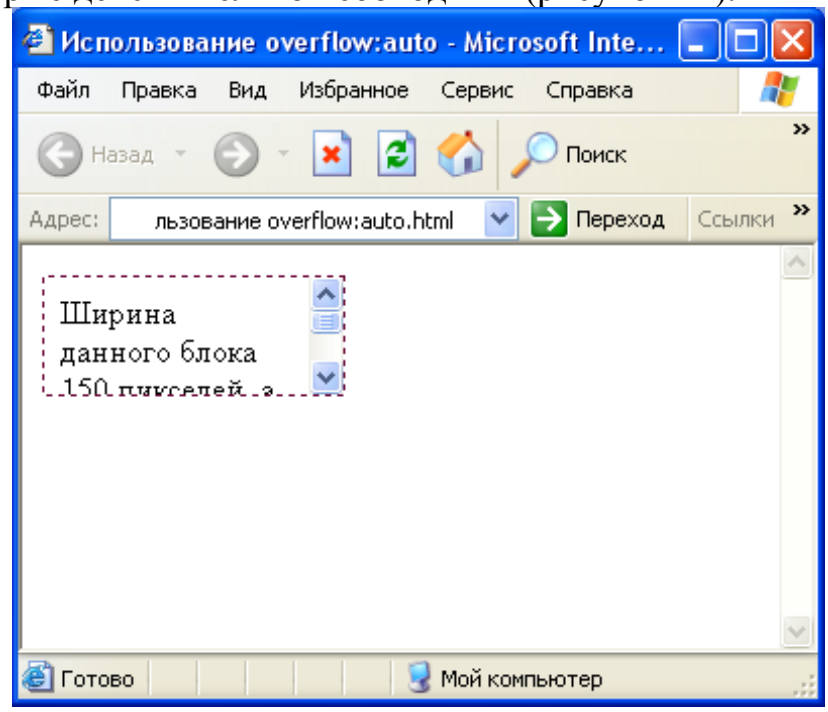

Рисунок 44 – Действие свойства overflow:auto

6 Фильтры

Фильтры – это эффекты изменения внешнего вида графики и текста на странице. С помощью фильтров можно отражать текст и графику, создавать эффекты движения, как это делается в графических редакторах.

Существуют статические и динамические фильтры. Статические фильтры изменяют внешний вид элемента, оставляя его неподвижным. Динамические фильтры позволяют трансформировать графический объект со скоростью, задаваемой пользователем /12/.

6.1 Статические фильтры

Статический фильтр можно задать как свойство в таблице стилей, используя запись вида:

Filter: название_фильтра

или при наличии параметров, такую запись:

Filter: название_фильтра (параметр1, параметр2, …, параметрN)

Существует множество статических фильтров, однако, наиболее популярные из них следующие: Alpha, Blur, Chroma, DropShadow, FlipH, FlipV, Glow, Invert, Gray, Light, Mask, Shadow, Xray, Wave, BasicImage.

Рассмотрим подробно каждый из этих фильтров и его возможные атрибуты.

6.1.1 Alpha

Задает уровень прозрачности элемента страницы.

Синтаксис:

Filter: Alpha (Opacity=начальная прозрачность, FinishOpacity=конечная прозрачность, Style=стиль, StartX=начало эффекта по горизонтали, StartY=начало эффекта по вертикали, FinishX=конец эффекта по горизонтали, FinishY=конец эффекта по вертикали)

При настройке этого фильтра можно использовать следующие параметры.

- Opacity - задает уровень прозрачности для элемента. Значение прозрачности может быть в пределах от 0 (полная прозрачность) до 100 (полностью непрозрачен).

- FinishOpacity - необязательный параметр, который определяет конечное значение прозрачности, если фильтр применяется как градиентный.

- Style - задает вид градиентной прозрачности: 0 (по умолчанию) - нет градиента, 1-линейный градиент, 2-круговой градиент, 3-прямоугольный градиент.

- StartX - горизонтальная координата, с которой начинается область градиентной прозрачности. Задается в пикселях.

- StartY - вертикальная координата, с которой начинается область градиентной прозрачности. Задается в пикселях.

- FinishX - горизонтальная координата, в которой заканчивается область градиентной прозрачности. Задается в пикселях.

- FinishY - вертикальная координата, в которой заканчивается область градиентной прозрачности. Задается в пикселях.

- Enabled - разрешение применения фильтра. Имеет два значения: true - разрешает применить фильтр, false - не разрешает.

Рассмотрим пример использования фильтра Alpha.

```
<HTML>
<HEAD><TITLE>Использование фильтра Alpha</TITLE>
</HEAD>
<BODY>
<IMG height=250 alt=original src="plast.jpg" width=250>
<IMG style= "FILTER: alpha (Opacity=0, FinishOpacity=100, Style=1, StartX=0, FinishX=100)" height=250 alt=alpha src="plast.jpg" width=250 >
<TABLE><TR align=middle><TD>
<DIV style= "FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 250px">ALPHA </DIV></TD><TD ><DIV
style="FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER: alpha(Opacity=0, FinishOpacity=100, Style=1, StartX=0, FinishX=100); WIDTH: 250px">ALPHA </DIV></TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR></TABLE>
 </BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 45.

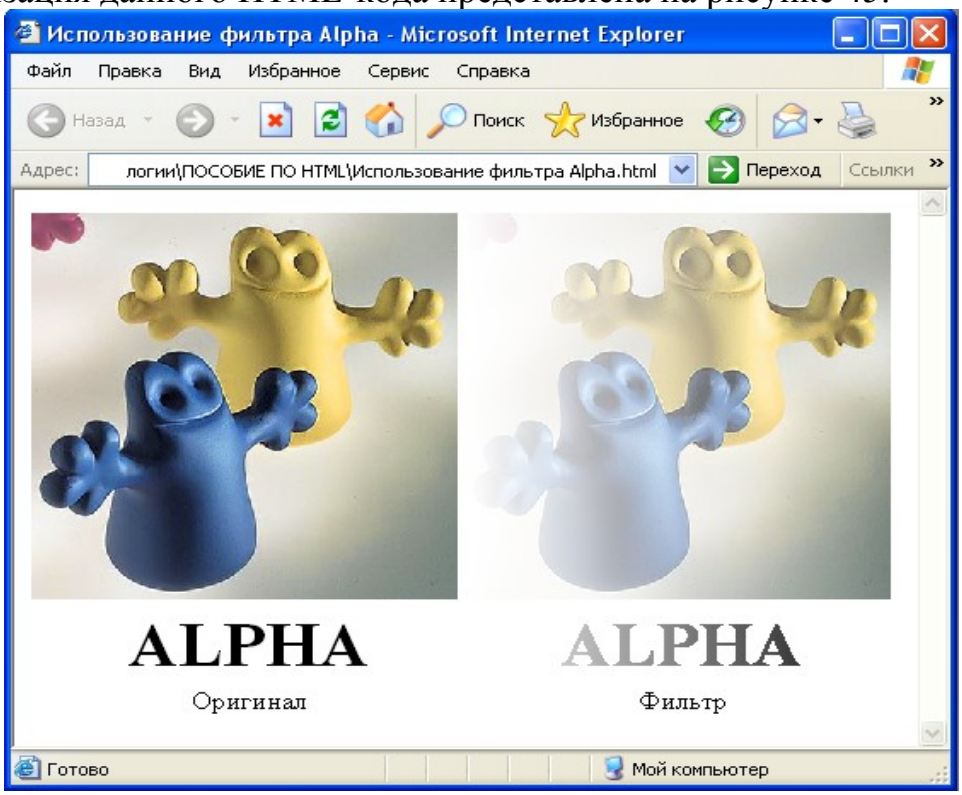

Рисунок 45 – Пример использования фильтра Alpha

6.1.2 Blur

Делает элемент страницы размытым.

Синтаксис:

Filter: Blur (Add=оригинал, Direction=направление, Strength=величина)

Изображение или текст размывается в направлении Direction, заданном в градусах (0 - вверх и далее по часовой стрелке с шагом 45 градусов). Если указано Add=1, то фильтр будет наложен на исходный объект, иначе будет отображен только результат работы фильтра.

При настройке этого фильтра можно использовать следующие параметры:

- MakeShadow=true - элемент страницы будет отображаться как тень.
- MakeShadow=false (по умолчанию) - элемент страницы будет отображаться как обычно.
- PixelRadius - задает размер области размытия. Может принимать значения от 0,0 до 100,0.
- ShadowOpacity - указывает прозрачность тени. Может принимать заначения от 0,0 (что соответствует полной прозрачности) до 1,0 (полная непрозрачность).
- Enabled - разрешение применения фильтра. Имеет два значения: true (по умолчанию) - разрешает применить фильтр false - не разрешает.

Рассмотрим пример использования фильтра Blur.

```
<HTML>
<HEAD><TITLE>Использование фильтра Blur</TITLE>
</HEAD>
<BODY>
<IMG height=250 alt=original src="plast.jpg" width=250 >
<IMG style= "FILTER: Blur(Add=1, Direction=90, Strength=25)" height=250
alt=blur src= "plast.jpg" width=250>
<TABLE> <TR align=middle>
 <TD>
<DIV style= "FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH:
250px">BLUR
</DIV></TD>
<TD>
<DIV style="FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER:
Blur(Add=1, Direction=135, Strength=5); WIDTH: 250px">BLUR
</DIV></TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR></TABLE>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 46.

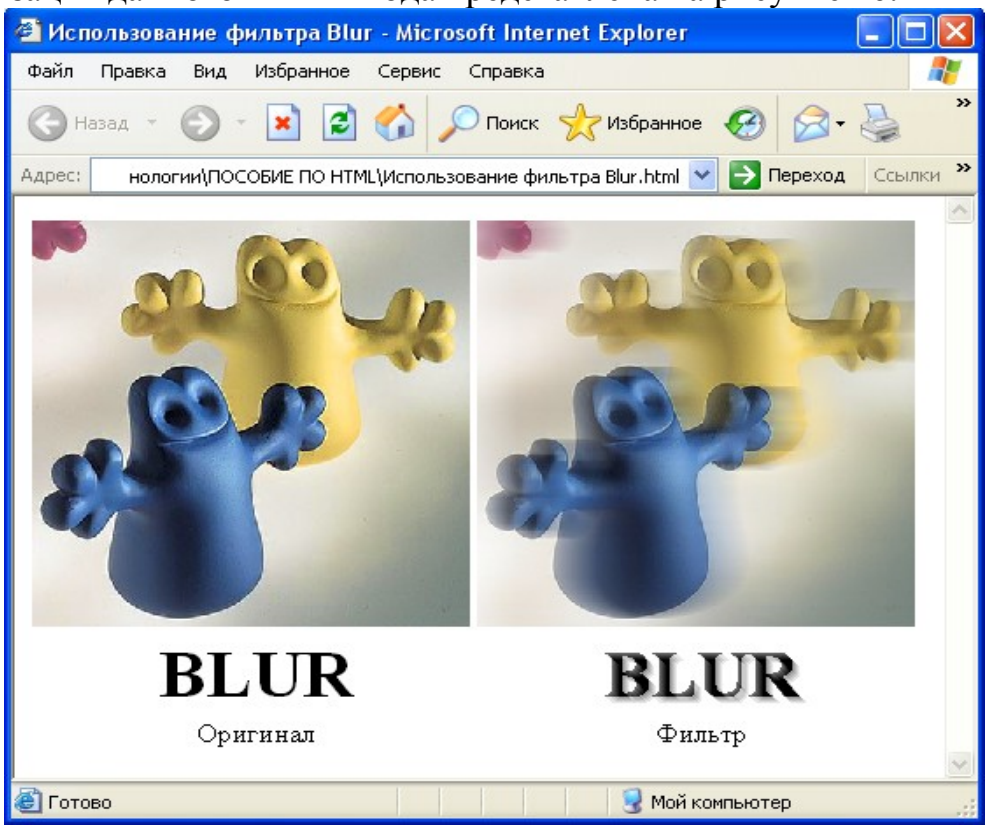

Рисунок 46 – Пример использования фильтра Blur

6.1.3 Chroma

Делает прозрачным отдельный цвет элемента Web-страницы.

Синтаксис:

Filter: Chroma (Color=цвет)

Цвет, заданный в форме #RRGGBB становится прозрачным.

При настройке этого фильтра можно использовать следующие параметры:

- Color - задает цвет, который будет прозрачным.

- Enabled - разрешение применения фильтра. Имеет два значения: true (по умолчанию) - разрешает применить фильтр false - не разрешает.

Рассмотрим пример использования фильтра Chroma.

```
<HTML>
<HEAD>
<TITLE>Использование фильтра Chroma</TITLE>
</HEAD>
<BODY>
<IMG height=250 alt=original src= “plast.jpg” width=250></TD>
<IMG style= “FILTER: Chroma(Color=blue)” height=250 alt=Chroma src=“plast.jpg” width=250>
<TABLE> <TR align=middle>
<TD width=250>Оригинал</TD>
<TD width=250>Фильтр</TD></TR>
```

</TABLE>
</BODY>
</HTML>

Реализация данного HTML-кода представлена на рисунке 47.

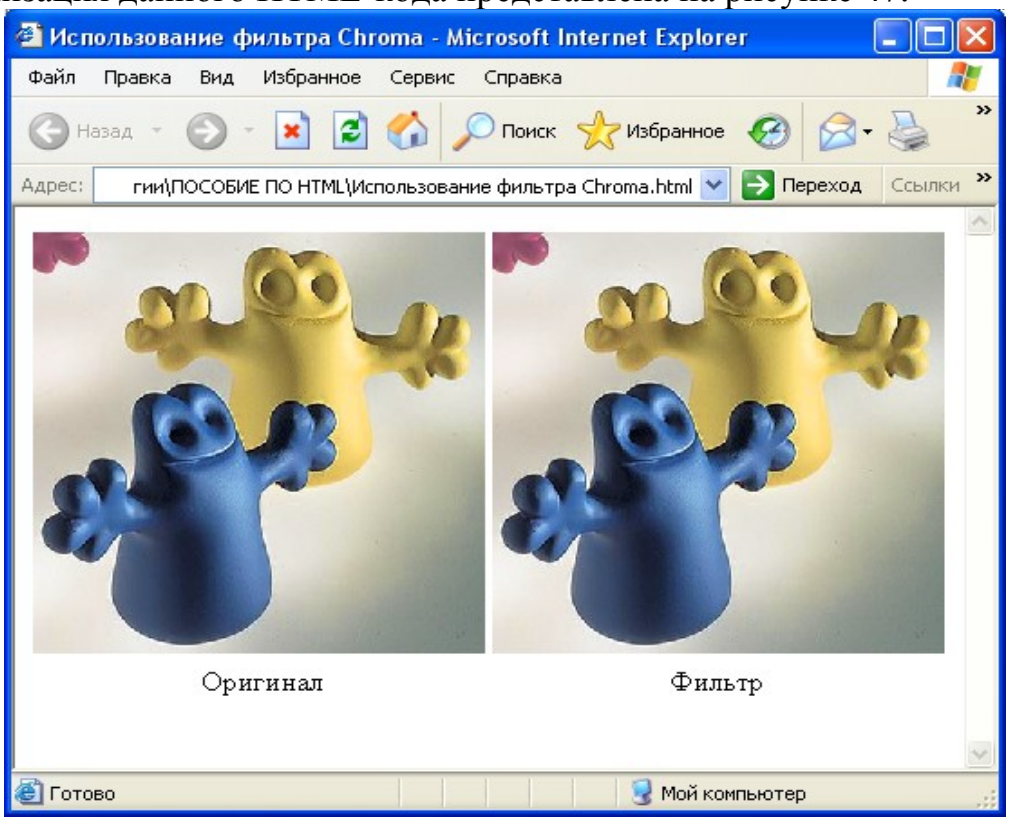

Рисунок 47 – Пример использования фильтра Chroma

6.1.4 DropShadow

Отбрасывает тень у элемента страницы, которая отображается отдельно от самого элемента.

Синтаксис:

Filter: DropShadow (Color=цвет, OffX=горизонтальное смещение, OffY=вертикальное смещение, Positive=тип тени)

Создает копию объекта, цветом Color, смещенную на указанное расстояние. Если Positive=1, то создается копия непрозрачных пикселей объекта, если - 0, то прозрачных.

При настройке этого фильтра можно использовать следующие параметры:

- Color - цвет.
- OffX - горизонтальное смещение тени.
- OffY - вертикальное смещение тени.
- Positive=true (по умолчанию) - тень создается из прозрачных пикселов элемента страницы.
- Positive=false - тень создается из непрозрачных пикселов элемента страницы (“негативная” тень).

- Enabled - разрешение применения фильтра. Имеет два значения: true (по умолчанию) - разрешает применить фильтр false - не разрешает.

Рассмотрим пример использования фильтра DropShadow.

```
<HTML>
<HEAD>
<TITLE>Использование фильтра DropShadow</TITLE></HEAD>
<BODY>
<TABLE>
<TR align=middle>
<TD>
<DIV style= “FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 200px”>SHADOW
</DIV></TD>
<TD> <DIV style= “FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER: DropShadow(Color=#a0a0a0, OffX=5, OffY=5, Positive=true); WIDTH: 200px”>SHADOW
</DIV> </TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR></TBODY></TABLE>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 48.

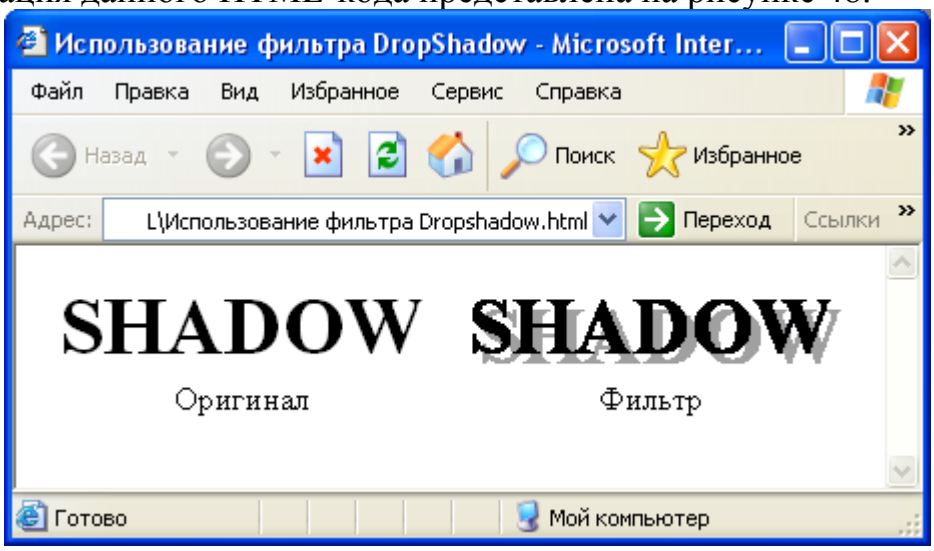

Рисунок 48 – Пример использования фильтра DropShadow

6.1.5 FlipH, FlipV

Синтаксис:
Filter: FlipH
Filter: FlipV

Создает зеркальное отражение объекта относительно горизонтальной (FlipH) или вертикальной (FlipV) оси.

Рассмотрим пример совместной реализации фильтров FlipH и FlipV. Изображение будет изменено относительно горизонтальной и вертикальной оси.

```
<HTML>
<HEAD><TITLE>Использование фильтров FlipV FlipH</TITLE>
</HEAD>
<BODY>
<IMG height=250 alt=original src= “plast.jpg” width=250 border=0>
<IMG style=“FILTER: FlipV FlipH” height=250 alt=FlipVFlipH src=“plast.jpg” ></TD></TR>
<TABLE>
<TR align=middle><TD>
<DIV style= “FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 250px”>FLIP </DIV></TD>
<TD><DIV style=“FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER: FlipV FlipH; WIDTH: 250px”>FLIP </DIV></TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR></TABLE>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 49.

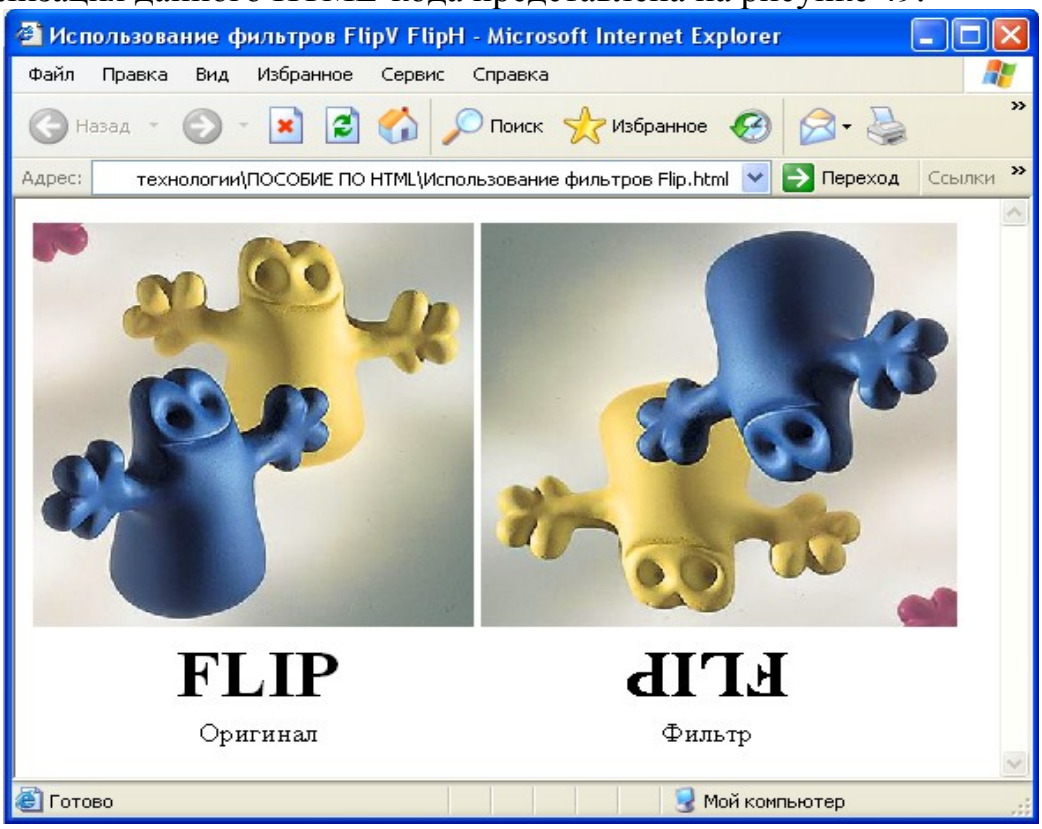

Рисунок 49 – Пример использования фильтров FlipH и FlipV

6.1.6 Glow

Создает эффект “сияния” элемента страницы.

