

Ульяновский государственный технический университет

С.М. Наместников

**ОСНОВЫ ЯЗЫКА ГИПЕРТЕКСТОВОЙ
РАЗМЕТКИ HTML И CSS**

Учебное пособие

Ульяновск 2014

Основы языка гипертекстовой разметки HTML и CSS: Учебное пособие/Сост. С. М. Наместников. – Ульяновск: УлГТУ, 2014. – с.

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ	5
ВВЕДЕНИЕ.....	6
ГЛАВА 1. СТРУКТУРА И ОСНОВНЫЕ ТЕГИ HTML	7
1.1. Понятие тега	7
1.2. Структура документа	8
1.3. Основные теги форматирования текста	10
1.3.1. Тег разделения на абзацы <p>	10
1.3.2. Тег переноса строки 	12
1.3.3. Тег отмены переноса строки <nobr> и мягкого переноса <wbr>.....	13
1.3.4. Заголовки внутри HTML-документа.....	14
1.3.5. Тег <hr> - горизонтальная линия.....	16
1.3.6. Тег <pre> - вывод предварительно отформатированного текста.....	17
1.3.7. Теги логического уровня форматирования текста	18
1.3.8. Теги физического уровня форматирования текста.....	20
1.3.9. Специальные символы	26
1.3.10. Ссылки в HTML-документе.....	27
ГЛАВА 2. СПИСКИ	32
2.1. Маркированный список.....	32
2.2. Нумерованный список.....	35
2.3. Список определений	36
ГЛАВА 3. ТАБЛИЦЫ	38
3.1. Создание простейших таблиц в HTML-документах	38
3.2. Параметры тега <table>	42
3.3. Создание более сложных таблиц в HTML-документах	46
3.4. Форматирование данных в ячейках таблиц	47
ГЛАВА 4. КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ	50
4.1. Встраивание CSS в HTML-документ.....	50
4.2. Единицы измерения в таблицах стилей.....	52
4.3. Селекторы-теги и формат записи свойств в CSS.....	53
4.4. Селектор CALSS	55
4.5. Селектор ID.....	59
4.6. Выбор первого дочернего элемента.....	60
4.7. Выбор сестринских элементов	61
4.8. Псевдоклассы	61
4.9. Селекторы псевдоэлементов.....	65
ГЛАВА 5. ПРИМЕРЫ РАЗМЕТОК СТРАНИЦ С ИСПОЛЬЗОВАНИЕМ CSS	67
5.1. Разметка страниц сайта http://selrecipes.ru	60
5.2. Разметка страниц сайта http://math.sernam.ru	72
5.3. Разметка страниц сайта http://reeders.ru	77
ГЛАВА 6. ФОРМЫ В HTML	82

6.1. GET и POST-запросы.....	82
6.2. Создание HTML-форм.....	83
6.3. Примеры HTML-форм.....	88

ПРЕДИСЛОВИЕ

Целью написания данного учебного пособия является компактное и ясное изложение основных принципов языка гипертекстовой разметки HTML и CSS. При отборе материала предпочтение отдавалось тем конструкциям, которые наиболее часто используются на практике. Поэтому приведенное здесь изложение HTML и CSS не претендует на полноту описания, но по мнению автора позволит читателю усвоить достаточный минимум для создания несложных HTML-документов.

Чтение курса «Информатика» придает автору уверенность, что данное пособие будет полезно студентам и аспирантам при изучении HTML и CSS. Материал, приведенный здесь, может быть использован также преподавателями вузов при подготовке и проведении занятий по соответствующим дисциплинам.

ВВЕДЕНИЕ

Язык гипертекстовой разметки HTML (HyperText Markup Language) служит для наглядного и хорошо структурированного представления информации в сети Интернет, который приобрел широкую популярность в середине 90-х годов. Программы, интерпретирующие данную разметку и выводящие информацию на экран, стали называться браузерами. На сегодняшний день известно много их разновидностей: Firefox, Chrome, Opera, Internet Explorer и др.