Синтаксис:

Filter: Glow (Color=цвет, Strength=величина)

Создает контур цветом Color вокруг границ объекта.

При настройке этого фильтра можно использовать следующие параметры:

- Color-цвет.
- Strength - расстояние в пикселях, на которое выполняется “тление”.
- Enabled - разрешение применения фильтра. Имеет два значения: true (по умолчанию) - разрешает применить фильтр false - не разрешает.

Рассмотрим пример использования фильтра Glow.

```
<HTML>
<HEAD><TITLE>Использование фильтра Glow</TITLE></HEAD>
<TABLE><TR align=middle>
<TD><DIV style=“FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 200px”>GLOW
</DIV></TD>
<TD><DIV style=“FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER: Glow(Color=#ff0000, Strength=5); WIDTH: 200px”>GLOW</DIV>
</TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR></TBODY></TABLE>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 50.

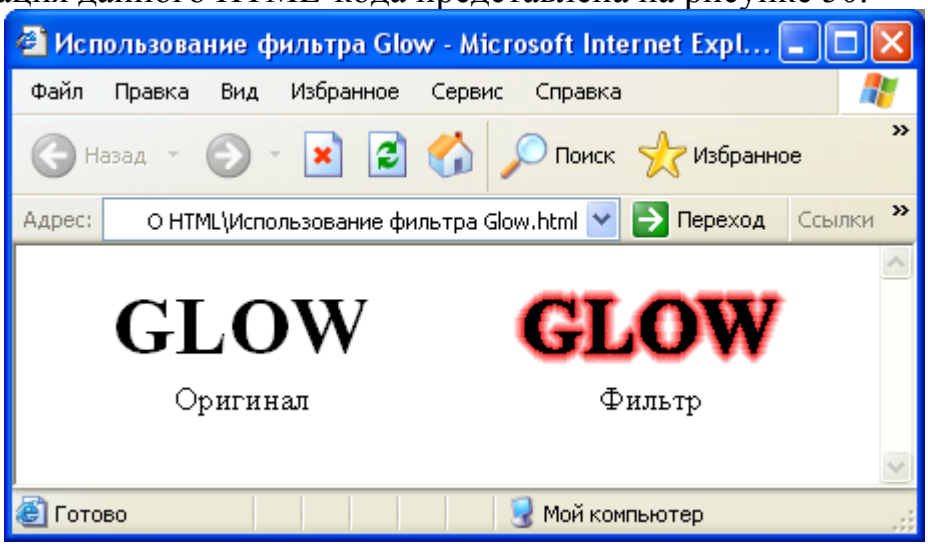

Рисунок 50 – Пример использования фильтра Glow

6.1.7 Invert

Инвертирует пиксели объекта.

Синтаксис:

Filter: Invert

Рассмотрим пример использования фильтра Invert.

```
<HTML>
<HEAD><TITLE>Использование фильтра Invert</TITLE></HEAD>
<BODY>
<IMG alt=original src="plast.jpg" height=250 width=250>
<IMG style="FILTER: Invert" alt=Invert src="plast.jpg" height=250 width=250 >
<TABLE >
<TR align=middle>
<TD><DIV style="FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 250px">INVERT
</DIV></TD>
<TD><DIV style="FONT-WEIGHT: bold; FONT-SIZE: 30pt; BACKGROUND-COLOR: white; FILTER: Invert; WIDTH: 250px">INVERT
</DIV></TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR></TABLE>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 51.

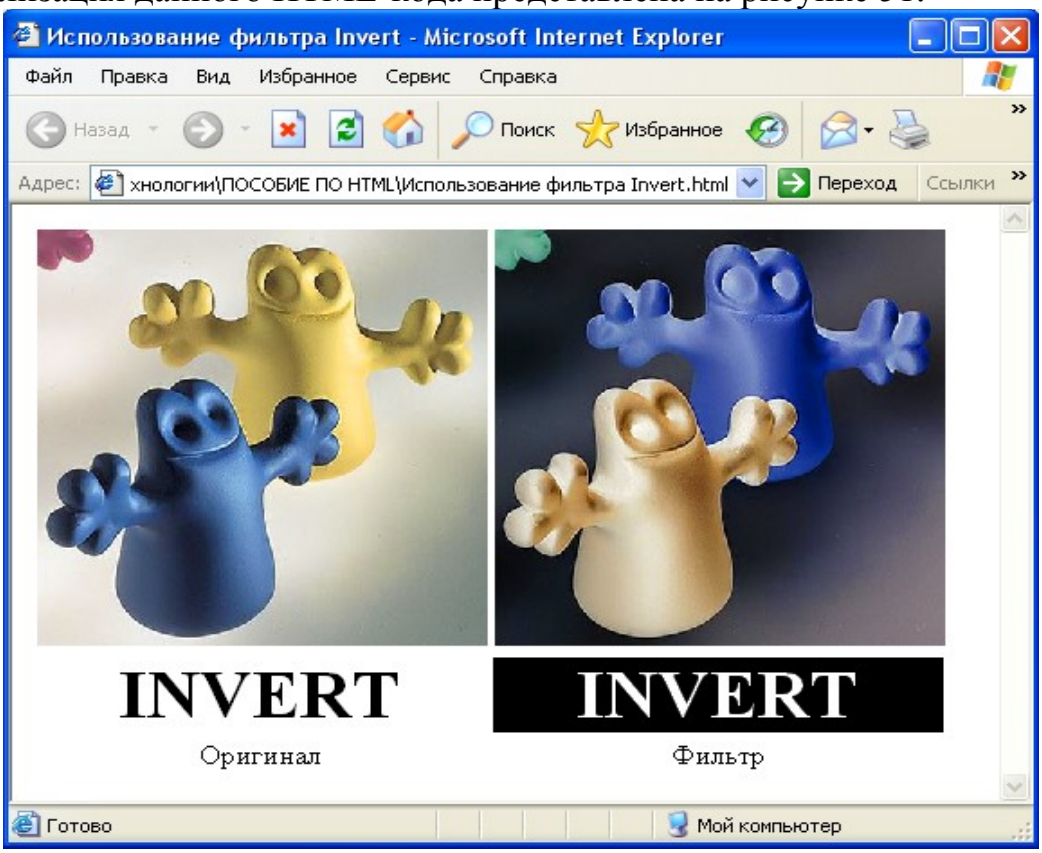

Рисунок 51 – Пример использования фильтра Invert

6.1.8 Gray

Выводит объект в градациях серого цвета, удаляя всю информацию обо всех остальных цветах.

Синтаксис: *Filter: Gray*

Рассмотрим пример использования фильтра Gray.

```
<HTML>
<HEAD>
<TITLE>Использование фильтра Gray</TITLE>
</HEAD>
<BODY>
<IMG height=250 alt=original src="plast.jpg" width=250 border=0></TD>
<IMG style= "FILTER: Gray" height=250 alt=Gray src="plast.jpg" width=250>
<TABLE>
<TR align=middle><TD>
<DIV style= "FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 250px; COLOR: red">GRAY </DIV></TD><TD>
<DIV style="FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER: Gray; WIDTH: 250px; COLOR: red">GRAY
</DIV>
</TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR>
</TABLE>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 52.

Рисунок 52 – Пример использования фильтра Gray

6.1.9 Light

Создает эффект освещенности элемента страницы, добавляет источник света, действующий на объект.

Синтаксис:

Filter: Light

Свойства этого фильтра можно изменять только с помощью скриптовых языков, таких как JavaScript или VBScript.

Методы, применяемые к этому фильтру:

- AddAmbient (iRed, iGreen, iBlue, iStrength) - добавляет рассеянный свет к фильтру. iRed, iGreen, iBlue - цветовые составляющие света (изменяются от 0 до 255). iStrength - интенсивность света (изменяется от 0 до 100).

- AddCone (iX1, iY1, iZ1, iX2, iY2, iRed, iGreen, iBlue, iStrength, iSpread) - добавляет направленный источник света (прожектор). iX1, iY1, iZ1 - координаты источника света, iX2, iY2 - координаты точки направления, iRed, iGreen, iBlue - цветовые составляющие источника света, iStrength - интенсивность светового потока, iSpread - угол между осью и образующей конуса света.

- AddPoint (iX, iY, iZ, iRed, iGreen, iBlue, iStrength) - добавляет рассеянный источник света, iX, iY, iZ - координаты источника света, iRed, iGreen, iBlue - цветовые составляющие источника света, iStrength - интенсивность светового потока.

- ChangeColor (iLightNumber, iRed, iGreen, iBlue, fAbsolute) - изменяет цвет источника света. iLightNumber - идентификатор источника цвета, iRed, iGreen, iBlue - цветовые составляющие источника света, fAbsolute - определяет являются ли цветовые составляющие абсолютными или относительными. Ненулевое значение означает, что составляющие подменят старые. Нулевое означает, что составляющие прибавляются к старым.

- ChangeStrength (iLightNumber, iStrength, fAbsolute) - изменяет интенсивность света. iLightNumber - идентификатор источника цвета, iStrength - интенсивность светового потока, fAbsolute – определяет, является ли интенсивность абсолютной или относительной. Ненулевое значение означает, что интенсивность подменят старую. Нулевое означает, что интенсивность прибавляется к старой.

- Clear () удаляет всю информацию, связанную с фильтром.

- MoveLight (iLightNumber, iX, iY, iZ, fAbsolute) - перемещает источник света новую точку с заданными координатами. iLightNumber - идентификатор источника цвета, iX, iY, iZ - новые координаты, fAbsolute - ненулевое значение означает, что новые координаты замещают старые. Нулевое - указанные числа прибавляются к старым координатам.

- Enabled - разрешение применения фильтра. Имеет два значения: true (по умолчанию) - разрешает применить фильтр false - не разрешает.

Рассмотрим пример использования фильтра Light.

```
<HTML>
<HEAD>
<TITLE>Использование фильтра Light
</TITLE>
</HEAD>
<BODY>
<IMG alt=original  src="plast.jpg" height=250 width=250 border=0>
<IMG id=light style="FILTER: Light" alt=light src="plast.jpg" height=250 width=250>
<SCRIPT>
if (document.all) {
document.all.light.filters.Light.addCone(190,190,0,0,0,255,255,0,50,30);
document.all.light.filters.Light.addAmbient(255,255,255,100); }
</SCRIPT>
<TABLE>
<TR align=middle>
<TD width=250>Оригинал</TD>
<TD width=250>Фильтр</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 53.

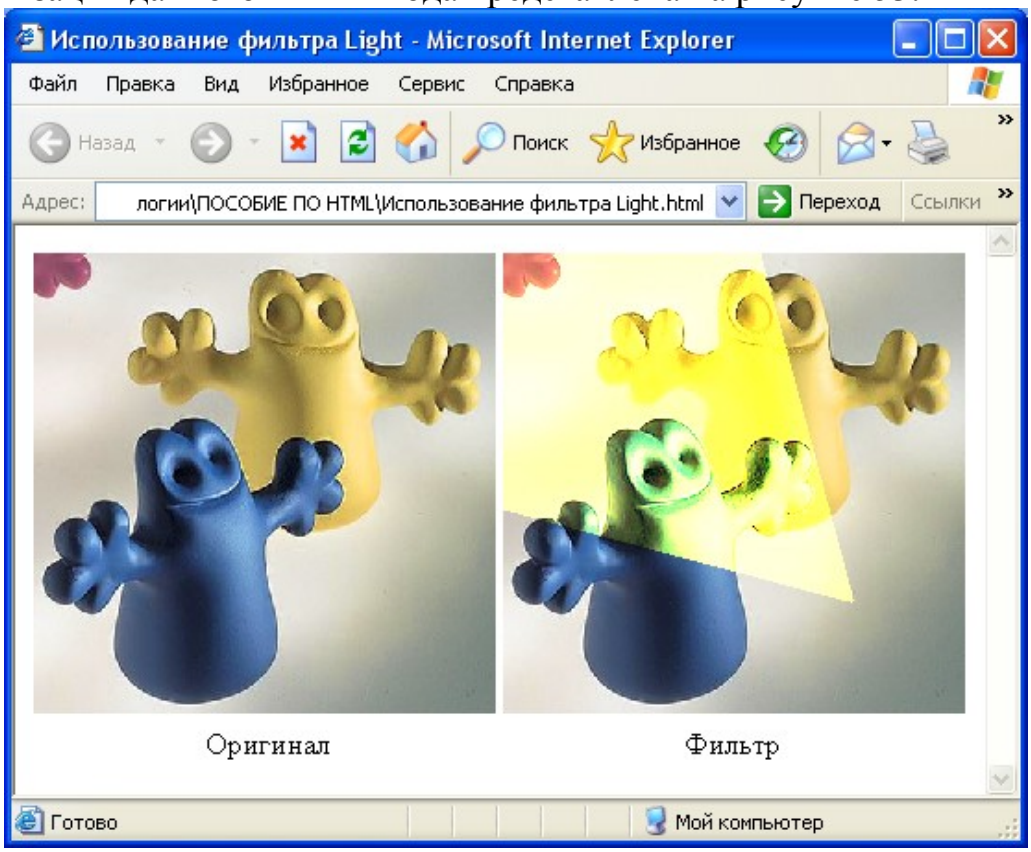

Рисунок 53 – Пример работы фильтра Light

6.1.10 Mask

Отображает прозрачный цвет элемента страницы заданным цветом, т.е. отображает элемент страницы как выделенный.

Синтаксис: *Filter: Mask (Color=цвет)*

Прозрачные пиксели объекта выводятся цветом Color, а непрозрачные - становятся прозрачными.

При настройке этого фильтра можно использовать следующие параметры:

- Color - задает цвет, которым замениться прозрачный цвет элемента страницы.

- Enabled - разрешение применения фильтра. Имеет два значения: true (по умолчанию) - разрешает применить фильтр false - не разрешает.

Рассмотрим пример использования фильтра Mask.

```
<HTML>
<HEAD>
<TITLE>Использования фильтра Mask</TITLE></HEAD>
<BODY>
<TABLE><TR align=middle>
<TD><DIV style= “FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 200px”>MASK
</DIV></TD><TD>
```

<DIV style="FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER: Mask (Color=#000000); WIDTH: 200px">MASK
</DIV></TD></TR><TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR>
</TABLE>
</BODY>
</HTML>

Реализация данного HTML-кода представлена на рисунке 54.

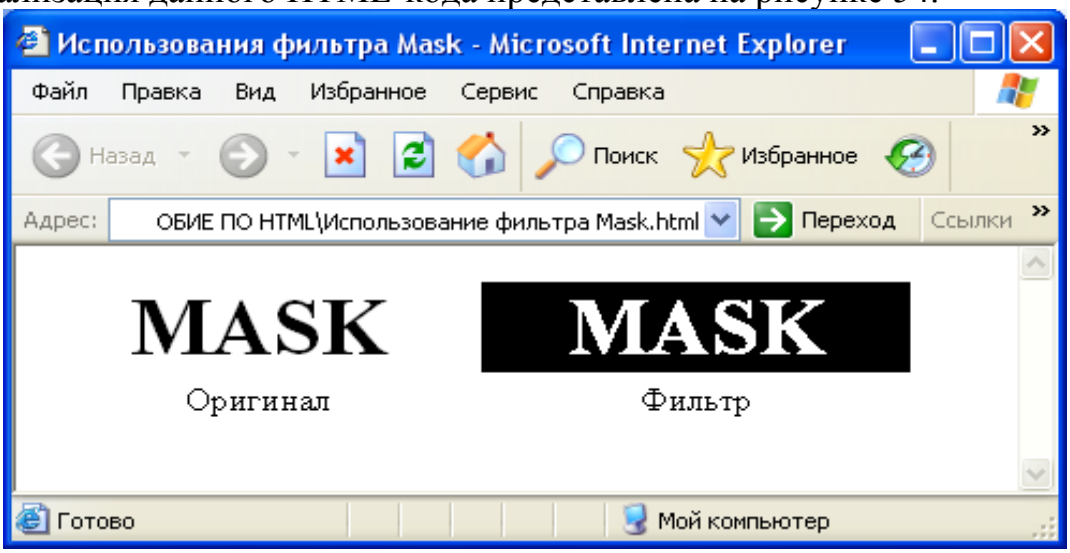

Рисунок 54 – Пример использования фильтра Mask

6.1.11 Shadow

Создает эффект отбрасывания тени у элемента страницы.

Синтаксис:

Filter: Shadow (Color=цвет, Direction=направление)

При этом, создается силует от одной из сторон объекта цветом Color в направлении Direction (задается как и в фильтре Blur).

При настройке этого фильтра можно использовать следующие параметры:

- Color - цвет тени.
- Direction - задает направление тени в градусах, которые должны быть кратны 45.
- Enabled - разрешение применения фильтра. Имеет два значения: true (по умолчанию) - разрешает применить фильтр false - не разрешает.

Рассмотрим пример использования фильтра Shadow.

<HTML>
<HEAD><TITLE>Использование фильтра Shadow</TITLE></HEAD>
<BODY>
<TABLE><TR align=middle>
<TD><DIV style="FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 200px">SHADOW </DIV></TD><TD>

<DIV style="FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER: Shadow (Color=#a0a0a0, Direction=135); WIDTH: 200px">SHADOW
</DIV></TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD></TR>
</TABLE>
</BODY>
</HTML>

Реализация данного HTML-кода представлена на рисунке 55.

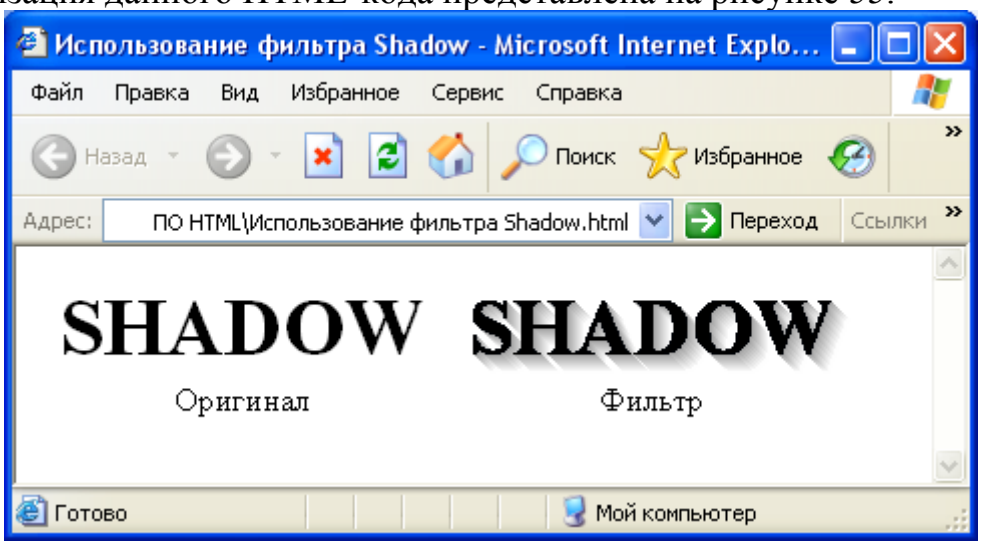

Рисунок 55 – Пример использования фильтра Shadow

6.1.12 Xray

Изменяет глубину цвета и отображает объект в черно-белом варианте. Содержимое Web-страницы напоминает рентгеновский снимок.

Синтаксис:

Filter: Xray

Рассмотрим пример использования фильтра Xray.

<HTML>
<HEAD>
<TITLE>Использование фильтра Xray
</TITLE>
</HEAD>
<BODY>
<IMG height=250 alt=original src="plast.jpg" width=250 border=0>
<IMG style="FILTER: Xray" height=250 alt=xray src="plast.jpg" width=250>
<TABLE>
<TR align=middle>
<TD width=250>Оригинал</TD>
<TD width=250>Фильтр</TD>
</TR>
</TABLE>

</BODY>
</HTML>

Реализация данного HTML-кода представлена на рисунке 56.

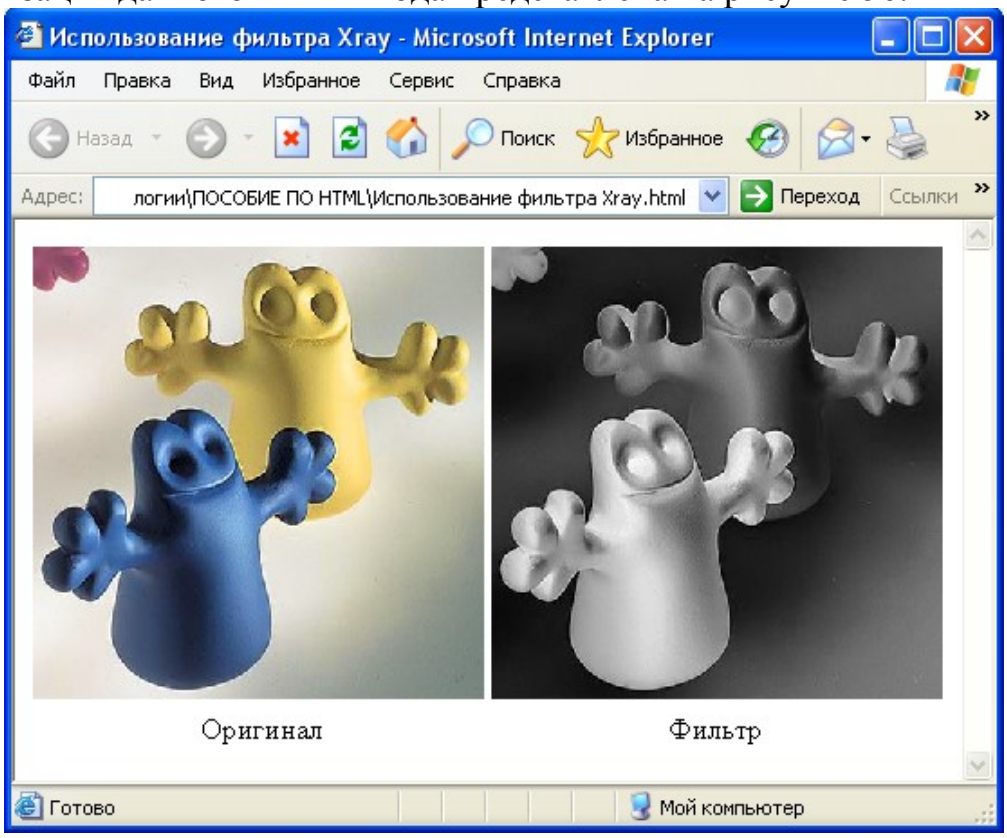

Рисунок 56 – Пример использования фильтра Xray

6.1.13 Wave

Создает волнистое искажение элемента Web-страницы.

Синтаксис:

Filter: Wave (Add=оригинал, Freq=частота, LightStrength=величина, Phase=начальная фаза, Strength=величина)

При этом можно использовать следующие параметры:

- Add - доступны два параметра: true - элемент страницы, к которому был применен фильтр, закроет результат фильтрации, false - результат фильтрации перекроет исходный элемент страницы.
- Freq - задает количество волн.
- LightStrength - окраска волн. Может принимать значение от 0 до 100.
- Phase - начальная фаза, по умолчанию - 0, изменяется от 0 до 100, таким образом, 25 соответствует 90^{o}.
- Strength - величина смещения, дистанция, указанная в пикселах, на которую организуется фильтрация.
- Enabled - разрешение применения фильтра. Имеет два значения: true (по умолчанию) - разрешает применить фильтр false - не разрешает.

Рассмотрим пример использования фильтра Wave.

<HTML>
<HEAD><TITLE>Использование фильтра Wave

```
</TITLE>
</HEAD>
<BODY>
<IMG height=250 alt=original src= “plast.jpg” width=250 border=0>
<IMG style=“FILTER: Wave(Add=false, Freq=5, LightStrength=50, Phase=0, Strength=5)” height=250 alt=wave src="plast.jpg" width=250>
<TR align=middle>
<TD>
<DIV style=“FONT-WEIGHT: bold; FONT-SIZE: 30pt; WIDTH: 250px”>WAVE
</DIV></TD>
<TD><DIV style=“FONT-WEIGHT: bold; FONT-SIZE: 30pt; FILTER: Wave(Add=0, Freq=5, LightStrength=50, Phase=0, Strength=2); WIDTH: 250px”>WAVE
</DIV></TD></TR>
<TR align=middle>
<TD>Оригинал</TD>
<TD>Фильтр</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 57.

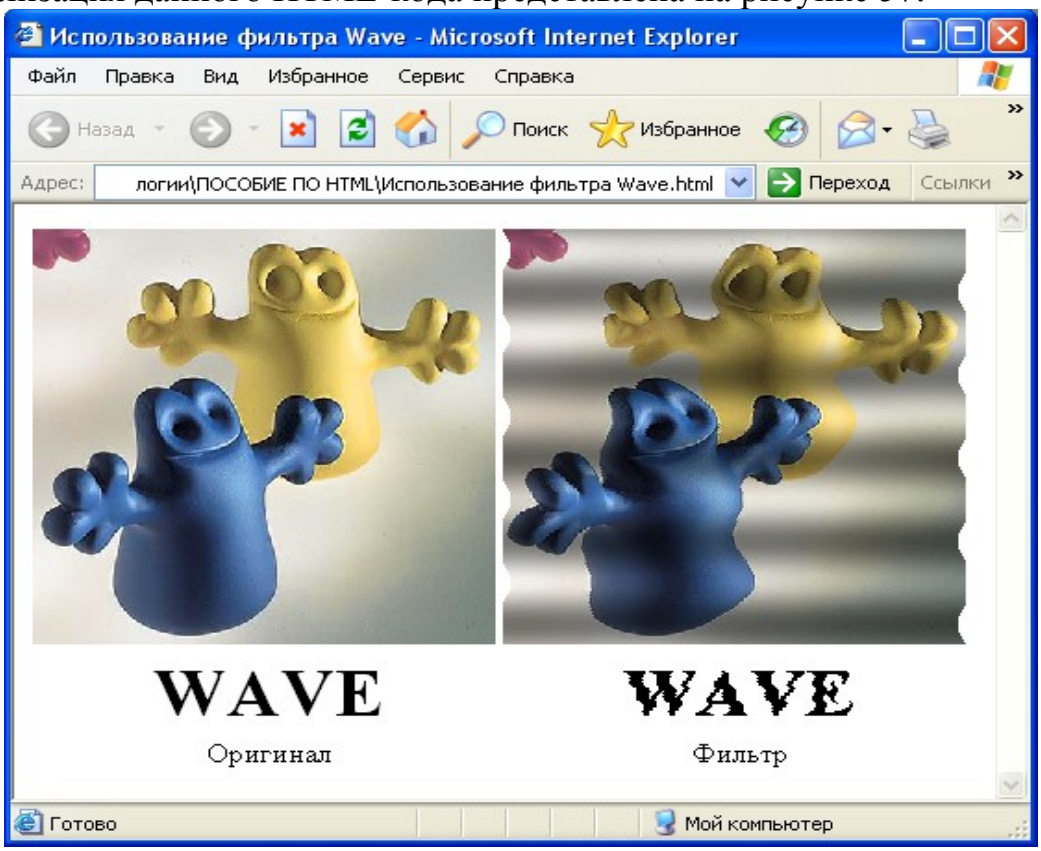

Рисунок 57 – Пример использования фильтра Wave

Если добавить к использованию фильтра Wave обработчик JavaScript, то можно добиться эффекта бесконечного применения фильтра к картинке или тексту, т.е. визуально будет отображаться бесконечная движущаяся волна.

Рассмотрим пример использования элементов языка JavaScript совместно с фильтром Wave.

```
<HEAD>
<TITLE>Wave-фильтр</TITLE>
</HEAD>
<BODY bgcolor=pink>
<SCRIPT language= “javascript”>
if (document.all)
{
var step=3;
var xstep=0;
var msg='Wave-фильтр';
document.write("<div id='water'
style='position:relative;width:370px;height:120px;font-family:Verdana;font-
size:100px;color:#ff0000'>
"+msg+"</div>");
function wave(){
document.all.water.style.filter='wave(freq=3, strength=5, phase='+xstep+',
lightstrength=45, add=0,
enabled=1)';
xstep+=step;
TIMER=setTimeout('wave()',10);}}
wave();
</SCRIPT>
```

Реализация данного HTML-кода представлена на рисунке 58.

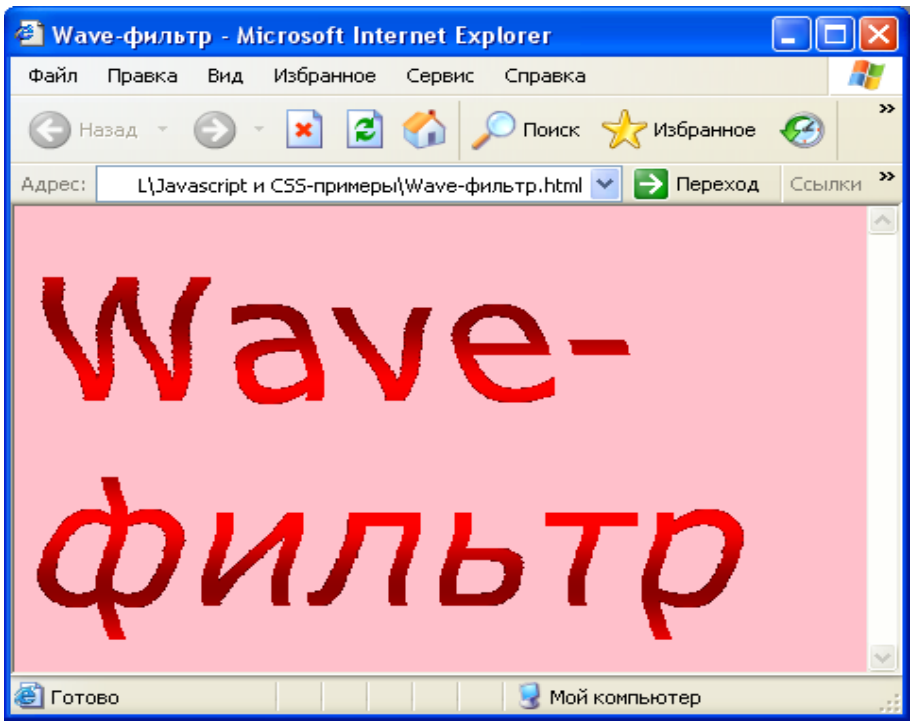

Рисунок 58 - Пример использования фильтра Wave с элементами JavaScript

Изменяет глубину цвета и отображает объект в черно-белом варианте. Должно напоминать рентгеновский снимок	

6.1.14 BasicImage

Устанавливает сразу несколько фильтров для содержимого.

Синтаксис:

Filter: progid:DXImageTransform.Microsoft.BasicImage (GrayScale= {0|1} Invert={0|1} Mask={0|1} MaskColor=цвет Mirror={0|1} Rotation={0|1|2|3} XRay={0|1})

При настройке этого фильтра можно использовать следующие параметры:

- GrayScale - при значении 1 выводит содержимое в серых тонах.
- Invert - при значении 1 инвертирует все цвета содержимого.
- Mask - при значении 1 выводит изображение в виде маски Непрозрачные точки становятся прозрачными. Прозрачные выводятся цветом, заданным в свойстве MaskColor.
- MaskColor - задает цвет для вывода маски.

Значение этого свойства можно задать только на скриптовом языке. Указание его непосредственно в стилях, не приводит ни к каким результатам. Для доступа к свойству можно воспользоваться следующей конструкцией на скриптовом языке JavaScript:

IDOBJ.filters.item(0).MaskColor = 0x00FF00;

Здесь IDOBJ - идентификатор элемента (значение свойства id), item(0) - озачает, что обращение идет к первому установленному для элемента фильтру (если к элементу применяется несколько фильтров, то в скобках указывается номер нужного фильтра по порядку минус один, таким образом, для доступа к свойствам второго фильтра нужно использовать запись item(1).

- Mirror - при значении 1 выводит зеркальное изображение содержимого.
- Rotation - поворачивает содержимое на заданный угол: 0 - 0º, 1 - 90º, 2 - 180º, 3 - 270º.
- XRay - при значении 1 содержимое отображается как “рентгеновский снимок”.

Рассмотрим конкретный пример использования фильтра Basicimage.

```
<HTML>
<HEAD>
<TITLE>Использование фильтра Basicimage</TITLE>
</HEAD>
<BODY>
<TABLE
style=“BORDER-RIGHT: black thin solid; BORDER-TOP: black thin solid; BORDER-LEFT: black thin solid; BORDER-BOTTOM: black thin solid; BORDER-COLLAPSE: collapse”>
<TR>
<TD>
<DIV id=filterDIV style= “BORDER-RIGHT: black 1px solid; PADDING-RIGHT: 5px; BORDER-TOP: black 1px solid; PADDING-LEFT: 5px; FONT-WEIGHT: bold; FONT-SIZE: 15pt; BACKGROUND: gold; FILTER: progid:DXImageTransform.Microsoft.gradient(startColorstr='BLACK',endColorstr='yellowgreen'),
progid:DXImageTransform.Microsoft.BasicImage(Rotation=0,Mirror=0,Invert=0,XRay=0,Grayscale=0,Opacity=1.00,Mask=0),; PADDING-BOTTOM: 5px;
BORDER-LEFT: black 1px solid; WIDTH: 240px; COLOR: white; PADDING-TOP: 5px; BORDER-BOTTOM: black 1px solid; HEIGHT: 180px; TEXT-ALIGN: center”><IMG src= “plast.jpg” width=250 height=250> </DIV></TD>
<TD>  <B>Rotation:</B>
                                                    <SELECT
onchange=“oRotation.innerText=this.options[this.selectedIndex].value;
filterDIV.filters.item(1).rotation=this.options[this.selectedIndex].value”>
<OPTION value=0 selected>0</OPTION>
```

<OPTION value=1>1</OPTION> <OPTION value=2>2</OPTION>
<OPTION
value=3>3</OPTION></SELECT>

  <B>Grayscale:
</B>
<SELECT
onchange="oGrayScale.innerText=this.options[this.selectedIndex].value;
filterDIV.filters.item(1).grayscale=this.options[this.selectedIndex].value">
<OPTION value=0 selected>0</OPTION> <OPTION
value=1>1</OPTION></SELECT>
<B>X-ray:</B> <SELECT
onchange="oXray.innerText=this.options[this.selectedIndex].value;
filterDIV.filters.item(1).xray=this.options[this.selectedIndex].value">
<OPTION value=0 selected>0</OPTION> <OPTION
value=1>1</OPTION></SELECT>

  <B>Mirror:</B>
<SELECT
onchange="oMirror.innerText=this.options[this.selectedIndex].value;
filterDIV.filters.item(1).mirror=this.options[this.selectedIndex].value">
<OPTION value=0 selected>0</OPTION> <OPTION
value=1>1</OPTION></SELECT>
  <B>Invert:</B>
<SELECT
onchange="oInvert.innerText=this.options[this.selectedIndex].value;
filterDIV.filters.item(1).invert=this.options[this.selectedIndex].value">
<OPTION value=0 selected>0</OPTION>
<OPTION value=1>1</OPTION></SELECT>
</TD></TR>
<TR>
<TD style="BORDER-RIGHT: black thin solid; BORDER-TOP: black thin solid; BORDER-LEFT: black thin solid; BORDER-BOTTOM: black thin solid"
colSpan=2><PRE id=oCode style="PADDING-RIGHT: 10px; PADDING-LEFT: 10px; FONT-SIZE: 9pt; BACKGROUND: white; PADDING-BOTTOM: 10px; COLOR: black; PADDING-TOP: 10px">
<SPAN style="FONT-SIZE: 8pt; FONT-FAMILY: verdana">filter:progid:DXImageTransform.Microsoft.BasicImage
(grayscale=<SPAN id=oGrayScale>0</SPAN>, xray=<SPAN id=oXray>0</SPAN>, mirror=<SPAN id=oMirror>0</SPAN>, invert=<SPAN id=oInvert>0</SPAN>, rotation=<SPAN id=oRotation>0</SPAN>)
</SPAN></TD></TR>
</TABLE>
</P>
</DIV>
</BODY>

</HTML>

Реализация данного HTML-кода представлена на рисунке 59. Здесь использованы возможные варианты совместного использования различных фильтров, например Rotation, Invert и Mirror. Таким образом, рисунок повернут на 180 градусов, цвета инвертированы и выведено зеркальное отображение рисунка. Также, при изменении различных фильтров внизу страницы отображается, какие фильтры выбраны в данный момент.

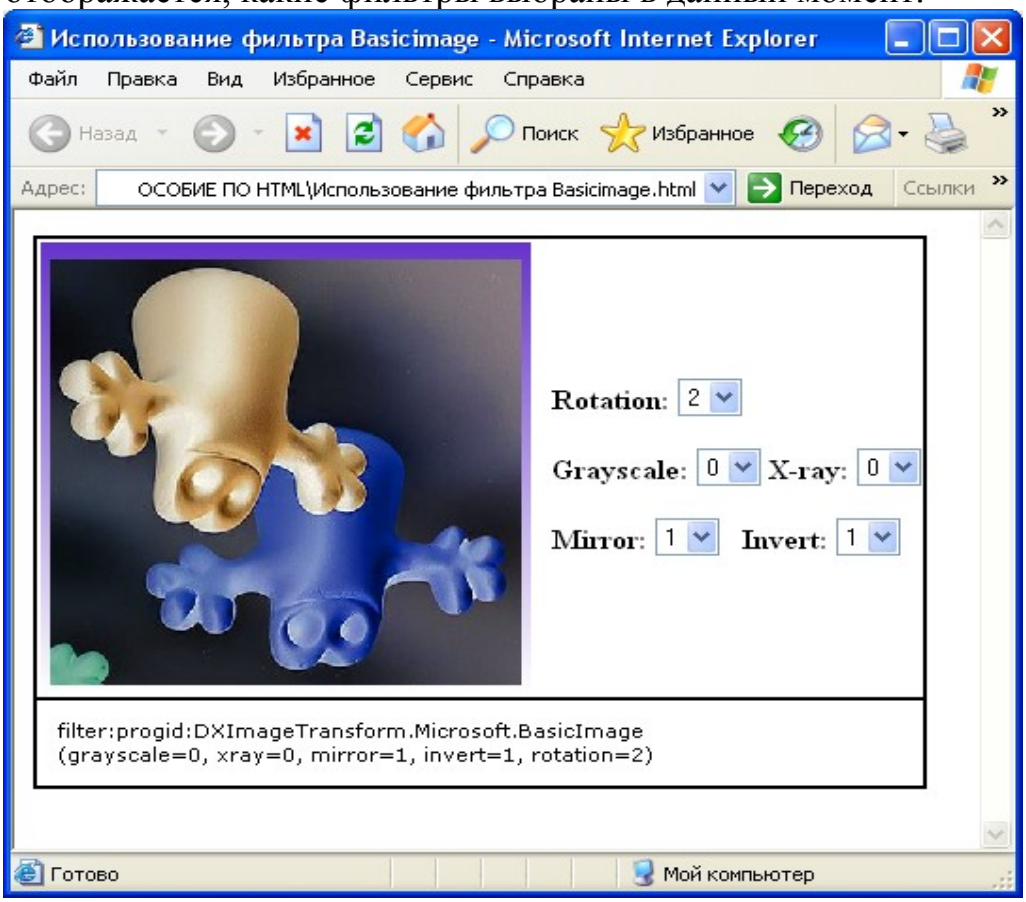

Рисунок 59 – Пример работы фильтра Basicimage

Таким образом, мы рассмотрели наиболее популярные фильтры на сегодняшний день. Для большего сравнения фильтров меду собой приведем пример HTML-страницы, на которой представлены некоторые из приведенных фильтров.

Рассмотрим сравнение фильтров fliph, flipv и blur.