Разнообразие браузеров впоследствии привело к нюансам и различиям в интерпретации некоторых инструкций языка HTML и соответственно проблемам адаптации HTML-документа к разным вариантам их интерпретации. Для разрешения данной проблемы был создан консорциум W3C (World Wide Web Consortium), в задачу которого входит составление спецификации, отражающей современный уровень развития возможностей языка с учетом разнообразных предложений разработчиков браузеров. Важными итогами работы консорциума стал выход спецификации HTML 4.0 и HTML 5.0. В частности в HTML 4.0 рекомендуется разделение описания структуры документа от его вида на экране монитора или мобильного устройства, что значительно упрощает разработку, исправление документа, а также поддержку большого числа платформ, сред и т.п.

Современные браузеры имеют довольно широкие возможности для представления информации и взаимодействию пользователей в сети Интернет. Кроме форматированного вывода текста они позволяют отображать изображения, видео, проигрывать музыку. Встроенный в HTML-документ язык JavaScript позволяет анализировать и обрабатывать содержимое страницы, взаимодействовать с сервером (откуда загружен документ), динамически менять свойства и содержание документа в окне браузера. По сути, на сегодняшний момент, браузер стал еще одной платформой взаимодействия пользователя с информацией и другими пользователями, а для ряда мобильных устройств составной частью операционной системы.

СТРУКТУРА И ОСНОВНЫЕ ТЕГИ HTML

1.1. Понятие тега

Сам по себе HTML-документ представляет собой текстовый файл, содержащий набор инструкций для представления информации в требуемом виде в окне браузера. Инструкции записываются в виде тегов (читается тэг, от англ. tag) и имеют следующий синтаксис:

```
<имя_тега [возможные параметры]>
```

например

```
<html> - тег начала HTML-документа;  
<body> - тег начала содержимого HTML-документа;  
<body bgcolor="blue"> - тег с параметром bgcolor, задающий фон  
страницы.
```

Чтобы указать браузеру место окончания действия того или иного тега, используется закрывающий тег. Это тег, имеющий то же самое имя с прямым слешем '/' перед ним, например,

```
<html>  
HTML-документ  
</html>
```

здесь </html> - завершающий тег для <html>, или

```
<body>  
Содержание HTML-документа  
</body>
```

здесь </body> - завершающий тег для <body>, и т.д.

Следует отметить, что не каждый открывающий тег должен иметь соответствующий закрывающий. Имеется набор тегов, действие которых начинается и заканчивается одним открывающим тегом, например:

```
 - тег для вывода изображения image.gif в  
окно браузера;  
<input type="text" value=""> - тег для ввода значений в окне  
браузера.
```

Далее, при рассмотрении тегов, будут указываться способы и особенности их применения.

Из приведенных выше примеров записи тегов можно заметить, что существуют теги, содержащие в себе информацию (а также другие теги), например, `html` и `body`, которые в дальнейшем будем называть **тегами-контейнерами**. Соответственно информацию, содержащуюся в них – **содержимым** тега-контейнера.

Как и в любом языке программирования, язык HTML имеет теги для создания комментариев внутри документа, т.е. текст, который имеется в документе, но не выводится браузером на экран. Для создания комментариев используется следующая пара тегов:

```
<!-- Это комментарий -->
```

Имеется ряд тегов, для которых завершающие теги опускаются большинством авторов документов. Например, тег `<p>` (начало абзаца), как правило, не имеет в документе завершающего тега `</p>`. Его завершение определяется браузером по «ходу» документа, например, если встретится очередной открывающий тег `<p>`.

Существуют общие правила интерпретации тегов браузерами. В отличие от языков программирования, в которых ошибочные операторы приводят к соответствующим ошибкам, в HTML не принято реагировать на неверную запись тегов. Неверно записанный тег или его параметр должен просто игнорироваться браузером. Это общее правило для всех браузеров, под действие которых попадают не только ошибочные теги, но и теги, не распознанные данной версией браузера.