```
<HTML>
<HEAD>
<TITLE>Фильтры </TITLE>
</HEAD>
<STYLE>
P {position: absolute; top:0; left:190; font-size:60}
</STYLE>
<BODY>
<IMG STYLE = “position: absolute; top:0; left:0; width: 140; height:100” src=”happy.jpg”>
```

<IMG STYLE = “position: absolute; top:120; left:10; width: 140; height:100; filter: flipv” src=”happy.jpg”>
<IMG STYLE = “position: absolute; top:230; left:10; width: 140; height:100; filter: fliph” src=”happy.jpg”>
<IMG STYLE = “position: absolute; top:340; left:10; width: 140; height:100; filter: blur" src=”happy.jpg”>
** Фильтр **
**<p style=“top:110; filter:flipv”>Фильтр flipv **
**<p style=“top:220; filter:fliph”>Фильтр fliph **
**<p style=“top:340; filter:blur”>Фильтр blur **
</BODY>
</HTML>

В броузере это будет выглядеть следующим образом:

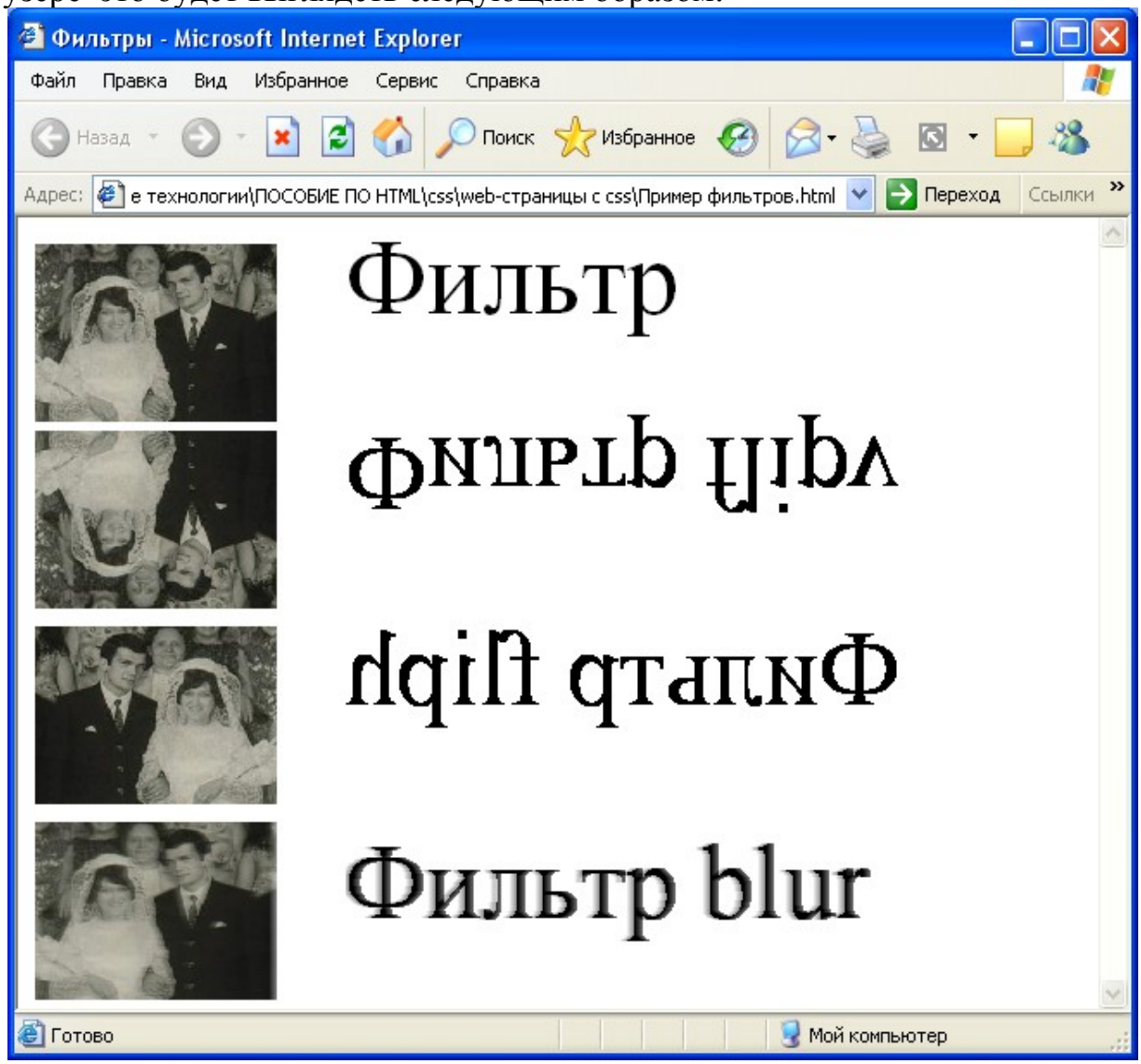

Рисунок 60 – Пример использования фильтров fliph, flipv и blur

В следующем примере рассматривается сравнение фильтров Xray и Wave.

<HTML>
<HEAD><TITLE>Фильтры </TITLE>

```
</HEAD>
<STYLE>
P {position: absolute; top:0; left:190; font-size:48}
</STYLE>
<BODY>
<IMG STYLE = “position: absolute; width: 140; height:100”
src=”happy.jpg”>
<IMG STYLE = “position: absolute; top:120; left:10; width: 140; height:100;
filter: xray” src=”happy.jpg”>
<IMG STYLE =“position: absolute; top:230; left:10; width: 140; height:100;
filter: wave" src=”happy.jpg”>
<p>Фильтр</p>
<p style=”top:130; filter:xray”>Фильтр Xray</p>
<p style=”top:240; filter:wave”>Фильтр Wave</p>
</BODY>
</HTML>
```

В броузере это будет выглядеть следующим образом:

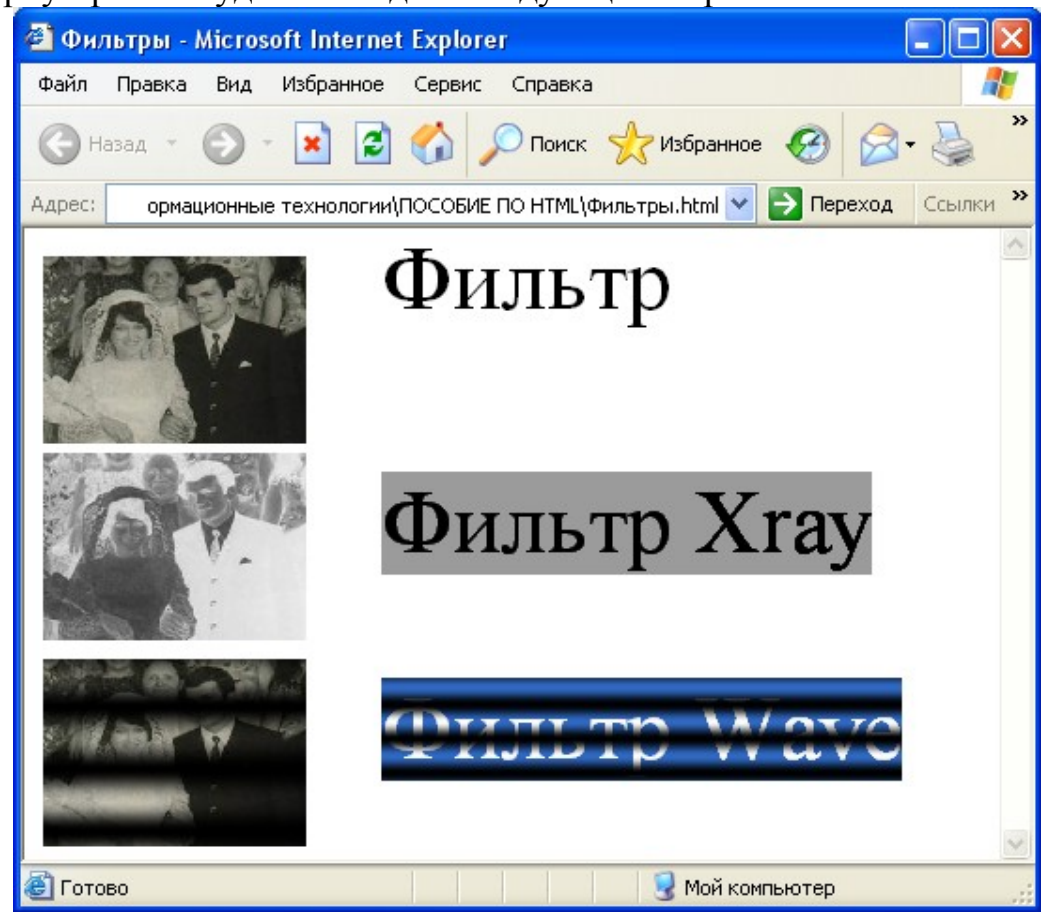

Рисунок 61 - Пример использования фильтров Xray и Wave

6.2 Динамические фильтры

Фильтры преобразования изображений позволяют организовать постепенное появление (исчезновение) изображения, а также трансформацию одного графического объекта в другой. Это так называемые динамические фильтры. В отличии от статических фильтров (например, alpha), эффекты

постепенного появления (исчезновения) графического изображения и преобразования одного изображения в другое можно получить с помощью совместного применения фильтра и сценария /13/.

Суть преобразования графического объекта заключается в том, что сначала необходимо зафиксировать первое изображение, затем выполнить замену этого изображения другим и/или изменить параметры того же самого изображения, а после этого выполнить собственно трансформацию.

Для преобразования изображений используется фильтры revealtrans и blendtrans.

6.2.1 Revealtrans

С помощью фильтра revealtrans задается эффект появления и исчезновения блоков HTML-страницы.

Синтаксис:

Filter: revealtrans

Фильтр revealtrans имеет следующие параметры:

- Duration – длительность преобразования в секундах.
- Transition – способ преобразования (число от 0 до 23).

Продолжительность задается дробным числом, целая часть которого - секунды, дробная - миллисекунды. Типы переходов заранее определены, поэтому требуется лишь указать номер выбранного способа. В таблице 1 приведены тип и номера превращений.

Таблица 1 – Типы воздействия фильтров и их номера

Тип превращения	Номер фильтра

Стягивающийся прямоугольник	0
Расширяющийся прямоугольник	1
Стягивающийся круг	2
Расширяющийся круг	3
Стирание вверх	4
Стирание вниз	5
Стирание вправо	6
Стирание влево	7
Вертикальные жалюзи	8
Горизонтальные жалюзи	9
Сужающиеся клетки	10
Закрывающаяся шахматная доска	11
Случайный наплыв	12
Вертикальное деление внутрь	13
Вертикальное деление наружу	14
Горизонтальное деление внутрь	15
Горизонтальное деление наружу	16
Стирание влево-вниз	17
Стирание влево-вверх	18
Стирание вправо-вниз	19
Стирание вправо-вверх	20
Случайные горизонтальные полосы	21
Случайные вертикальные полосы	22
Случайный выбор номера (из 0-22)	23

Рассмотрим пример использования фильтра Revealtrans.

```
<HTML>
<HEAD>
<TITLE>Пример использования фильтра Revealtrans
</TITLE>
</HEAD>
<BODY onload = “transform()”>
<IMG ID =“myimg” SRC =“minicat.jpg” STYLE =“position:absolute;
top:40; left: 40; filter: revealtrans (duration = 3, transition = 19)”>
<H3 align= “center”>Пример использования фильтра Revealtrans</H3>
</BODY><SCRIPT>
function transform() {
document.all.myimg.style.visibility =“hidden”
myimg.filters (“revealtrans”).apply()
myimg.style.visibility = "visible"
myimg.filters (“revealtrans”).play()}
</SCRIPT>
</HTML>
```

Реализация данной HTML-страницы в броузере представлена на рисунке 62.

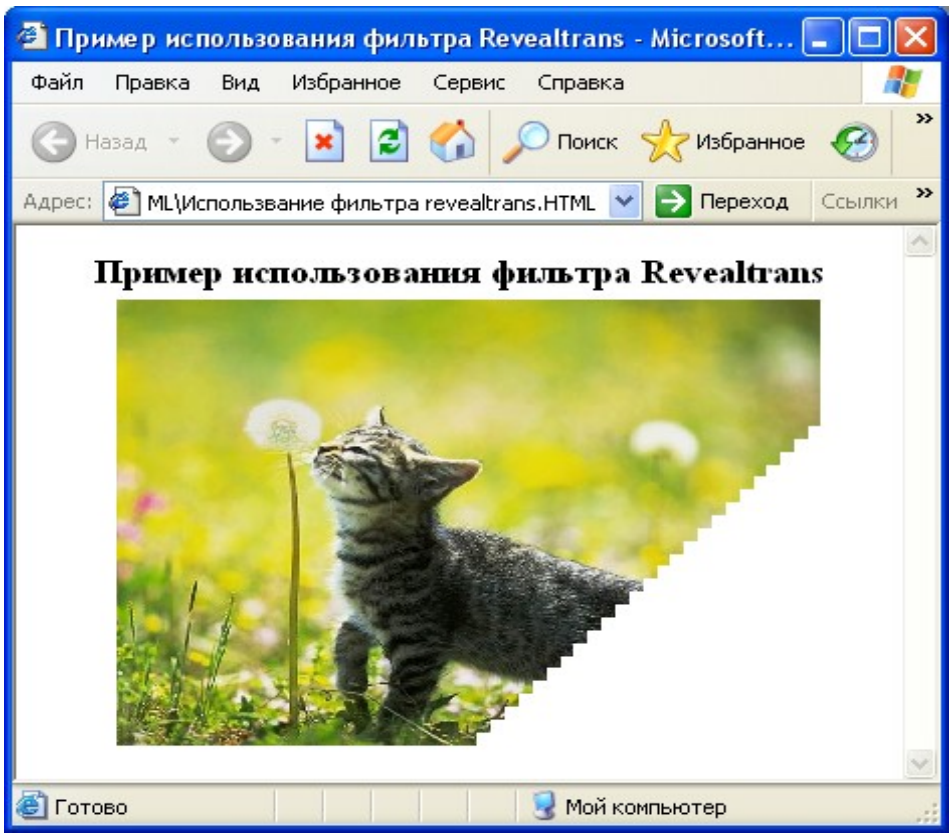

Рисунок 62 - Пример использования фильтра Revealtrans

Для задания параметров запуска и остановки применяются следующие методы фильтров: Apply(), Play(продолжительность), Stop.

Кроме того, для каждого элемента существует событие onFilterChange, которое вызывается, если фильтр запустился, отработан или был изменен. Вызов метода Stop аналогично вызывает событие onFilterChange. Самыми важными свойствами фильтров являются: Transition и Status. Первое задает тип перехода, второе показывает состояние фильтра (0 - остановлен, 1 - назначен, 2 - выполняется).

Фильтр работает при изменении атрибута стиля visibility, т.е. когда необходимо показать спрятанный (hidden) или спрятать видимый (visible) элемент. Каждый элемент содержит коллекцию фильтров *element*.filters. Доступ к фильтру может быть осуществлен по его индексу. Если у элемента определен один фильтр, то доступ к нему осуществляется так: *element*.filters(0). Общая процедура запуска выглядит следующим образом:

element.filters(0).apply;

element.style.visibility =“hidden”; // или “visible”

element.filters(0).play (*продолжительность*);

Если необходимо программировать циклическое повторение фильтров, то необходимо проверить статус фильтра, и если он равен 2, то вызов метода play () приведет к ошибке /14/.

В рассмотренном выше примере, в функции transform (), изображение делается невидимым. Затем, с помощью метода apply () фиксируется изображение как исходный кадр в цепочке кадров преобразования. Далее готовится конечный кадр – изображение делается видимым. Наконец,

применяется фильтр с помощью метода play(). Тип преобразования (transition=20), соответствующий эффекту выезда из левого верхнего угла.

Рассмотрим еще один пример, в котором рисунок постепенно становится видимым через открывающиеся вертикальные жалюзи. Изображение графического файла сначала невидимо (свойство visibility:hidden). В теге DIV применен динамический фильтр revealtrans, управляющий появлением изображений. Сценарий содержит одну функцию dyn_filter(), которую применяет фильтр (метод apply()), делает объект видимым (свойству visibility присваивается значение “visible”), устанавливает тип преобразования (transform=8) и задает длительность преобразования 2с (метод play (), которому передан числовой параметр 2). Далее определено, что эффект постепенного появления картинки начинается сразу же после загрузки страницы. Это достигается привязкой функции dyn_filter() к событию onload в теге <BODY>.

Код страницы выглядит следующим образом:

```
<HTML>
<HEAD>
<TITLE> Динамический фильтр
</TITLE>
<SCRIPT>
function dyn_filter ()
{Image1.filters (0).apply()
I1.style.visibility= “visible”
Image1.filters(0).transition=8
Image1.filters(0).play(2)}
</SCRIPT>
</HEAD>
<BODY ONLOAD= “dyn_filter()”>
<DIV ID=Image1 STYLE=“position:absolute; height:210; width:300; left:30; top:30; filter:revealtrans”>
<IMG ID=I1 STYLE=“position:absolute; height:210; width:300; visibility:hidden" src="first.jpg”>
</DIV>
<H3 STYLE=“position:absolute; top:240; left:30”>
Динамическое преобразование картинки с помощью фильтра</H3>
</BODY>
</HTML>
```

Реализация данной HTML-страницы в броузере представлена на рисунке 63.

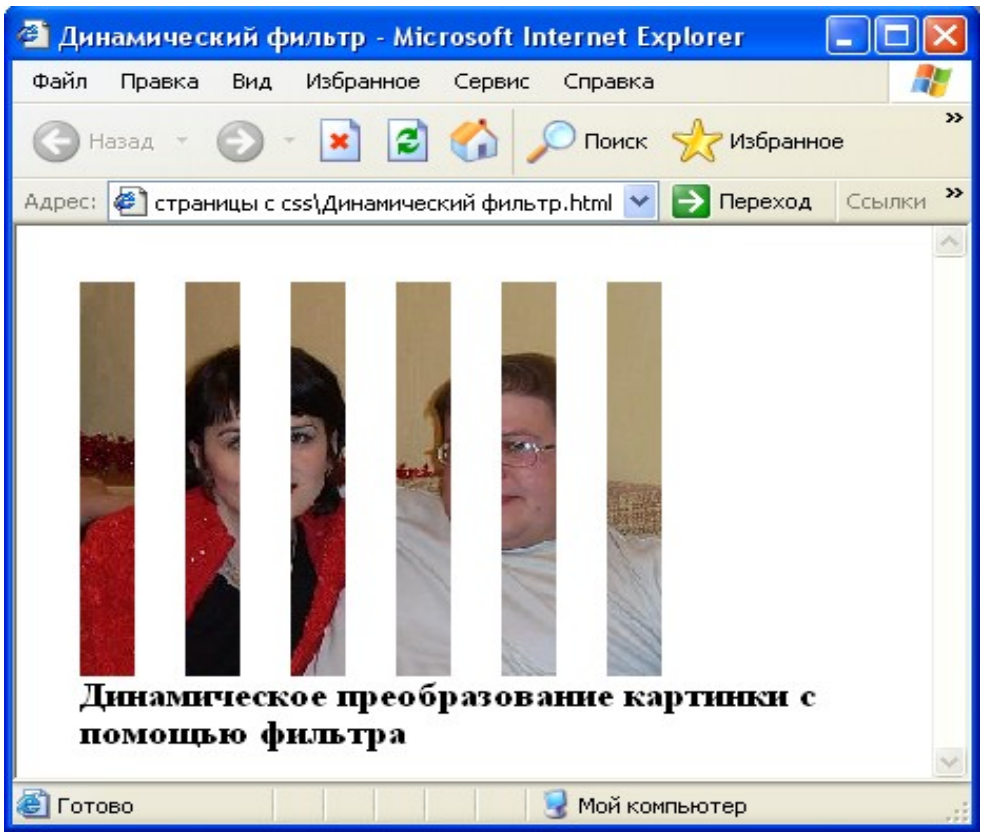

Рисунок 63 - Пример использования динамического фильтра Revealtrans

При этом, Image1 - это имя (идентификатор) элемента-контейнера, заданного тегом DIV. Контейнер содержит единственный элемент (изображение), но он использовался затем, чтобы применить к нему метод apply(), который не поддерживается элементом, созданным тегом <IMG>. Filters - коллекция всех фильтров. Фильтр (единственный) указан номером в коллекции: Image1.filters(0).

6.2.2 Blendtrans

Фильтр blendtrans создает эффект постепенного повышения яркости и насыщенности изображения, фильтр, который осуществляет изменение “видимости” путем изменения прозрачности объекта. Синтаксис:

Filter: blendtrans

Этот фильтр имеет только один параметр – продолжительность преобразования (duration).

Код Web-страницы будет выглядеть следующим образом:

```
<HTML>
<HEAD>
<TITLE> Динамический фильтр-2
</TITLE>
<SCRIPT>
function dyn_filter (){
Image1.filters (0).apply()
I1.style.visibility="visible"
Image1.filters(0).play(2)}
```

</SCRIPT>
</HEAD>
<BODY ONLOAD="dyn_filter()">
<DIV ID=Image1 STYLE="position:absolute; height:210; width:300; left:30; top:30; filter:blendtrans">
<IMG ID=I1 STYLE="position:absolute; height:210; width:300; visibility:hidden" src="first.jpg">
</DIV>
<H3 STYLE="position:absolute; top:240; left:30">Динамическое преобразование картинки с помощью фильтра-2
</H3>
</BODY>
</HTML>

Реализация данной HTML-страницы в броузере представлена на рисунке 64.

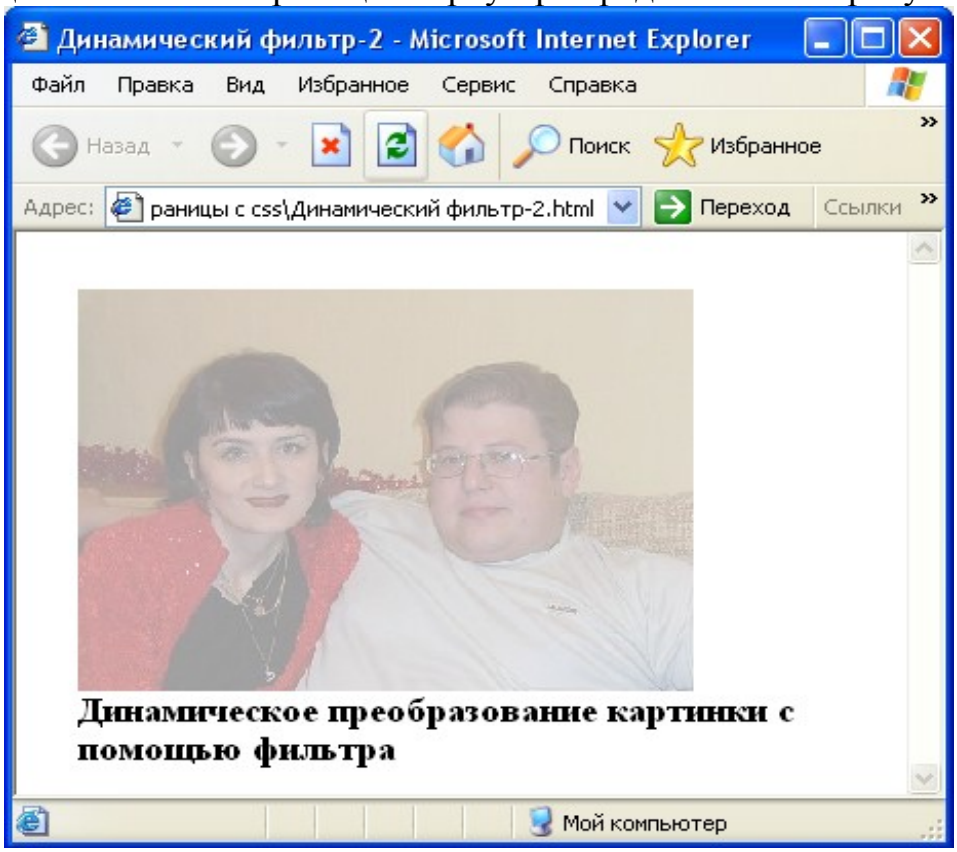

Рисунок 64 - Пример использования динамического фильтра Blendtrans

7 Создание динамических эффектов средствами CSS

В CSS имеются множество возможностей создания динамических эффектов. Наибольшую привлекательность эти эффекты проявляют при использовании совместно с элементами языка JavaScript для обработки различных событий.

7.1 Изменение фона в таблице

Для изменения фона таблицы средствами CSS необходимо написать небольшой обработчик соответствующего события, например, попадание курсора мышки в область таблицы. Раасмотрим HTML-код такой страницы.