1.2. Структура документа

При создании HTML-документов рекомендуется соблюдать следующую структуру:

```
<html>

<head>
<!-- Раздел заголовка -->
</head>

<body>
<!-- Тело документа -->
</body>

</html>
```

Здесь `<html>` - это тег, обозначающий начало HTML-документа. Браузер, встретивший данный тег, будет «знать», что представленную ниже информацию следует интерпретировать как HTML-страницу. Тег `<head>` означает раздел заголовка, который обычно включает:

1. Заголовок страницы

```
<title>Заголовок страницы</title>
```

2. Кодировку, в которой представлен HTML-документ

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
```

3. Краткое описание содержания документа

```
<meta name="description" content="библиотека, содержащая ...">
```

4. Список ключевых слов документа

```
<meta name="keywords" content="библиотека, книги, статьи">
```

5. Подключаемые внешние javascript файлы (скрипты)

```
<script type="text/javascript" src="http://reeders.ru/js/jquery-1.9.0.min.js"></script>
```

6. Внешние таблицы стилей (css) для форматирования документа

```
<link href="http://reeders.ru/css/style.css" rel="stylesheet" type="text/css" />
```

Раздел тега `<head>` может содержать и другие конструкции языка HTML, например, базовый адрес сайта: `<base href="http://reeders.ru/" />` и многое другое. В конечном итоге разработчик сам решает, что должно быть записано в данном разделе. Он может быть и пустым, т.е. не содержать ничего. Тогда браузер будет руководствоваться своими настройками и алгоритмами при отображении такого документа. Наконец, раздел `<head>` может совсем отсутствовать и это будет аналогично пустому разделу. Однако при разработке HTML-страницы крайне не рекомендуется пропускать данный раздел и его «минимальная конфигурация» должна иметь следующий вид:

```
<head>
<title>Заголовок страницы</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<meta name="description" content="библиотека, содержащая ...">
<meta name="keywords" content="библиотека, книги, статьи">
</head>
```

В тегах `<body>...</body>` заключается отображаемая информация (тело) страницы: текст, изображения, таблицы, звук, видео, объекты (например, flash-проигрыватель), javascript-программы, таблицы стилей (css) и т.д.

Подробнее о том, что и как можно записывать в раздел `<body>`, т.е. как формировать HTML-документ будет рассматриваться в последующих параграфах данного пособия.

Тег `<body>` имеет следующий набор необязательных параметров:

Параметр	Пример	Описание
<code>alink="цвет"</code>	<code><body alink="red"></code>	Устанавливает цвет активных ссылок
<code>link="цвет"</code>	<code><body link="blue"></code>	Устанавливает цвет непосещенных ссылок
<code>vlink="цвет"</code>	<code><body vlink="green"></code>	Устанавливает цвет посещенных ссылок
<code>background="URL"</code>	<code><body background="a.gif"></code>	Устанавливает изображение фона документа
<code>bgcolor="цвет"</code>	<code><body bgcolor="white"></code>	Устанавливает цвет фона документа
<code>text="цвет"</code>	<code><body text="black"></code>	Устанавливает цвет текста в документе

Помимо перечисленных в таблице параметров существуют дополнительные, но они поддерживаются не всеми браузерами и поэтому опущены в данном пособии.

HTML-документ рекомендуется всегда завершать закрывающим тегом `</html>` и тем самым сигнализировать браузеру о завершении документа. Это полезная информация может быть использована, например, при обрыве связи во время загрузки страницы. Тогда отсутствие тега `</html>` может означать неполную загрузку страницы. На практике встречается и много других случаев, когда важно иметь корректную разметку страницы для исключения неверного отображения содержания документа в исключительных ситуациях.

1.3. Основные теги форматирования текста

Текст HTML-документа, как правило, представляет собой непосредственно текст и ссылки, ведущие либо на отдельные фрагменты того же документа, либо на другие документы. В данном параграфе рассмотрим способы представления данной информации в окне браузера.

1.3.1. Тег разделения на абзацы <p>

В качестве первого тега форматирования текста имеет смысл рассмотреть тег разделения на абзацы `<p>`. Предположим в текстовом редакторе подготовлено сообщение:

Это пример текста, который требуется отобразить в браузере.

Это новый абзац текста и должен быть выведен в браузере с новой строки.

Каждая строка этого текста написана в отдельной строке текстового редактора (например, NotePad++, MS Word и т.п.). Теперь составим простой HTML-документ для отображения этой информации:

```
<html>
<head>
<title>Пример 1</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
<meta name="description" content="пример">
<meta name="keywords" content="пример">
</head>

<body>
Это пример текста, который требуется отобразить в браузере.
Это новый абзац текста и должен быть выведен в браузере с новой
строки.
</body>
</html>
```

и сохраним его в текстовый файл ex1.htm. В результате в окне браузера будет высвечена следующая информация:

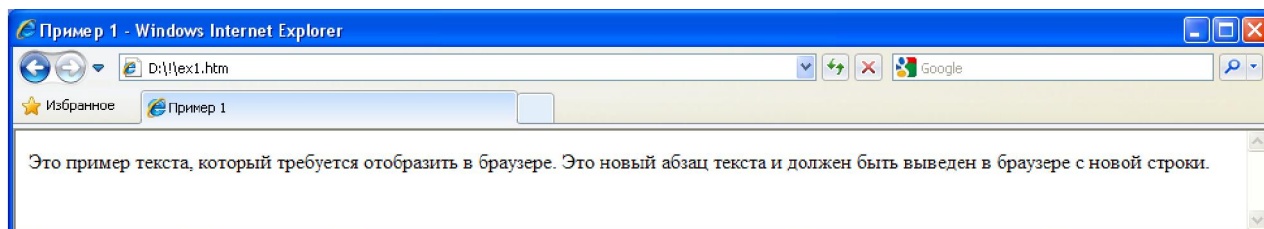


Рис. 1.1. Отображение файла ex1.htm в окне браузера

Из примера видно, что несмотря на разделение текста по строкам в текстовом редакторе, браузер выводит их в одну строку без деления на абзацы. Это общее правило вывода текста, т.к. браузер форматирует текст только на основании тегов, присутствующих в документе. Соответственно для отображения строк с нового абзаца достаточно поставить перед ними тег `<p>`:

```
<html>
<head>
<title>Пример 2</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
<meta name="description" content="пример">
<meta name="keywords" content="пример, абзац">
</head>

<body>
```

```
<p>Это пример текста, который требуется отобразить в браузере.  
<p>Это новый абзац текста и должен быть выведен в браузере с  
новой строки.  
</body>  
</html>
```

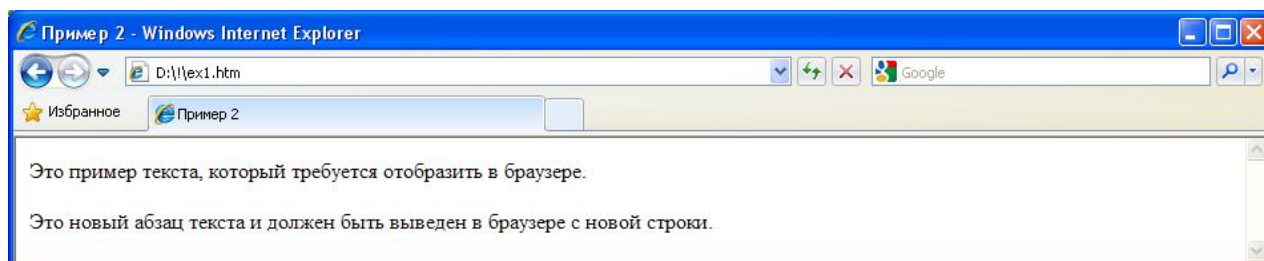


Рис. 1.2. Результат отображения примера 2

Тег `<p>` имеет необязательный параметр `align`, который может принимать значения:

- `left` – выравнивание по левому краю;
- `center` – выравнивание по центру;
- `right` – выравнивание по правому краю;
- `justify` – выравнивание по ширине страницы.

По умолчанию используется выравнивание по левому краю. Если требуется другой способ выравнивания текста, то нужно записать тег `<p>` с параметром `align` следующим образом:

```
<p align=center> или <p align="center">
```

кавычки в значении параметра можно опускать, если значение не содержит пробелы.

1.3.2. Тег переноса строки `
`

По умолчанию перенос строк в пределах абзаца выполняется браузером автоматически по символам-разделителям слов (обычно это пробел). Место переноса определяется размером окна вывода, т.е. если следующее слово не помещается в выделенную ширину, оно переносится на следующую строку. Вместе с тем иногда существует необходимость выполнить принудительный перенос в строго заданном месте текста. Это можно сделать с помощью тега переноса строки `
` как показано ниже в примере:

```
<html>  
<head>  
<title>Пример 3</title>  
<meta http-equiv="Content-Type" content="text/html;  
charset=windows-1251">  
<meta name="description" content="пример стихотворения">  
<meta name="keywords" content="пример, абзац, перенос">  
</head>
```

```
<body>
<p>Куда ты скачешь гордый конь?
<br>И где опустишь ты копыта?
<br>О мощный властелин судьбы
<br>Не так ли ты над самой бездной
<br>На высоте уздой железной
<br>Россию поднял на дыбы?!
<p>А.С. Пушкин
</body>
</html>
```

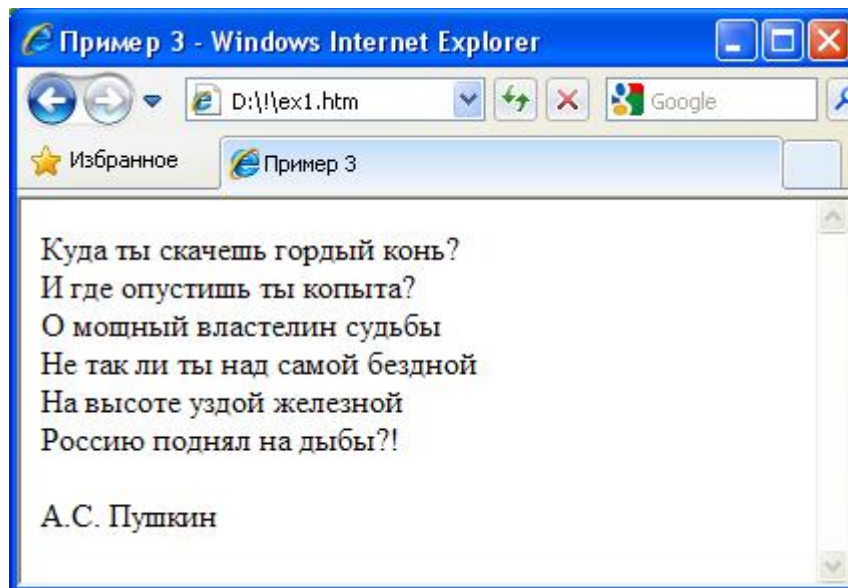


Рис. 1.3. Результат отображения примера 3

Из полученного результата рис. 1.3 видно, что тег `
` при переносе не образует пустую строку в отличие от тега `<p>`. В результате текст стихотворения отображается естественным образом, а фамилия автора благодаря тегу `<p>` отделена от него пустой строкой.

1.3.3. Тег отмены переноса строки `<nobr>` и мягкого переноса `<wbr>`

По умолчанию перенос строк в пределах абзаца выполняется браузером автоматически. Однако при необходимости можно указать браузеру выводить строку целиком без переносов. Если строка не будет помещаться на всю ширину, то появится горизонтальная полоса прокрутки для просмотра текста. Отмена переноса достигается с помощью тега `<nobr>` и соответствующего закрывающего тега `</nobr>` как показано ниже:

```
<html>
<head>
<title>Пример 4</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>
```