```
<HTML>
<HEAD>
<TITLE> Изменение фона в таблице </TITLE>
</HEAD>
<BODY>
<table id=table onMouseover="table.bgcolor="red"" onMouseOut=
"table.bgColor="white"" bgcolor=white border=1>
<tr>
<td>
При наведении курсора мышки на таблицу она поменяет цвет.
</td>
</tr>
</table>
</BODY>
</HTML>
```

Реализация этой HTML-страницы в броузере представлена ниже.

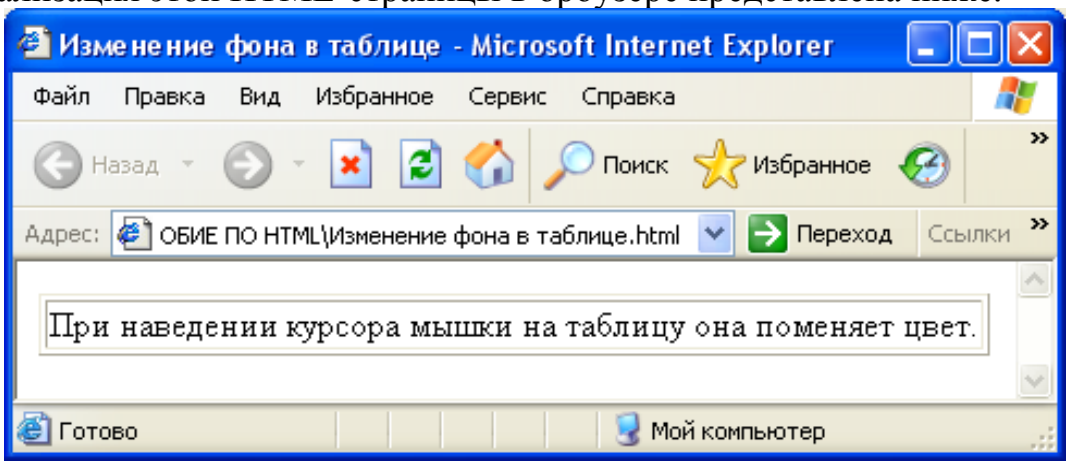

Рисунок 65 - Изменение фона в таблице, неактивное состояние

При наведении мышки на область таблицы она изменит цвет с белого на красный. Соответственно, когда курсор мышки покинет эту область, она вновь станет белой. Аналогичным образом можно менять не только цвет, но и фоновый рисунок таблицы.

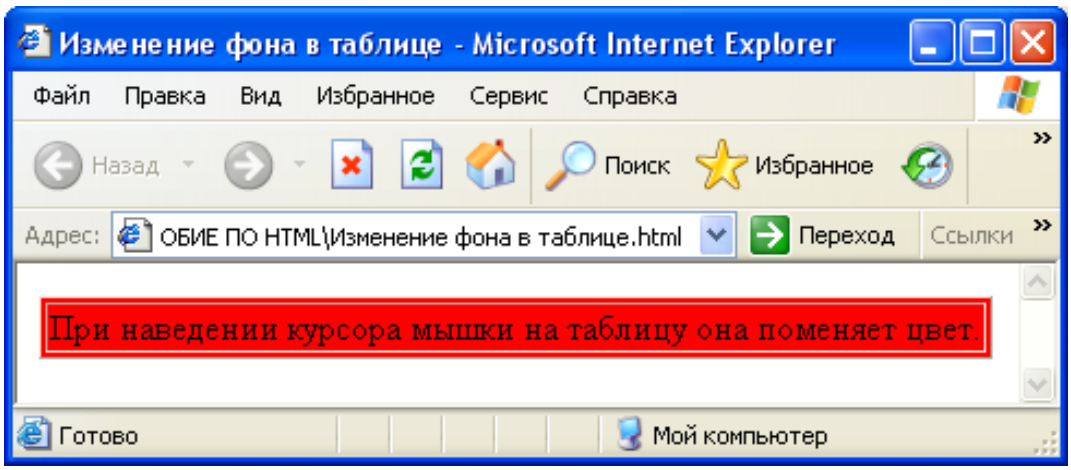

Рисунок 66 - Изменение фона в таблице, активное состояние

7.2 Изменения стиля текста

В CSS имеется возможность смены цвета текста при наведении мыши. При этом используется свойство onmouseover /15/.

В следующем примере цвет текста, заключенного в тег <B>, изменяется с синего на ярко-зеленый при наведении на него указателя мыши:

```
<HTML>
<HEAD>
<TITLE>Изменение стиля текста при наведении мыши</TITLE>
</HEAD>
<H2>Этот <B onmouseover =“colorch ('lime')” onmouseout =“colorch ('navy')” style=“color:navy;”> Текст</B> при наведении на него указателя мыши изменяет цвет
</H2>
<SCRIPT>
function colorch (color) {
event.srcElement.style.color = color}
</SCRIPT>
</HTML>
```

Реализация данного HTML-кода представлена на рисунке 67.

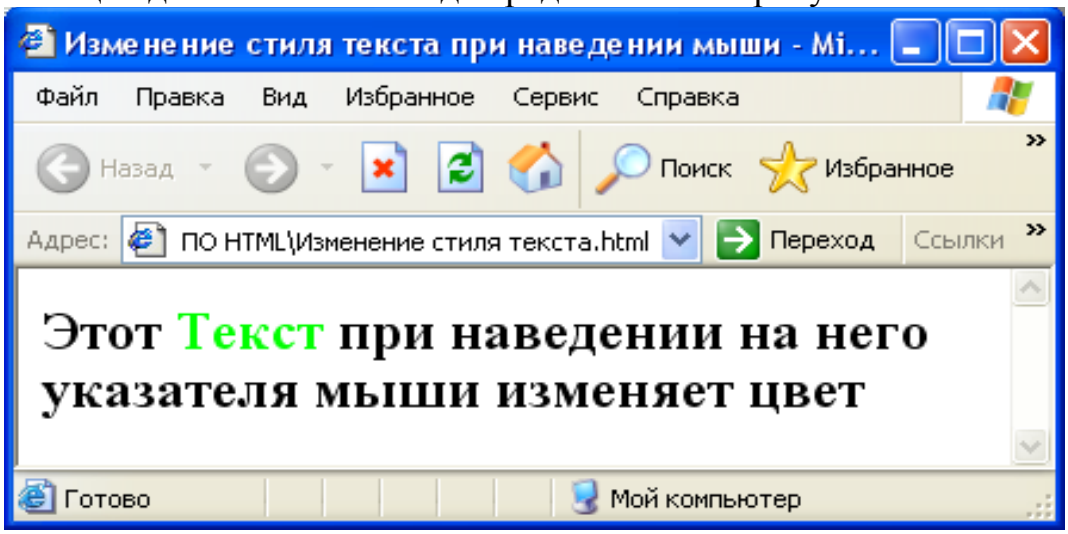

Рисунок 67 – Изменение стиля текста

7.3 Динамические гиперссылки

Для того, чтобы сделать обычные текстовые ссылки реагирующими на наведение мышки, но не использовать обработчик событий JavaScript, в CSS существуют так называемые псевдоклассы ссылок. Выглядят они следующим образом:

a:link - непосещенная ссылка;
a:visited - посещенная ссылка;
a:active - активная (нажатая) ссылка;
a:hover - ссылка, над которой находится курсор мышки.

Рассмотрим пример Web-страницы с использованием псевдоклассов:

```
<HTML>
<HEAD>
<TITLE>
Пример псевдоклассов ссылок
</TITLE>
<STYLE>
body {color: green; font-family: Arial, sans-serif; font-size: 12pt; }
a:link {color: maroon; }
a:visited {color: #CC0000; }
a:active {color: #FF6666; }
a:hover {color: blue; }
</STYLE>
</HEAD>
<BODY>
<P>Посетите <a href="http://www.Rambler.ru/">
 RAMBLER</a>,
чтобы узнать все обо всем
</P>
</BODY>
</HTML>
```

Реализация этой HTML-страницы в броузере представлена на рисунках 68 и 69.

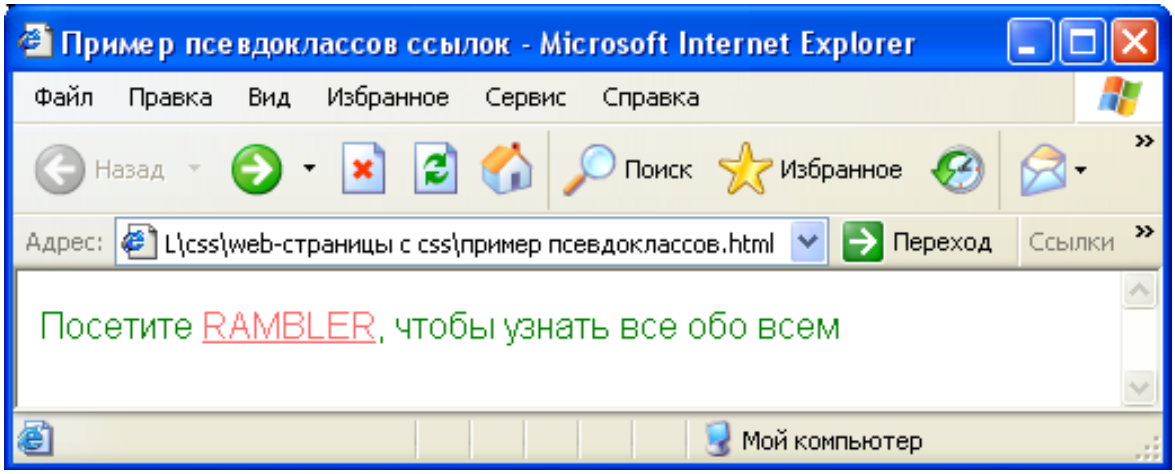

Рисунок 68 – Пример неактивной ссылки

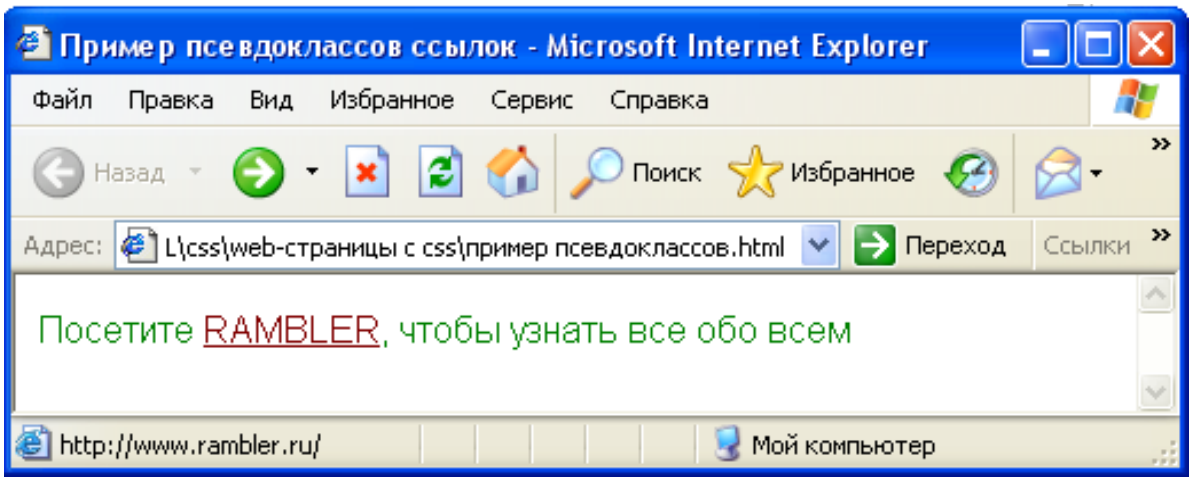

Рисунок 69 – Пример активной ссылки

Для того, чтобы одна группа ссылок подсвечивалась одним цветом, а другая другим, необходимо несколько иначе написать таблицу стилей:

```
<HTML>
<HEAD>
<TITLE>Подсветка ссылок</TITLE>
<STYLE>
body a:Hover {color : red;}
.group1 {text-decoration: none; font-family: arial; font-size: 18pt; color: maroon}
a:hover.group1 {font-family: arial, font-size: 18pt; color: blue}
.group2 {font-family: arial, font-size: 10pt; color: green}
a:hover.group2 {font-family: arial, font-size: 10pt; color: gray}
</STYLE>
</HEAD>
<BODY><P><A class="group2" href ="1.html"> Гиперссылка первого класса</A><P>
<A class="group1" href ="2.html"> Гиперссылка второго класса </A>
</BODY>
</HTML>
```

Так будет выглядеть эта HTML-страница в броузере:

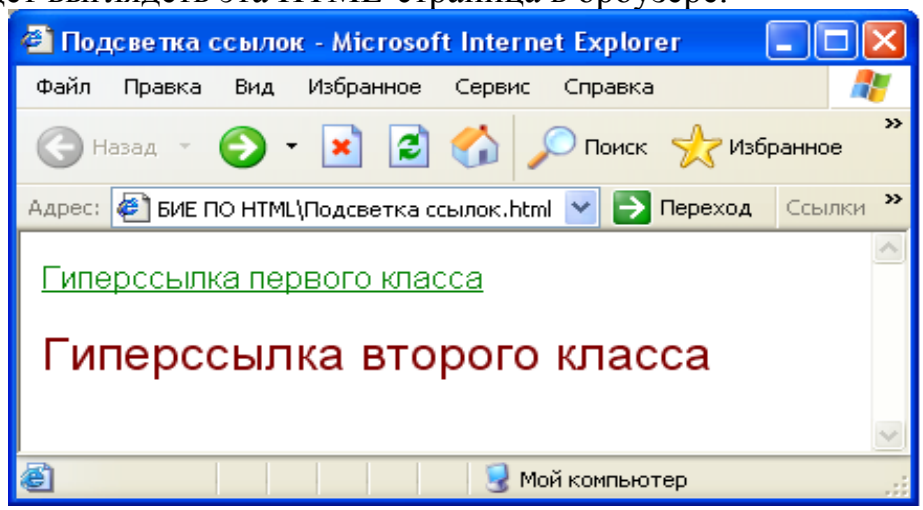

Рисунок 70 - Подсветка ссылок

В этом случае ссылки, для которых указан класс group1, будут отображаться неподчеркнутыми, бордовым, шрифтом семейства Arial размером 18pt. При наведении мышки они станут подчеркнутыми и перекрасятся в синий цвет.

А ссылки, для которых указан класс group2, будут подчеркнутыми и отображаться шрифтом размера 10pt зеленым цветом. При наведении мышки они станут серыми.

7.4 Изменение цвета полосы прокрутки

Оформление полосы прокрутки значительно отличается от классической схемы, т.е. от традиционного серого цвета. Цвет полосы меняется в соответствии с общей цветовой гаммой сайта. Рассмотрим свойства, позволяющие изменить вид полосы прокрутки, - каждое из них отвечает за цвет отдельного элемента.

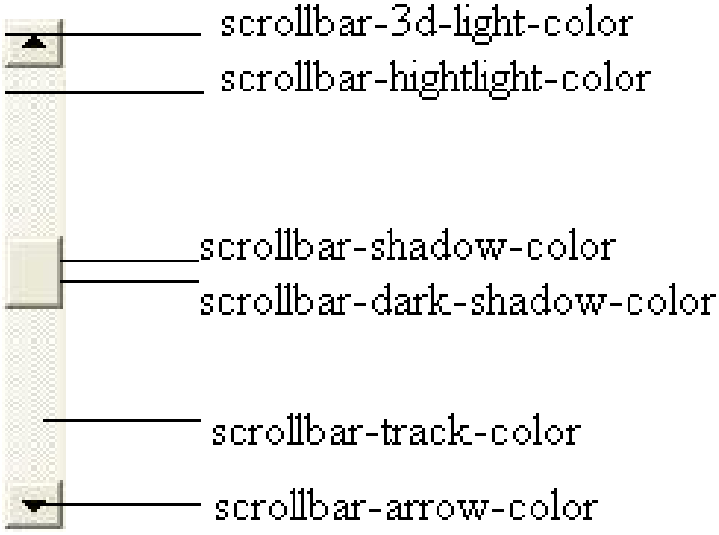

Рисунок 71 - Свойства для задания стиля полосы прокрутки

Свойства:

- scrollbar-3d-light-color – цвет блика ползунка, который располагается на левой и верхней сторонах бегунка и стрелок (по умолчанию белый),
- scrollbar-arrow-color – цвет стрелок прокрутки (по умолчанию черный),
- scrollbar-base-color – основной цвет самой полосы (по умолчанию серый),
- scrollbar-dark-shadow-color – цвет для тени, которая располагается на правой и нижней сторонах бегунка и стрелок (по умолчанию черный),
- scrollbar-face-color – основной цвет стрелок и бегунка (по умолчанию серый),
- scrollbar-hightlight-color – цвет блика на полосе прокрутки,
- scrollbar-shadow-color – цвет тени на полосе прокрутки,
- scrollbar-track-color – цвет фона полосы прокрутки /16/.

Приведем пример переопределения стиля для полосы прокрутки.

```
<HTML>
<HEAD> <TITLE>Стиль полосы прокрутки</TITLE>
<STYLE>
BODY {scrollbar-3d-light-color: #DDA0DD;
scrollbar-arrow-color: #0000ff;
scrollbar-base-color: #FF0000;
scrollbar-dark-shadow-color: #ff0000;
scrollbar-face-color: #87CEEB;
scrollbar-hightlight-color: #F9FDDF;
scrollbar-shadow-color: #800000;
scrollbar-track-color: #EE82EE;}
</STYLE>
</HEAD>
<BODY bgcolor=pink> <H2> Пример изменения стиля для полосы прокрутки средствами CSS</H2>
</BODY>
</HTML>
```

Реализация данной HTML-страницы в броузере представлена на рисунке 72.

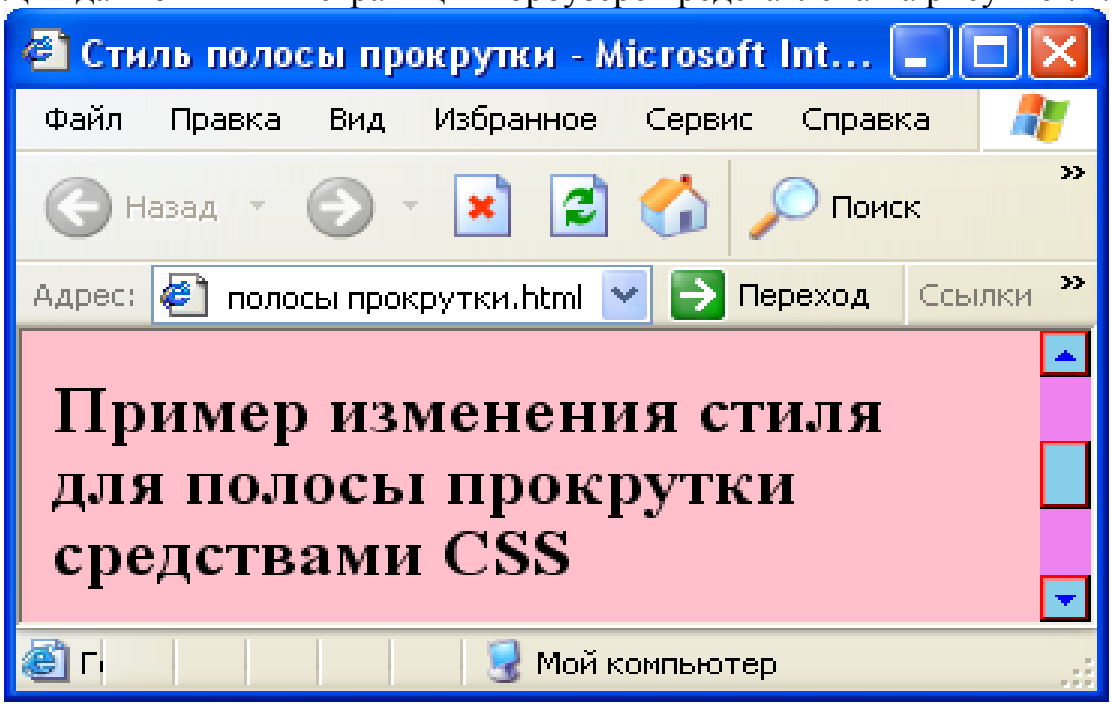

Рисунок 72 – Пример изменения стиля полосы прокрутки

7.5 Изменение стиля курсора

В CSS помимо изменения стиля полосы прокрутки имеется возможность изменения типа курсора.

При этом, стиль курсора можно задать, либо используя графический файл, либо зарезервированное слово, отображающее начертание курсора.

Синтаксис переопределения стиля с использованием зарезервированных слов, следующий:

Style=“cursor:name”

Если же используются графические файлы в качестве указателя курсора, то необходимо указать имя файла, например:

Style=“cursor:url(Cursor1.cur)”

Рассмотрим пример HTML-страницы с использованием различных типов курсоров. Первый пример - использование в качестве курсора графических файлов.

```
<HTML>
<HEAD><TITLE>Курсоры</TITLE>
</HEAD>
<BODY>
<H3>Стиль курсора можно задавать, используя графические файлы</H3>
<TABLE border=2>
<TR height=30>
<TD style=“cursor:url(Cursor1.cur)”><B>Cursor1.cur</B></TD>
<TD><img src=“Cursor1.jpg” height=30 width=30</TD></TR>
<TR height=30>
<TD style=“cursor:url(Cursor2.cur)”><B>Cursor2.cur</B></TD>
<TD><img src=“Cursor2.jpg” height=30 width=30</TD></TR>
<TR height=30>
<TD style=“cursor:url(Cursor3.cur)”><B>Cursor3.cur</B></TD>
<TD><img src=“Cursor3.jpg” height=30 width=30</TD></TR>
<TR height=30>
<TD style=“cursor:url(Cursor4.cur)”><B>Cursor4.cur</B></TD>
<TD><img src=“Cursor4.jpg” height=30 width=30</TD></TR>
<TR height=30>
<TD style=“cursor:url(Cursor5.jpg)”><B>Cursor6.ani</B></TD>
<TD><img src=“Cursor5.jpg” height=30 width=30</TD></TR>
<TR height=30>
<TD style=“cursor:url(Cursor7.cur)”><B>Cursor7.cur</B></TD>
<TD><img src=“Cursor6.jpg” height=30 width=30</TD></TR>
</TABLE>
</BODY>
</HTML>
```

Реализация данной HTML-страницы в броузере реализована на рисунке 73.

Второй пример – использование в качестве курсоров зарезервированных слов.

```
<HTML>
<HEAD><TITLE>Курсоры</TITLE>
</HEAD>
<BODY>
<h3>Стиль курсора можно задавать, используя зарезервированные имена</h3>
 <TABLE border=2>
```

<TR height=30>
<TD style="cursor:crosshair">Графическое выделение
</TD><TD>Тип курсора <B>Crosshair</B></TD></TR>
<TR height=30><TD
style="cursor:move">Перемещение</B></TD><TD>Тип курсора <B>
Move </B></TD></TR>
<TR height=30><TD style="cursor:e-resize"> Стрелка
вправо</B></TD><TD>Тип курсора <B>E-resize</B></TD></TR>
<TR height=30><TD style="cursor:n-resize"> Стрелка
вверх</B></TD><TD>Тип курсора <B>N-resize</B></TD></TR>
<TR height=30><TD style="cursor:text">Выделение
текста</B></TD><TD>Тип курсора <B>Text</B></TD></TR><TR
height=30>
<TD style="cursor:wait">Система недоступна</B></TD><TD>Тип
курсора
<B>Wait</B></TD></TR>
<TR height=30><TD style="cursor:help">Выбор
справки</B></TD><TD>Тип курсора
<B>Help</B></TD></TR>
</TABLE>
</BODY>
</HTML>

Реализация данной HTML-страницы в броузере реализована на рисунке 74.

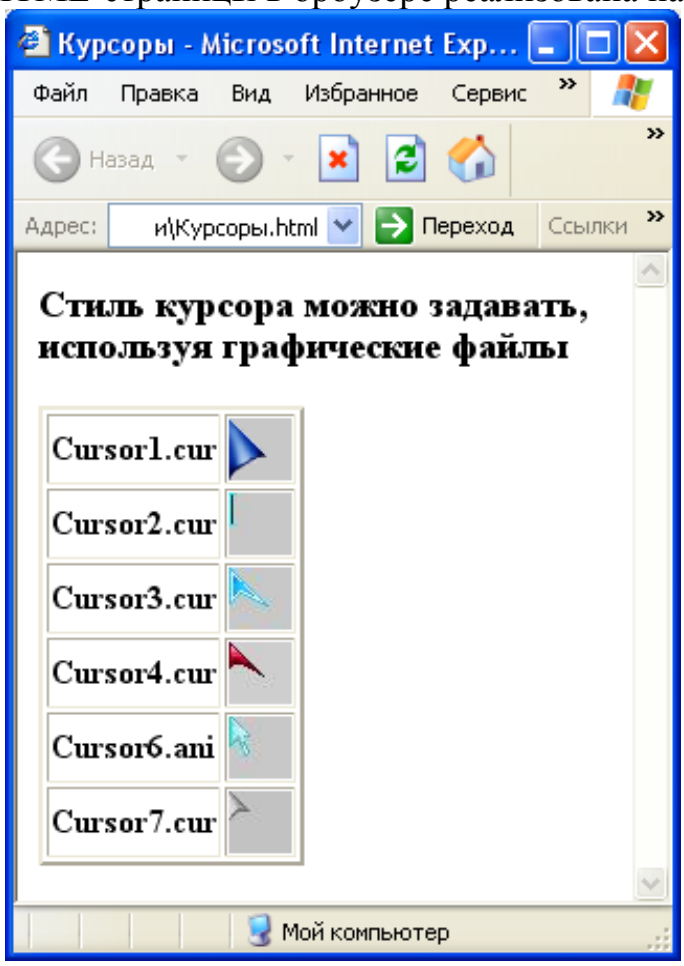

Рисунок 73 – Использование в качестве курсора графических файлов

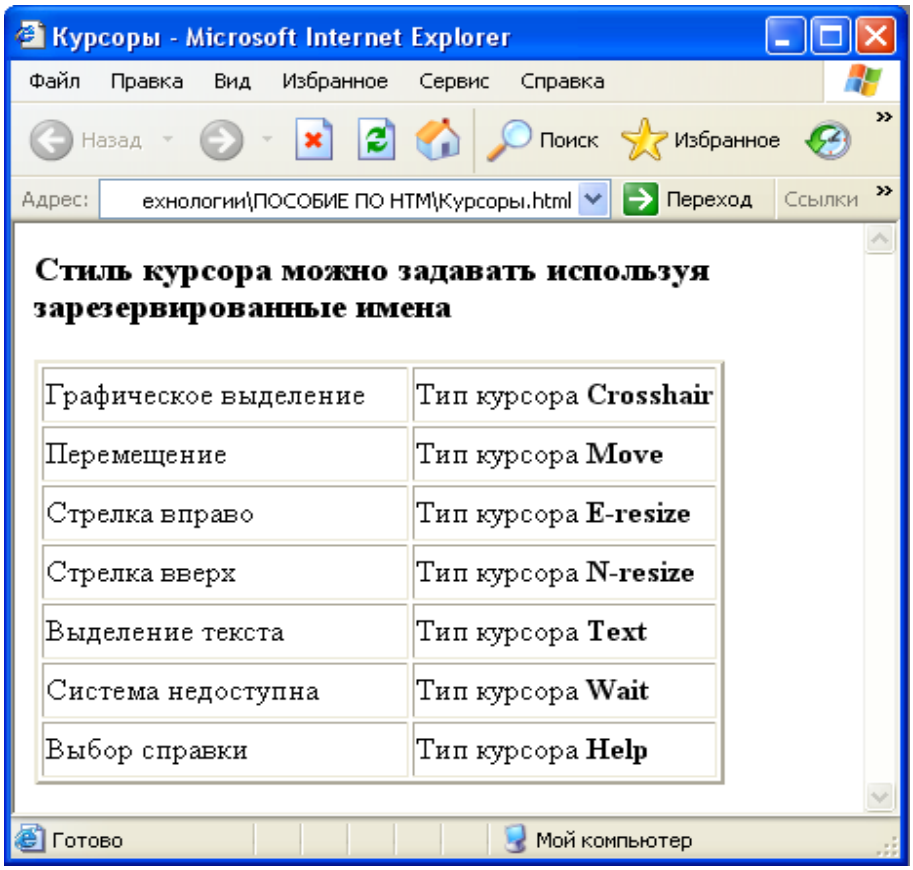

Рисунок 74 – Использование в качестве курсоров зарезервированных слов

7.6 Создание ссылок-кнопок

С помощью таблиц стилей можно превратить обычные ссылки в кнопки, т.е. они будут простыми текстовыми ссылками, а отображаться в броузере будут как кнопки.

Задается стиль для ссылок в обычном состоянии, имитирующий выпуклую кнопку. Определяется фон, рамка и цвет текста. Эффект выпуклости достигается подбором цвета границ. Когда кнопка выпуклая, ее верхняя и левая границы должны быть светлее по тону, чем нижняя и правая, когда кнопка выбирается (на нее наводится курсор), цвета границ меняются и кажется, что происходит “нажатие” кнопки.

Проиллюстрируем пример HTML-страницы, использующей ссылки-кнопки.

```
<HTML>
<HEAD>
<TITLE>Кнопки-гиперссылки</TITLE>
<STYLE>
A {
Background: #0099FF;
Border-top: 1px solid #D5D5D5;
Border-left: 1 px solid #D5D5D5;
Border-bottom: 1 px solid #4E4E4E;
```

```
Border-right: 1px solid #4E4E4E;
Color: #FFFF99;
Font: 0.8 em Tahoma, sans-serif;
Text-decoration: none;
Padding: 2px 5px;
}
A:hover{
Border-top: 1px solid #4E4E4E;
Border-left: 1 px solid #4E4E4E;
Border-bottom: 1 px solid #D5D5D5;
Border-right: 1px solid #D5D5D5;
Background: #FF0000;
}
</STYLE>
</HEAD>
<BODY>
<A HREF="#">Главная</A>
<A HREF="#">О компании</A>
<A HREF="#">Продукция</A>
<A HREF="#">Галерея</A>
</BODY>
</HTML>
```

Реализация данной HTML-страницы в броузере представлена на рисунке 75.

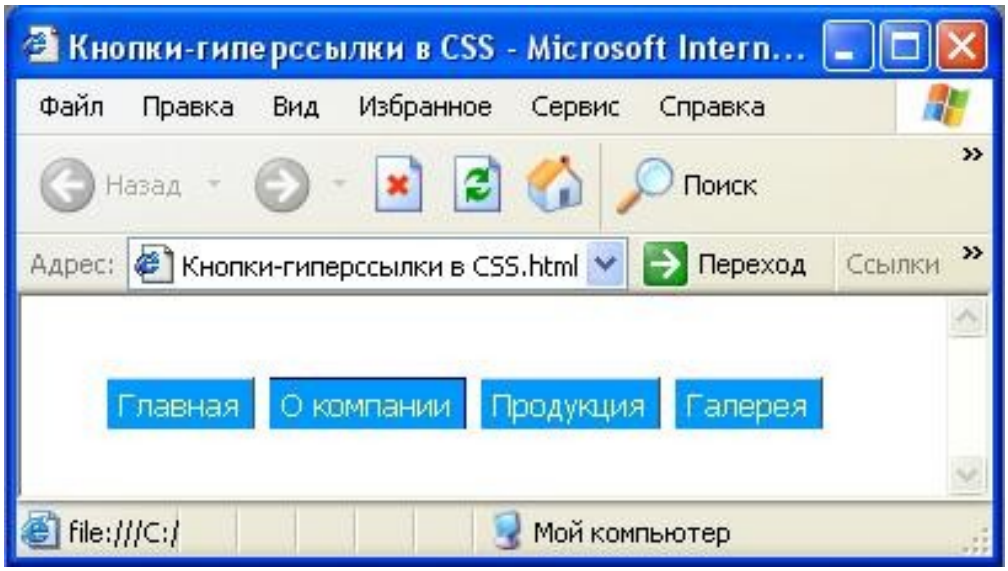

Рисунок 75 - Кнопки-гиперссылки

7.7 Подсветка кнопок и текста

В CSS имеется возможность переопределять стиль для элементов форм. В частности, можно менять цвет, начертание у таких элементов управления формами как однострочное текстовое поле, выпадающее меню, кнопки и т.д.

Рассмотрим пример переопределения стиля для элемента формы - кнопка. Заданы три элемента, задающие кнопки, они находятся в контейнере формы <FORM>. К этому контейнеру привязываются обработчики событий onmouseover (наведение указателя мыши) и onmouseout (удаление указателя мыши). При наведении курсора мыши на любую кнопку она меняет цвет с желтого на синий.

```
<HTML>
<HEAD>
<TITLE>Подсветка кнопок и текста</TITLE>
<STYLE>
mystyle { font-weight: bold;background - color:"blue"}
</STYLE>
</HEAD>
<FORM onmouseover =“colorch('blue')” onmouseout =“colorch('yellow')”>
<INPUT TYPE =“BUTTON" VALUE =“Первая” CLASS =“mystyle”>
<INPUT TYPE =“BUTTON" VALUE=“Вторая” CLASS =“mystyle”)”>
<INPUT TYPE =“BUTTON" VALUE=“Третья” CLASS =“mystyle”>
</FORM>
<SCRIPT>
function colorch(color) {                    //изменение цвета кнопок
if (event.srcElement.type ==“button”)
event.srcElement.style.backgroundColor = color;
}
</SCRIPT>
</HTML>
```

Реализация данной HTML-страницы а броузере представлена на рисунке 76.

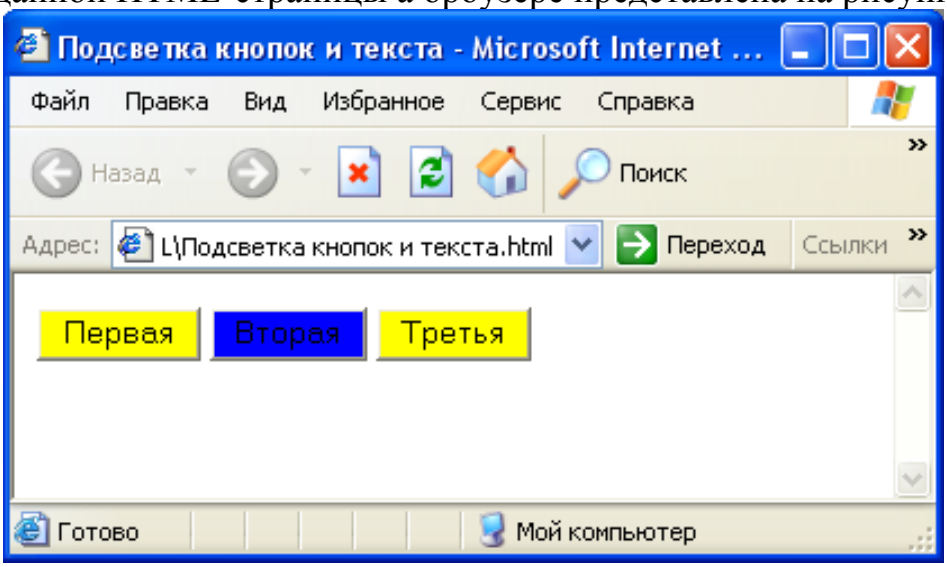

Рисунок 76 - Подсветка кнопок и текста

7.8 Выпадающее меню

Для создания динамического выпадающего меню используется особое свойство CSS-слои. Создавая слой, четко определяются его параметры, и позиционируется в определенное место на экране /17/. Таким образом, блок

имеет три координаты, две координаты x и y, которые определяют положение блока на плоскости экрана, третья, пространственная координата z, которая определяет номер слоя, на котором находится блок.

Рассмотрим пример работы со слоями на примере выпадающего меню. Web-страница посвящена кино и содержит пять пунктов меню. Каждый пункт, да и само меню имеет свой собственный стиль, заданный селекторами CSS. При наведении курсора мыши на каждый пункт меню, появляется подменю, соответствующее данному пункту, т.е. обрабатывается событие onmouseover. Обработка производится с помощью скритов языка Javascript.

Ниже приводится полный HTML-код данной страницы.

```
<HTML>
<HEAD>
<TITLE>Выпадающее Меню</TITLE>
<STYLE TYPE= “text/css”>
BODY{
font-family: Tahoma, sans-serif;
margin: 0 px;}
#head{
height:50px;
width: 100%;
position:absolute;
left: 0 px;
top: 0 px;}
#head H1{
color: navy;
margin-top:0 em;
padding-right: 0.5em;
text-align: right;}
#menu{
background: #C0C0C0;
border-bottom: 1 px solid;
border-top: 1 px solid;
padding: 2px 10 px;
position: absolute;
top:50px;
width:100%;}
#menu A{
color: navy;
font: bold 13px Tahoma, sans-serif;
text-decoration: none;
margin-left:3px;}
#content{
font-size: 0.8em;
padding: 0.5em;
position: absolute;
```

```
left: 0 px; top: 80px;}
#content H1{
color: navy;
font-size: 1.4em;
margin-top: 0 em;
padding-right: 0.5em;
text-align: right;}
#main{
background:#D9D9EC;
border: 2px solid navy;
width:80px;
position: absolute;
left: l0px;
top: 74px;
visibility: hidden;
z-index:1;}
#main A, #movie A, #issue A, #survey A, #forum A{
color: navy;
font-size: 13px;
display: block;
padding: 0px 3px;
text-decoration: none;}
#main A: hover, #movie A:hover, #issue A:hover, #survey A:hover, #forum
A:hover{
color: #FFF;
background:#6C6CB7;}
#movie{
background:#D9D9EC;
border: 2px solid navy;
width:92 px;
position: absolute;
left: 92px;
top: 74px;
visibility: hidden;
z-index:2; }
#issue{
background: #D9D9EC;
border: 2px solid navy;
width:80px;
position:absolute;
left: 195 px;
top: 74 px;
visibility: hidden;
z-index:3; }
#survey{
```

```
background:#D9D9EC;
border: 2px solid navy;
width:90px;
position: absolute;
left: 275 px;
top: 74 px;
visibility: hidden;
z-index:4; }
#forum{
background: #D9D9EC;
border: 2px solid navy;
width:80px;
position: absolute;
left: 370px;
top: 74px;
visibility: hidden;
z-index:5; }
</STYLE>
<SCRIPT TYPE="text/javascript">
function Hidall() {
document.getElementById(“main”).style.visibility='hidden'
document.getElementById(“movie”).style.visibility='hidden'
document.getElementById(“issue”).style.visibility='hidden'
document.getElementById(“survey”).style.visibility='hidden'
document.getElementById(“forum”).style.visibility='hidden'}
function Main () {
document.getElementById(“main”).style.visibility='visible'
document.getElementById(“movie”).style.visibility='hidden'
document.getElementById(“issue”).style.visibility='hidden'
document.getElementById(“survey”).style.visibility='hidden'
document.getElementById(“forum”).style.visibility='hidden'}
function Movie () {
document.getElementById(“main”).style.visibility='hidden'
document.getElementById(“movie”).style.visibility='visible'
document.getElementById(“issue”).style.visibility='hidden'
document.getElementById(“survey”).style.visibility='hidden'
document.getElementById(“forum”).style.visibility='hidden'}
function Issue () {
document.getElementById(“main”).style.visibility='hidden'
document.getElementById(“movie”).style.visibility='hidden'
document.getElementById(“issue”).style.visibility='visible'
document.getElementById(“survey”).style.visibility='hidden'
document.getElementById(“forum”).style.visibility='hidden'}
function Survey () {
document.getElementById(“main”).style.visibility='hidden'
```

```
document.getElementById(“movie”).style.visibility='hidden'
document.getElementById(“issue”).style.visibility='hidden'
document.getElementById(“survey”).style.visibility='visible'
document.getElementById(“forum”).style.visibility='hidden'}
function Forum () {
document.getElementById(“main”).style.visibility='hidden'
document.getElementById(“movie”).style.visibility='hidden'
document.getElementById(“issue”).style.visibility='hidden'
document.getElementById(“survey”).style.visibility='hidden'
document.getElementById(“forum”).style.visibility='visible'}
</SCRIPT>
</HEAD>
<BODY topmargin=“0” leftmargin=“0” rightmargin=“0”
 bottommargin=“0” onLoad=“HideAll()”>
<DIV ID= “head” onmouseover=“Hideall ()”><H1>Все о кино</H1>
</DIV>
<DIV ID=“menu”>
<A HREF=“#” onmouseover=“Main( )” >Главная    | |   </A>
<A HREF=“#” onmouseover=“Movie( )” >Фильмы    | |   </A>
<A HREF=“#” onmouseover=“Issue ( )” >Статьи    | |  </A>
<A HREF=“#” onmouseover=“Survey ( )” >Обзоры    | | </A>
<A HREF=“#” onmouseover=“Forum ( )”>Форум </A>
</DIV>
<DIV id=“content” onmouseover=“Hidall()”>
<H1>X-Men 3</H1>
<P>20th Century Fox и Marvel Studios сдвинулись с места в деле “Люди X 3”. Новый сценарий пишет Саймон Кинберг (“Мистер и Миссис Смит”, “Электра”, “Фантастическая четверка”). Кинберг сообщил, что сюжет пока находится в разработке.
20th Century Fox и Marvel Studios сдвинулись с места в деле “Люди X 3». Новый сценарий пишет Саймон Кинберг (“Мистер и Миссис Смит”, “Электра”, “Фантастическая четверка”). Кинберг сообщил, что сюжет пока находится в разработке. “Мне понравились 2 первых фильма сериала, — сказал Кинберг. — Я и раньше работал с Fox и Marvel над другими проектами, и когда они предложили мне поработать над этим проектом, я с удовольствием согласился. Ведь я же вырос на этом комиксе. Я люблю их всей душой. Это будет лучший фильм в трилогии. Это будет нашим “Возвращение короля”</P>
</DIV>
<DIV id=“main”>
<A HREF=“#”>Новости</A>
<A HREF=“#”>Карта сайта</A></DIV>
<DIV id=“movie”>
<A HREF=“#”>Комедии</A>
<A HREF=“#”>Мелодрамы</A>
```