```

<body>
<nobr>
<p>Пример длинной строки текста, которая не будет переноситься
браузером. Вместо этого появится горизонтальная полоса прокрутки
для просмотра данной строки.
</nobr>
</body>
</html>

```

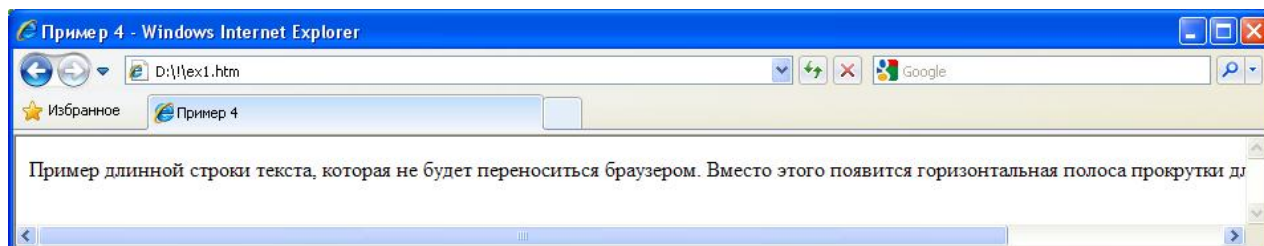


Рис. 1.4. Результат отображения примера 4

При работе с такими строками возникает проблема строк большой длины, когда пользователю приходится постоянно прокручивать горизонтальный скроллинг для чтения текста, что весьма неудобно. Для ограничения максимальной длины таких строк предусмотрен тег `<wbr>`, который указывает браузеру возможность переноса строки в том месте, где он стоит и будет использован только тогда, когда это действительно необходимо. Например, если строка помещается целиком во всю ширину, то перенос выполняться не будет, т.к. в этом нет необходимости, в противном случае строка будет поделена на части при выводе на экран.

1.3.4. Заголовки внутри HTML-документа

Наряду с общим заголовком страницы, заданным в теге `<title>` существует возможность добавлять подзаголовки непосредственно в тексте документа. Это выполняется с помощью тегов заголовков `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` и соответствующих закрывающих тегов, имеющих 6 уровней. При этом заголовок 1-го уровня самый большой, 2-го – поменьше и т.д. до 6-го уровня. Соответственно при расстановке заголовков рекомендуется использовать их в порядке возрастания, т.е. сначала `<h1>`, за ним `<h2>` и т.д. Также может быть несколько заголовков одного уровня на странице, например 2-3 заголовка уровня `<h1>`. Не рекомендуется использовать большое число заголовков в одном документе. Приемлемым количеством считается до 5 заголовков одного уровня.

Теги заголовков относятся к тегам уровня блоков, т.е. по умолчанию на одной строке с ними не может размещаться другая информация, например, документ вида

```

<body>
Текст перед заголовком<h1>Заголовок h1</h1>Текст после заголовка
</body>

```

на экране браузера будет выглядеть следующим образом:

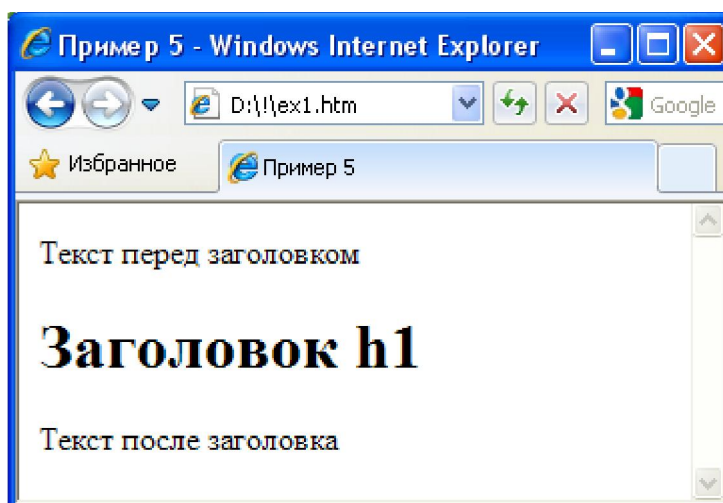


Рис. 1.5. Демонстрация тега <h1> как тега уровня блока

Как видно из примера под заголовок выделяется вся строка и никакая другая информация там не размещается. Аналогичное поведение справедливо для всех тегов уровня блоков. Далее при рассмотрении материала будет указываться на данное свойство тегов.

У тегов заголовков имеется необязательный параметр `align`, аналогичный по набору значений и действию с таким же параметром тега `<p>`:

- `left` – выравнивание по левому краю;
- `center` – выравнивание по центру;
- `right` – выравнивание по правому краю;
- `justify` – выравнивание по ширине страницы.