```
<A HREF="#">Триллеры</A>
<A HREF="#">Ужасы</A>
<A HREF="#">Фантастика</A>
<A HREF="#">Боевики</A>
<A HREF="#">Мультфильмы</A>
</DIV>
<DIV id="issue" name="issue">
<A HREF="#">О кино </A>
<A HREF="#">Об актерах </A>
<A HREF="#">О съемках </A>
</DIV>
<DIV id= "survey" name= "survey">
<A HREF="#">Кино-новинки </A>
<A HREF="#">DVD-новинки </A>
</DIV>
<DIV id="forum" name= "forum">
<A HREF="#">Регистрация</A>
</DIV>
</BODY>
</HTML>
```

Необходимо отметить, что расположенные на отдельных слоях блоки подменю исходно накладываются друг на друга. Пример реализации данного HTML-кода представлен на рисунках 77-81.

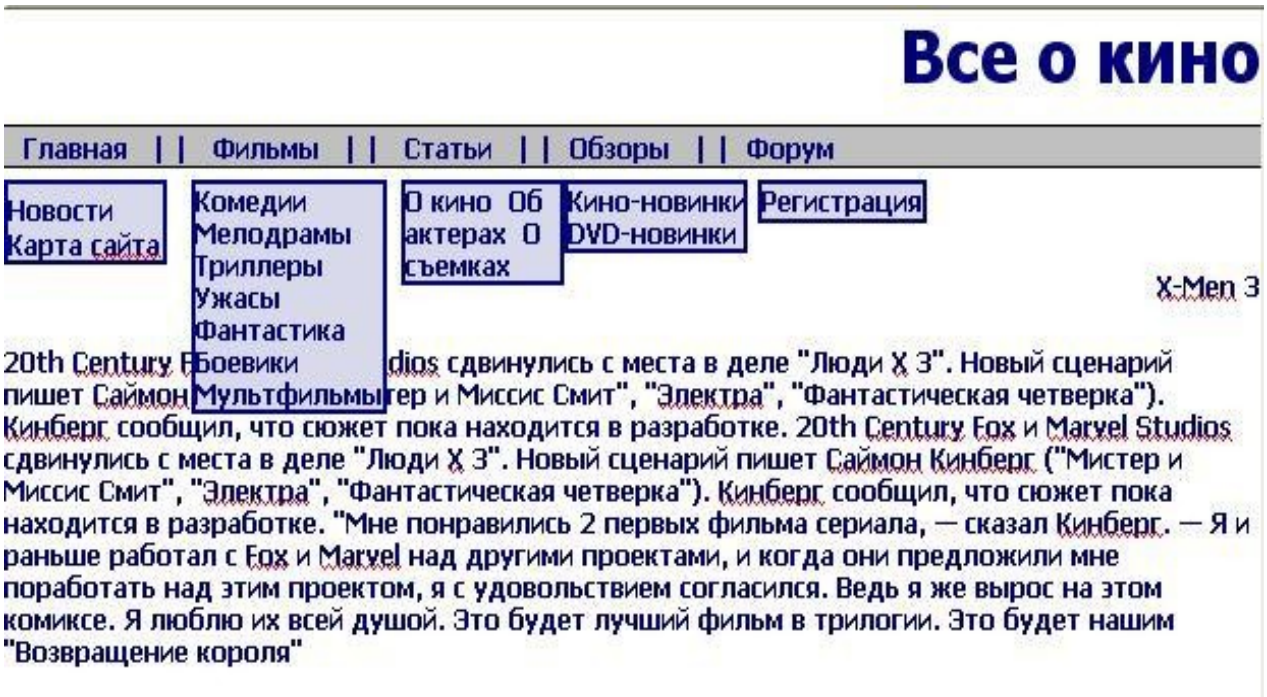

Рисунок 77 – Исходное меню

А теперь рассмотрим пример работы данной HTML-страницы. При этом, во время работы в один момент времени доступен только один пункт меню.

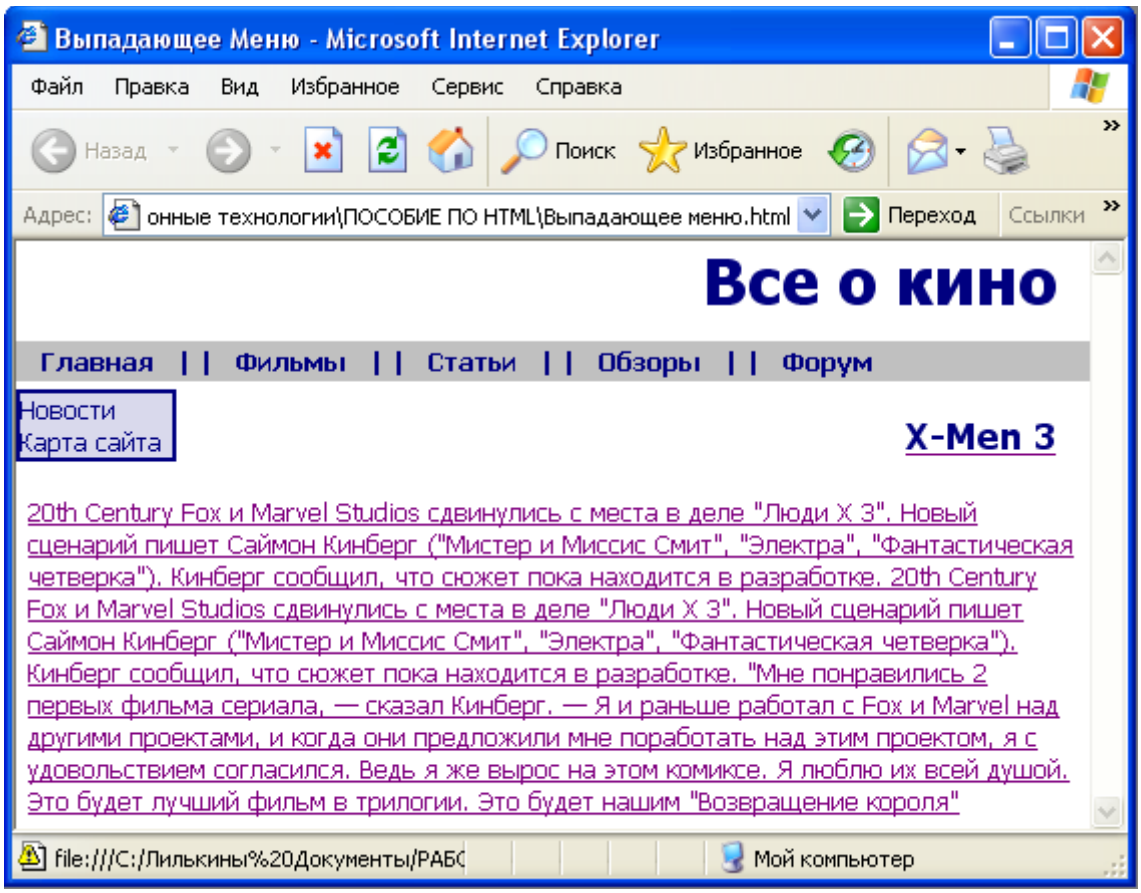

Рисунок 78– Активизация пункта меню “Главная”

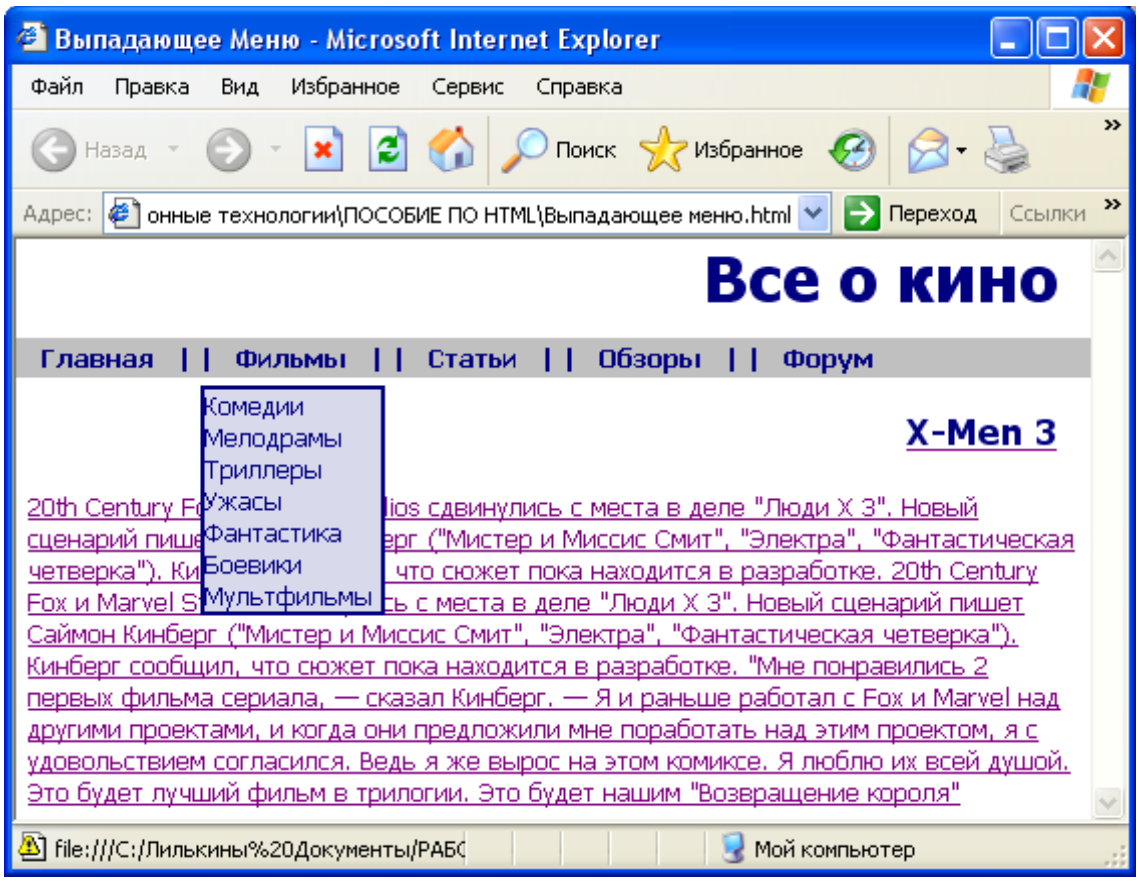

Рисунок 79 – Активизация пункта меню “Фильмы”

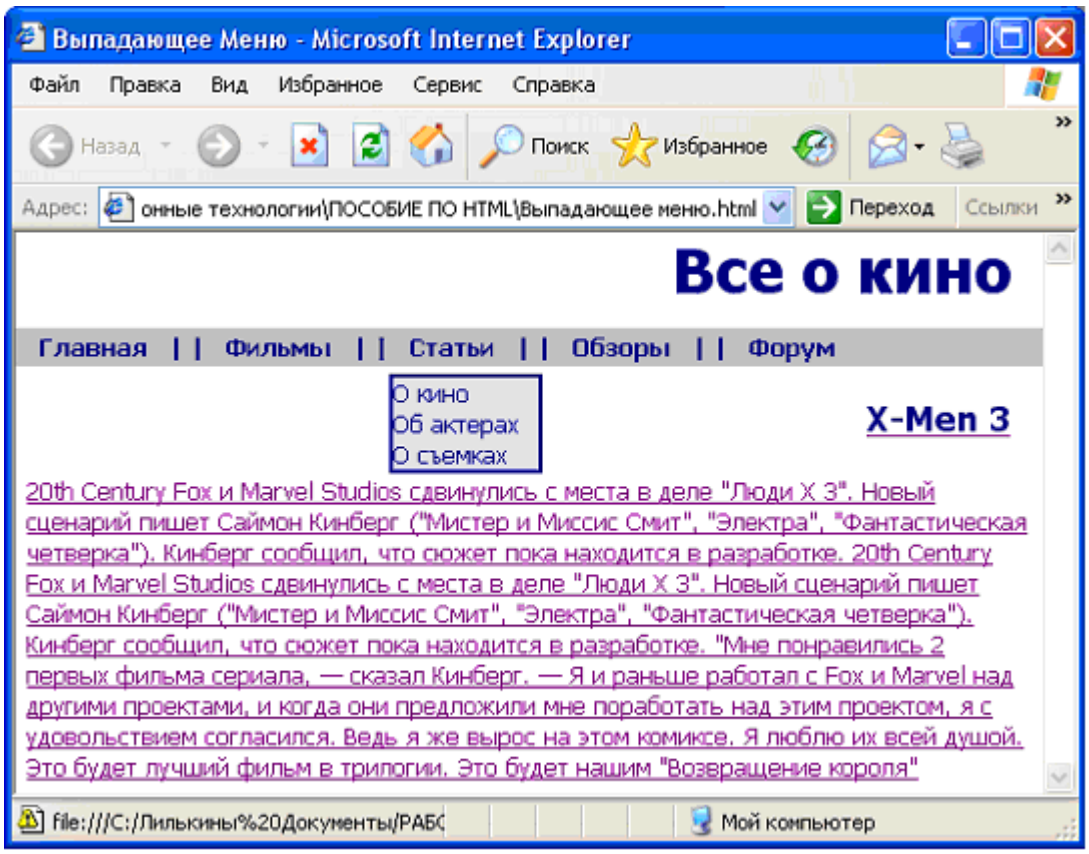

Рисунок 80 – Активизация пункта меню “Статьи”

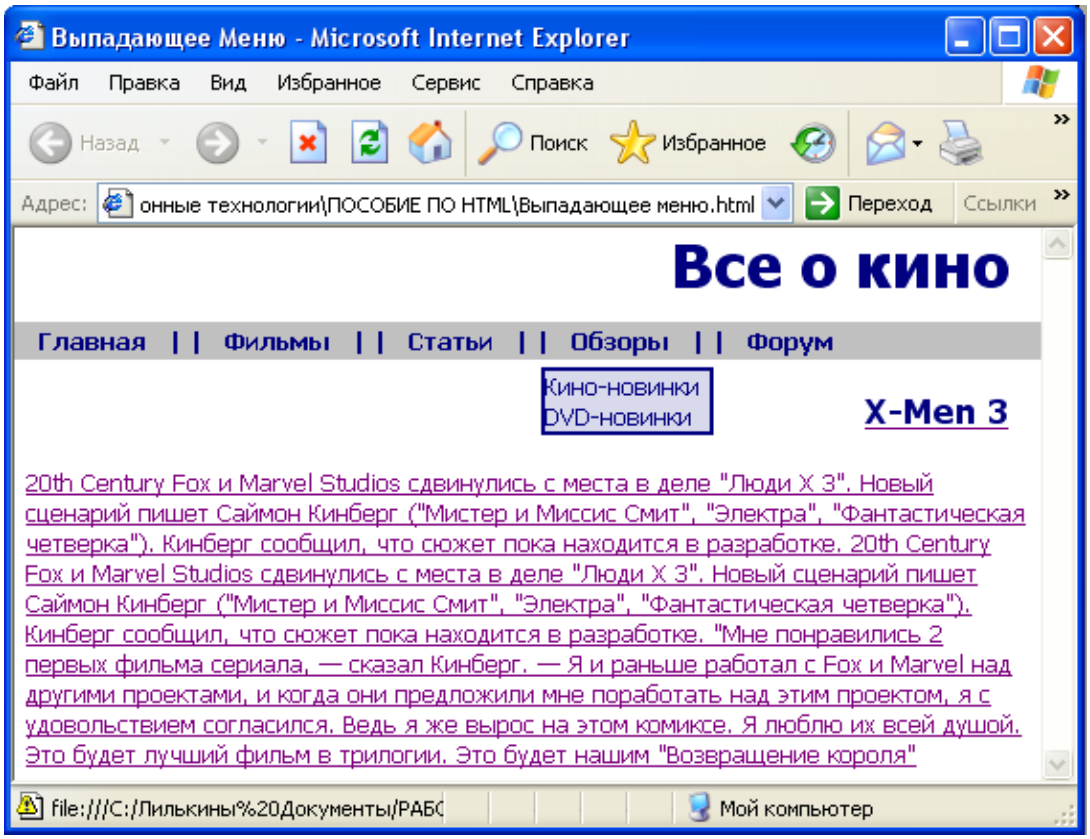

Рисунок 81 – Активизация пункта меню “Обзоры”

Таким образом, CSS дает немалые возможности создания динамических Web-сайтов для последующего отображения в сети Internet.

Заключение

В данном учебном пособии рассматриваются основы создания динамических Web-сайтов с помощью наиболее популярного на сегодняшний день языка CSS – Cascading Style Sheets.

Материал в пособии изложен с нарастанием сложности. Сначала студенты изучают особенности языка CSS, его синтаксис. В частности работают с такими элементами как селекторы: селекторы типа, селекторы класса, контекстные и ID-селекторы. После учатся интегрировать в HTML-код страницы различные виды каскадных таблиц стилей, такие как внедренные, внутренние и внешние. Затем студенты учатся присваивать различным свойствам в CSS значения. Рассматриваются свойства шрифта, цвета и фона, текста, списков, расположение элементов, свойства границ и блоков.

После этого студенты работают с блоковыми элементами страницы, учатся их позиционировать, используя абсолютное, относительное позиционирование, фиксированные блоки и стандартный поток. Рассматриваются следующие свойства стандартного потока: float, clear, display, visibility, overflow.

Затем студенты остановятся на подробном изучении фильтров. Сначала изучаются статические фильтры и их основные особенности. В частности, фильтр Alpha задает прозрачность, Blur задает размытость, Chroma делает прозрачным отдельный цвет элемента страницы, DropShadow отбрасывает тень у элемента страницы, FlipH и FlipV создают зеркальное отражение объекта относительно горизональной и вертикальной оси соотвественно, Glow создает эффект “тления” элемента страницы, Invert инвертирует пиксели объекта, Gray выводит объект в градациях серого цвета, удаляя всю информацию обо всех остальных цветах, Light добавляет источник света, действующий на объект, Mask отображает элемент страницы как выделенный, Shadow создает эффект отрасывания тени, Xray отображает содержимое как рентгеновский снимок, Wave создает волнистое искажение, BasicImage устанавливает сразу несколько фильтров для содержимого. Затем изучаются динамические фильтры revealtrans и blendtrans. При этом, для их обработки используются элементы языка JavaScript.

И, наконец, студенты учатся совмещать элементы CSS с обработчиками языка JavaScript, т.е. создавать динамические эффекты Web-страницы. К ним, в частности, относятся изменение фона в таблице, изменение стиля текста, динамические гиперссылки, изменение цвета полосы прокрутки, стиля курсора, создание ссылок-кнопок, подсветка кнопок и текста, выпадающее меню и многое другое.

Таким образом, предлагаемое учебное пособие поможет студентам научиться разрабатывать динамические Web-сайты, состоящие из большого числа Web-страниц, оформленных единым стилем. Помимо этого студенты научатся обрабатывать элементы каскадных таблиц стилей с помощью языка управления сценариями JavaScript.

Список использованных источников

1 **Основы Web-технологий** / П.Б. Храмцов, С.А. Брик, А.М. Русак, А.И. Сурин; под редакцией П.Б. Храмцова. – М.: ИНТУИТ.РУ “Интернет-Университет Информационных Технологий”, 2003. – 512 с. – ISBN 5-9556-0001-9.

2 **Матросов, А.В., Сергеев, А.О., Чаунин, М.П.** HTML 4.0. – СПб.: БХВ, 2001. – 672 с. – ISBN 5-8206-0072-X.

3 **Введение в HTML** [Электронный ресурс]: учебник для вузов / П.Б. Храмцов, С.А. Брик, А.М. Русак, А.И. Сурин; под редакцией П.Б. Храмцова, 2003. - Режим доступа: WWW.URL: http://www.intuit.ru.

4 **Коротеева, Е.** Введение в HTML [Электронный ресурс], 1995. - Режим доступа: WWW.URL:http://books.kulichki.net/index.php.

5 **Пауэлл, Т.А.** Полное руководство по HTML / Пер. с англ. А.В. Качанов. – Мн.: ООО “Попурри”, 2002. – 912 с. – ISBN 5-9556-0001-9.

6 **Мальчук, Е.В.** HTML и CSS. Самоучитель. – М.: Издательский дом “Вильямс”, 2006. – 416 с. - ISBN 5-8459-0907-4.

7 **Лукач. Ю. С.** Справочник Веб-разработчика [Электронный ресурс], 2002. - Режим доступа: WWW.URL:http://wdh.suncloud.ru/

8 **Применение каскадных таблиц стилей (CSS)** [Электронный ресурс]: учебник для вузов / П.Б. Храмцов, С.А. Брик, А.М. Русак, А.И. Сурин; под редакцией П.Б. Храмцова, 2003. - Режим доступа: WWW.URL: http://www.intuit.ru.

9 **Щербаков С. В.** Основы CSS [Электронный ресурс], 2000. - Режим доступа: WWW.URL:http://books.kulichki.net/index.php.

10 **Штайнер, Г.** HTML/XML/CSS: справочник/ Г. Штайнер. – М.: Лаборатория Базовых знаний, 2001. – 512 с.: ил. – ISBN5-93208-091-4.

11 **Ливенгстон, Д.** Браун, М. CSS и DHTML: Web-профессионалам / Д. Ливенгстон, М. Браун. - BHV-Киев, 2002.

12 **Коржинский, С.Н.** Настольная книга Web-мастера: эффективное применение HTML, CSS и JavaScript. – М.: КноРус, 2000.

13 **Гаевский, А.Ю.** 100% самоучитель по созданию Web-страниц и Web-сайтов. HTML и JavaScript. – М.: Технолоджи-3000, 2005. – 464 с. - ISBN 5-94472-027-1.

14 **Монахов, В.В.** Языки программирования Internet. Основы языка JavaScript [Электронный ресурс]. – СПБ, 2002. - Режим доступа: WWW.URL: http://barsic.spbu.ru/www/edu/js.html.

15 **Дунаев, В.** Самоучитель JavaScript, 2-е изд. – СПб.: Питер, 2005. – 395 с. – ISBN 5-469-00-804-5.

16 **Николенко, Д.В.** Практические занаятия по Javascript. – СПб.: “Наука и техника”, 2000. - 129 с. – ISBN 5-7931-0126-8.

17 **Храмцов, П. Б.** Практическое введение в программирование на JavaScript [Электронный ресурс], 2003. - Режим доступа: WWW.URL: http://books.kulichki.net/data/javascript/js1.

Приложение А
(*справочное*)
Контрольные вопросы для закрепления материала

1 Каскадные таблицы стилей (CSS). Основное назначение и преимущества использования.
2 Синтаксис таблиц стилей. Особенности использования селекторов типа, селекторов класса, ID-селекторов, контекстных селекторов.
3 Способы внедрения каскадных таблиц стилей в HTML-документ. Внутренние таблицы, внешние, внедренные таблицы стилей.
4 Каскадирование стилей в CSS.
5 Особенности наследования стилей элементами Web-страницы.
6 Свойства CSS. Свойства шрифта, цвета и фона, текста, списков. Расположение элементов, свойства границ, блоков.
7 Единицы измерения, применяемые при присвоении значений в CSS.
8 Особенности использования псевдокалссов в CSS.
9 **Форматирование блоковых элементов Web-страницы средствами CSS.**
10 **Использование параметров margin, padding и border при работе с блоковыми элементами.**
11 **Использование слоев в CSS.**
12 Абсолютное позиционирование элементов. Работа с блоками.
13 Абсолютное позиционирование элементов. Работа с текстом.
14 Применение свойства z-index для управления порядком наслоения абсолютно позиционированных элементов друг на друга.
15 Относительное позиционирование средствами CSS. Работа с блоками.
16 Относительное позиционирование средствами CSS. Работа с текстом.
17 Фиксированные блоки в CSS.
18 Плавающие блоки в CSS.
19 Статическое позиционирование.
20 Особенности использования ствойств float, clear, display, visibility, overflow при статическом позиционировании элементов в CSS.
21 Фильтры в CSS. Отличия статических и динамических фильтров.
22 Статические фильтры в CSS. Особенности использования фильтров Alpha, Blur, Chroma, DropShadow, FlipH, FlipV, Glow.
23 Статические фильтры в CSS. Особенности использования фильтров Invert, Gray, Light, Mask, Shadow, Xray, Wave.
24 Использование нескольких фильтров в CSS, фильтр BasicImage.
25 Динамические фильтры в CSS. Особености использования фильтров Blendrans и Reveatrans.
26 Создание динамических эффектов средствами CSS.

27 Изменение цвета полосы прокрутки средствами CSS.
28 Изменение стиля курсора средствами CSS.
29 Создание динамического выпадающего меню средствами CSS.
30 Особенности применения элементов языка JavaScript при работе с каскадными таблицами стилей.

Приложение Б
(*справочное*)
Тесты

1) Как принято переводить сокращение CSS?
а) Стилевые каскады полотен;
б) Каскадные стилевые слои;
в) Каскадные таблицы стилей.
г) любой вариант возможен;
д) нет верного ответа.

2) Какие типы каскадных таблиц не существуют:
а) внедренные;
б) внутренние;
в) альтернативные;
г) внешние;
д) нет верного ответа.

3) Как называются таблицы стилей, описанные в секции заголовка документа?
а) внедренные;
б) внутренние;
в) альтернативные;
г) внешние;
д) нет верного ответа.

4) Как называются таблицы стилей, подключенные с помощью inline-описания?
а) внедренные;
б) внутренние;
в) альтернативные;
г) внешние;
д) нет верного ответа.

5) Какого типа селекторов не существует в CSS?
а) селектор типа;
б) селектор класса;
в) ID-селектор;
г) DI-селектор;

д) контекстные селекторы.

6) К какому типу селекторов можно отнести следующий пример: H1 {color:red; size:20pt;}?

а) селектор типа;
б) селектор класса;
в) ID-селектор;
г) DI-селектор;
д) контекстные селекторы.

7) К какому типу селекторов можно отнести следующий пример: H1.blue {color:blue; size:20pt;}?

а) селектор типа;
б) селектор класса;
в) ID-селектор;
г) DI-селектор;
д) контекстные селекторы.

8) К какому типу селекторов можно отнести следующий пример: .warning {font-family:Arial Narrow, sans-serif; font-size: 10pt; color:red}?

а) селектор типа;
б) селектор класса;
в) ID-селектор;
г) DI-селектор;
д) контекстные селекторы.

9) К какому типу селекторов можно отнести следующий пример: #tulip {color: yellow;font-size:20pt;}?

а) селектор типа;
б) селектор класса;
в) ID-селектор;
г) DI-селектор;
д) контекстные селекторы.

10) К какому типу селекторов можно отнести следующий пример: P EM {color:silver;}?

а) селектор типа;
б) селектор класса;
в) ID-селектор;
г) DI-селектор;
д) контекстные селекторы.

11) Какая из таблиц стилей используется в заданном примере: <H1 style=“border-width: 1; border: solid; text-align: center”>?

а) внедренная;

б) внутренняя;
в) альтернативная;
г) внешняя;
д) нет верного ответа.

12) Какая из таблиц стилей используется в заданном примере: <Body style=“font-family: 'Times New Roman', serif; font-size: 12pt; color: darkgreen;”>?
а) внедренная;
б) внутренняя;
в) альтернативная;
г) внешняя;
д) нет верного ответа.

13) Какая из таблиц стилей используется в заданном примере: <HTML><TITLE>Пример </TITLE>
<HEAD>
<STYLE type=“text/css”>
H1 {border-width: 1; border: solid; text-align: center}
</STYLE>
</HEAD>
<BODY>
<H1>Пример использования стилей</H1>
</BODY>
</HTML>?
а) внедренная;
б) внутренняя;
в) альтернативная;
г) внешняя;
д) нет верного ответа.

14) Подключение какой таблицы стилей используется в заданном примере: <LINK rel= “stylesheet” href=“style.css” type=“text/css”>?
а) внедренной;
б) внутренней;
в) альтернативной;
г) внешней;
д) нет верного ответа.

15) Подключение какой таблицы стилей используется в заданном примере: <LINK rel=“alternate stylesheet” title=“big print” href=“bigprint.css” type=“text/css”>?
а) внедренной;
б) внутренней;
в) альтернативной;

г) внешней;
д) нет верного ответа.

16) Какая из таблиц стилей имеет наибольший приоритет?
а) внедренная;
б) внутренняя;
в) альтернативная;
г) внешняя;
д) нет верного ответа.

17) Какая из таблиц стилей имеет наименьший приоритет?
а) внедренная;
б) внутренняя;
в) альтернативная;
г) внешняя;
д) нет верного ответа.

18) Какое свойство CSS задает начертание (гарнитуру) шрифта в CSS?
а) font-family;
б) font-size;
в) font-style;
г) font-variant;
д) font-heigth.

19) Какое свойство CSS задает размер шрифта?
а) font- weight;
б) font-size;
в) font-style;
г) font-variant;
д) font-heigth.

20) К какому свойству относятся параметры x-small, large?
а) font-variant;
б) font-family;
в) font-size;
г) font- weight;
д) font- heigth.

21) Используя какое свойство CSS, можно усилить насыщенность (“жирность”) шрифта?
а) font- weight;
б) font-size;
в) font-family;
г) font-heigth;
д) font-variant.

22) К какому свойству относятся параметры normal, italic?

а) font- weight;

б) font-size;

в) font-family;

г) font-style;

д) нет верного ответа.

23) Какое свойство преобразует указанным образом текст в элементе, например, прописные буквы, нижний регистр, верхний регистр?

а) text-align;

б) text-decoration;

в) text-indent;

г) text-transform;

д) word-spacing.

24) Какое свойство задает отступ первой строки в элементе?

а) text-align;

б) text-decoration;

в) text-indent;

г) text-transform;

д) word-spacing.

25) Какое свойство определяет виды украшения текста, например мерцание, надчеркивание, подчеркивание?

а) text-align;

б) text-decoration;

в) text-indent;

г) text-transform;

д) word-spacing.

26) За что отвечает свойство background-attachment?

а) устанавливает исходное положение фонового изображения элемента;

б) указывает, должно ли фоновое изображение фиксироваться или прокручиваться вместе с документом;

в) устанавливает цвет фона для элемента;

г) указывает фоновое изображение элемента;

д) определяет, каким образом фоновое изображение заполняет элемент.

27) Какое из перечисленных свойств в CSS определяет расстояние между символами в тексте?

а) Letter-spacing;

б) Word-spacing;

в) Line-height;

г) Word-mode;

д) Direction.

28) Какое из перечисленных свойств в CSS задает направление строк текста: горизонтальное или вертикальное?

а) Letter-spacing;
б) Word-spacing;
в) Line-height;
г) Word-mode;
д) Direction.

29) Какое из перечисленных свойств в CSS определяет дополнительное расстояние между словами в тексте?

а) Letter-spacing;
б) Word-spacing;
в) Line-height;
г) Word-mode;
д) Direction.

30) С помощью какого свойства в CSS задается цвет стрелок полосы прокрутки?

а) scrollbar-arrow-color;
б) scrollbar-base-color;
в) scrollbar-dark-shadow-color;
г) scrollbar-face-color;
д) scrollbar-hightlight-color.

31) С помощью какого свойства в CSS задается основной цвет стрелок и бегунка (по умолчанию серый)?

а) scrollbar-arrow-color;
б) scrollbar-base-color;
в) scrollbar-dark-shadow-color;
г) scrollbar-face-color;
д) scrollbar-hightlight-color.

32) С помощью какого свойства в CSS задается цвет блика на полосе прокрутки?

а) scrollbar-arrow-color;
б) scrollbar-base-color;
в) scrollbar-dark-shadow-color;
г) scrollbar-face-color;
д) scrollbar-hightlight-color.

33) Какое из свойств в CSS определяет форму курсора мыши в виде песочных часов, которую он принимает при наведении на элемент страницы?

а) cursor: wait;

б) cursor: hand;
в) cursor: move;
г) cursor: crosshair;
д) cursor: default.

34) Какое из свойств определяет форму курсора мыши в виде обычной стрелки (курсор по умолчанию)?
а) cursor: wait;
б) cursor: hand;
в) cursor: move;
г) cursor: crosshair;
д) cursor: default.

35) Какие единицы измерения не относятся к абсолютным?
а) дюймы (in);
б) сантиметры (cm);
в) миллиметры (mm);
г) точки (pt),
д) пиксели (px).

36) Какое свойство CSS используется для задания полей, неких свободных, никем не занятых пространств вокруг границы блока.
а) padding;
б) margin;
в) border;
г) frame;
д) нет верного ответа.

37) Какое свойство определяет нижнее поле элемента?
а) margin-top;
б) margin-right;
в) margin-left;
г) margin-bottom;
д) margin -type.

38) Какое свойство CSS используется для задания отступов, чтобы отделить основное содержимое блока от границы таким образом, чтобы граница располагалась на некотором расстоянии от содержимого?
а) padding;
б) margin;
в) border;
г) frame;
д) нет верного ответа.

39) Какое свойство указывает величину отступа от элемента сверху?

а) padding -top;
б) padding -right;
в padding -left;
г) padding -bottom;
д) padding -type.

40) Какое свойство границ определяют толщину границ блоков для верхнего сегмента блока?
а) border-top-width;
б) border-right-width;
в) border-bottom-width;
г) border-left-width;
д) border-middle-width.

41) Какое свойство границ задает цвет правого сегмента границы
а) border-top-color;
б) border-right-color;
в) border-bottom-color;
г) border-left-color;
д) border-middle-color.

42) Какое свойство границ задает тип линии, которой будет отображаться нижняя граница блока?
а) border-top-style;
б) border-right-style;
в) border-bottom-style;
г) border-left-style;
д) border-middle-style.

43) Какое значение свойства границы border-style задает границу таким образом, что граница выглядит, как вдавленная в страницу линия, напоминающая желобок?
а) solid;
б) inset;
в) outset;
г) ridge;
д) groove.

44) Какое значение свойства границы border-style задает границу таким образом, что граница выглядит, как рельефная линия, выступающая над поверхностью страницы?
а) solid;
б) inset;
в) outset;
г) ridge;

д) groove.

45) Какое значение свойства границы border-style задает границу таким образом, что весь блок выглядит как бы выступающим над поверхностью страницы?

а) solid;
б) inset;
в) outset;
г) ridge;
д) groove.

46) Какое значение свойства границы border-style задает границу таким образом, что создается впечатление “вдавленности” всего блока в поверхность страницы?

а) solid;
б) inset;
в) outset;
г) ridge;
д) groove.

47) Какое значение свойства границы border-style задает границу таким образом, граница отображается в виде сплошной линии, толщина которой равна значению border-width?

а) solid;
б) inset;
в) outset;
г) ridge;
д) groove.

48) Какой тип позиционирования подразумевает назначение точных координат расположения элемента?

а) абсолютное;
б) относительное;
в) фиксированное;
г) статическое;
д) нет верного ответа.

49) Какой тип позиционирования позволяет разместить блоки на странице в координатах охватывающего их блока?

а) абсолютное;
б) относительное;
в) фиксированное;
г) статическое;
д) нет верного ответа.

50) Какой тип позиционирования позволяет зафиксировать блок на странице, он всегда будет находиться в области просмотра страницы, т.е. не будет прокручиваться?

а) абсолютное;
б) относительное;
в) фиксированное;
г) статическое;
д) нет верного ответа.

51) При использовании какого типа позиционирования позиционированные объекты могут накладываться друг на друга?

а) абсолютное;
б) относительное;
в) фиксированное;
г) статическое;
д) нет верного ответа.

52) Среди параметров стиля, какое специальное свойство позиционирования используется для указания порядка, в котором элементы будут перекрывать друг друга, т.е. позволяет управлять порядком наслоения абсолютно позиционированных элементов друг на друга?

а) left;
б) top;
в) x;
г) y;
д) z-index.

53) Среди параметров стиля, какое специальное свойство позиционирования используется для задания расстояния в пикселах от верхнего края окна (y-координата)?

а) left;
б) top;
в) x;
г) y;
д) z-index.

54) При использовании свойства z-index тот блок, у которого значение свойства z-index больше будет располагаться

а) поверх всех остальных блоков;
б) под всеми остальными блока;
в) справа от всех остальных блоков;
г) слева от всех остальных блоков;
д) внизу от всех остальных блоков.

55) Какое свойство статических блоков определяет, будет ли блок плавающим, и в какую сторону он будет перемещаться?

а) float;
б) clear;
в) display;
г) visibility;
д) overflow.

56) Какое свойство статических блоков определяет, какие стороны перемещаемых блоков не могут соседствовать с другими перемещаемыми блоками?

а) float;
б) clear;
в) display;
г) visibility;
д) overflow.

57) Какое значение свойства display отключает отображение блока в окне броузера так, как будто его никогда не было в коде страницы?

а) none;
б) inline;
в) block;
г) list-item;
д) среди представленных вариантов ответов нет верного.

58) С помощью какого значения свойства display элемент становится частью списка?

а) none;
б) inline;
в) block;
г) list-item;
д) среди представленных вариантов ответов нет верного.

59) Какое значение свойства display определяет строчный блок?

а) none;
б) inline;
в) block;
г) list-item;
д) среди представленных вариантов ответов нет верного.

60) Какое значение свойства display определяет некий элемент как структурный блок?

а) none;
б) inline;
в) block;

г) list-item;
д) среди представленных вариантов ответов нет верного.

61) Какое свойство статических блоков задает тип конкретного блока?
а) float;
б) clear;
в) display;
г) visibility;
д) overflow.

62) Какое свойство статических блоков управляет видимостью блока?
а) float;
б) clear;
в) display;
г) visibility;
д) overflow.

63) Какое значение свойства overflow добавляет полосы прокрутки к блоку отображения?
а) scroll;
б) visible;
в) hidden;
г) auto;
д) каждое из значений обладает такой возможностью.

64) При использовании какого значения свойства overflow, содержимое, превышающее размеры блока увеличивает размеры блока отображения?
а) scroll;
б) visible;
в) hidden;
г) auto;
д) нет верного ответа.

65) Какое значение свойства overflow предписывает броузеру обрезать содержимое, которое превышает размеры блока?
а) scroll;
б) visible;
в) hidden;
г) auto;
д) каждое из значений обладает такой возможностью.

66) Какое значение свойства overflow добавляет полосы прокрутки к блоку отображения в случае, если размеры содержимого элемента превосходят размеры блока отображения?

а) scroll;
б) visible;
в) hidden;
г) auto;
д) нет верного ответа.

67) Какое свойство статических блоков определяет поведение элемента, когда размеры его содержимого не соответствуют размерам блока отображения?
а) float;
б) clear;
в) display;
г) visibility;
д) overflow.

68) Какой из перечисленных фильтров в CSS задает уровень прозрачности элемента страницы?
а) Mask;
б) Light;
в) Glow;
г) Alpha;
д) FlipV.

69) Какой из перечисленных фильтров создает эффект движения объекта, размытие?
а) Blur;
б) Fliph;
в) Alpha;
г) Wave;
д) Xray.

70) Какой статический фильтр в CSS делает прозрачным отдельный цвет элемента Web-страницы?
а) Chroma;
б) Xray;
в) Light;
г) Invert;
д) Glow.

71) Какой статический фильтр в CSS отбрасывает тень у элемента страницы, которая отображается отдельно от самого элемента?
а) DropShadow;
б) Xray;
в) Chroma;
г) Blur;

д) Shadow.

72) Какой из перечисленных фильтров отражает объект симметрично относительно центральной горизонтальной оси?

а) Blur;
б) Fliph;
в) Xray;
г) Wave;
д) Alpha.

73) Какой статический фильтр в CSS создает зеркальное отражение объекта относительно вертикальной оси?

а) Fliph;
б) Light;
в) Alpha;
г) Flipv;
д) Wave.

74) Какой статический фильтр в CSS создает эффект “сияния” элемента страницы?

а) Glow;
б) Invert;
в) Flipv;
г) Light;
д) Fliph.

75) Какой из перечисленных статических фильтров инвертирует пиксели объекта?

а) Invert;
б) Glow;
в) BasicImage.
г) Chroma;
д) Wave.

76) Какой из перечисленных статических фильтров выводит объект в градациях серого, удаляя всю информацию о цветах?

а) Chroma;
б) Alpha;
в) Glow;
г) Mask;
д) Gray.

77) При использовании фильтра Light какой метод добавляет рассеянный свет к фильтру?

а) AddAmbient;
б) AddCone;

в) AddPoint;
г) ChangeColor;
д) MoveLight.

78) При использовании фильтра Light какой метод добавляет направленный источник света (прожектор)?
а) AddAmbient;
б) AddCone;
в) AddPoint;
г) ChangeColor;
д) MoveLight.

79) При использовании фильтра Light какой метод добавляет рассеянный источник света?
а) AddAmbient;
б) AddCone;
в) AddPoint;
г) ChangeColor;
д) MoveLight.

80) При использовании фильтра Light какой метод изменяет цвет источника света?
а) AddAmbient;
б) AddCone;
в) AddPoint;
г) ChangeColor;
д) MoveLight.

81) При использовании фильтра Light какой метод перемещает источник света новую точку с заданными координатами?
а) AddAmbient;
б) AddCone;
в) AddPoint;
г) ChangeColor;
д) MoveLight.

82) Какой статический фильтр в CSS создает эффект освещенности элемента Web-страницы, добавляет источник света, действующий на объект?
а) Mask;
б) Light;
в) Glow;
г) DropShadow;
д) FlipV.

83) Какой статический фильтр в CSS отображает прозрачный цвет элемента Web-страницы заданным цветом?

а) Glow;
б) Invert;
в) Gray;
г) Light;
д) Mask.

84) Какой статический фильтр в CSS создает эффект отбрасывания тени у элемента страницы?
а) Alpha;
б) Blur;
в) Chroma;
г) DropShadow;
д) Shadow.

85) Какой из перечисленных фильтров изменяет глубину цвета объекта и отрисовывает черно-белым, делая его похожим на рентгеновский снимок?
а) Alpha;
б) Blur;
в) Fliph;
г) Xray;
д) Invert.

86) Какой статический фильтр в CSS выполняет синусоидальное преобразование объекта вдоль вертикальной оси, создает волнистое искажение элемента Web-страницы?
а) FlipH;
б) Light;
в) Mask;
г) Wave;
д) BasicImage.

87) При использовании фильтра Basicimage какой параметр выводит содержимое Web-страницы в серых тонах?
а) GrayScale;
б) Invert;
в) Mask;
г) Mirror;
д) Rotation.

88) При использовании фильтра Basicimage какой параметр инвертирует все цвета содержимого Web-страницы?
а) GrayScale;
б) Invert;
в) Mask;
г) Mirror;

д) Rotation.

89) При использовании фильтра Basicimage какой параметр выводит изображение в виде маски, при этом непрозрачные точки становятся прозрачными, прозрачные выводятся заданным цветом?

а) GrayScale;
б) Invert;
в) Mask;
г) Mirror;
д) Rotation.

90) При использовании фильтра Basicimage какой параметр выводит зеркальное изображение содержимого Web-страницы?

а) GrayScale;
б) Invert;
в) Mask;
г) Mirror;
д) Rotation.

91) При использовании фильтра Basicimage какой параметр поворачивает содержимое Web-страницы на заданный угол?

а) GrayScale;
б) Invert;
в) Mask;
г) Mirror;
д) Rotation.

92) Какой из статических фильтров устанавливает действие сразу несколько фильтров для содержимого Web-страницы?

а) Mask;
б) Shadow;
в) Xray;
г) Wave;
д) BasicImage.

93) Какой из перечисленных фильтров не относится к статическим?

а) Blur;
б) Invert;
в) Xray;
г) Wave;
д) Revealtrans.

94) С помощью какого фильтра задается эффект появления и исчезновения блоков?

а) Xray;
б) Revealtrans;

в) Basicimage;
г) Mask;
д) Blendtrans.

95) Какой фильтр создает эффект постепенного повышения яркости и насыщенности изображения?
а) Blendtrans;
б) Revealtrans;
в) Basicimage;
г) Alpha;
д) Shadow.

96) При использовании псевдоклассов в CSS, каким образом задается ссылка, над которой находится курсор мышки?
а) a:link;
б) a:visited;
в) a:active;
г) a:hover;
д) a: alink.

97) Можно ли использовать совместно с CSS элементы языка JavaScript для обработки различных событий?
а) нет, это совершенно два разных языка;
б) можно только для создания динамических эффектов;
в) можно, только для обработки элементов форм;
г) можно только с версией CSS1;
д) можно всегда.

98) Можно ли в CSS для задания стиля курсора использовать графические файлы?
а) можно;
б) нельзя;
в) можно, но только в броузере Internet Explorer, версии 5,5 и выше?
г) можно, только файлы формата JPEG;
д) можно, только с использованием обработчиков языка JavaScript.

99) При использовании фильтра Revealtrans, какой атрибут задает способ преобразования объекта?
а) Duration;
б) Transition;
в) Apply;
г) Play;
д) Stop.

100) С помощью какого метода в динамически фильтрах фиксируется изображение как исходный кадр в цепочке кадров преобразования?
а) Duration;

б) Transition;
в) Apply;
г) Play;
д) Stop.