```
<html>
<head>
<title>Пример 6</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>

<body>
<h1>Заголовок 1</h1>
<h2 align=center>Заголовок 2</h2>
<h3>Заголовок 3</h3>
<h4 align=right>Заголовок 4</h4>
<h5>Заголовок 5</h5>
<h6 align=left>Заголовок 6</h6>
</body>
</html>
```

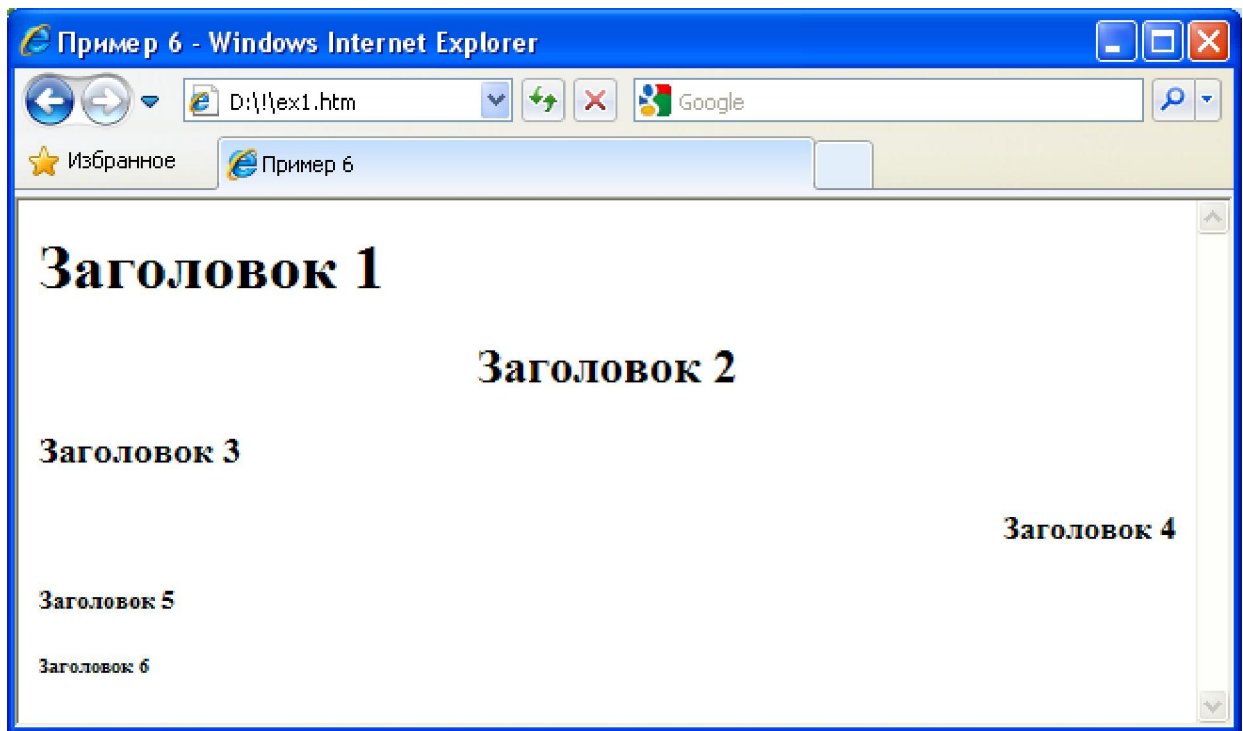


Рис. 1.6. Примеры отображения заголовков уровня от 1 до 6

1.3.5. Тег `<hr>` - горизонтальная линия

Для разделения текста документа на логические части помимо заголовков можно использовать горизонтальную линию, отделяющие одну часть документа от другой. Рисование такой линии выполняется браузером том месте документа, где встречается тег `<hr>`. Данный тег является тегом уровня блока и не требует завершающего тега, т.к. не является тегом-контейнером. Пример его использования представлен ниже:

```
...  
<body>  
Первая часть документа  
<hr>  
Вторая часть документа  
</body>  
...
```

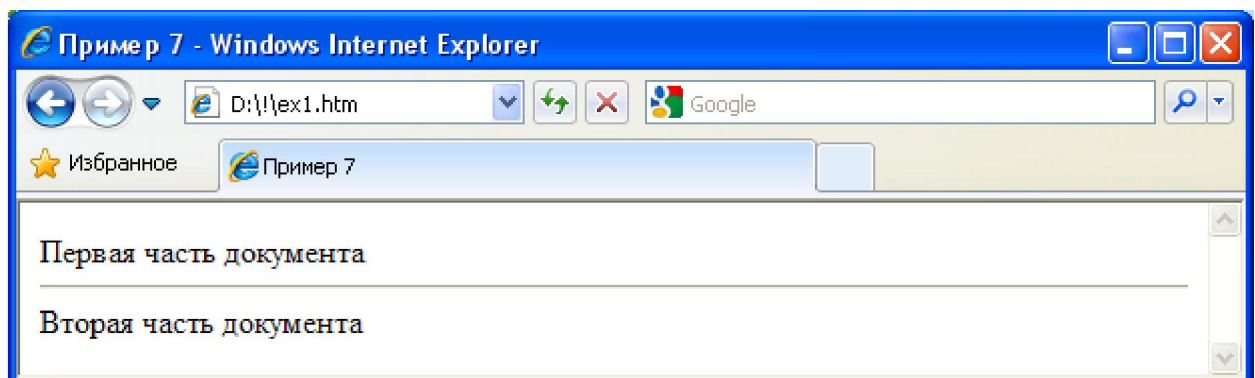


Рис. 1.7. Пример использования тега `<hr>`

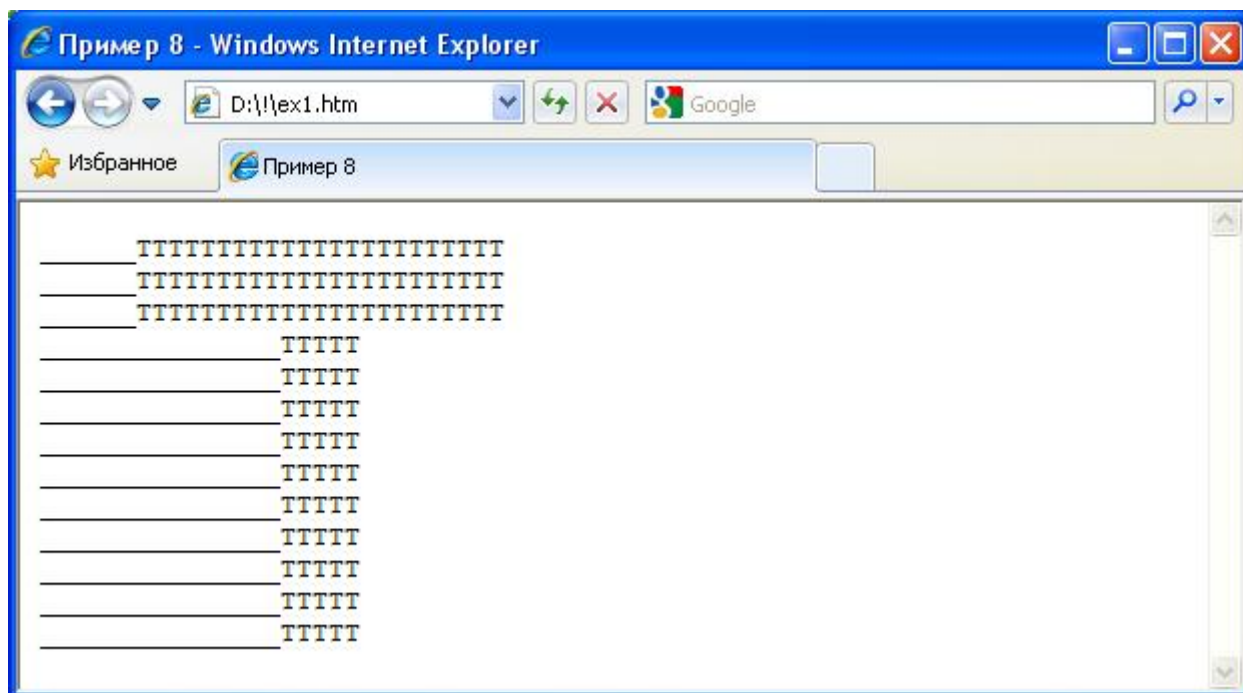


Рис. 1.8. Пример работы тега <pre>

Наиболее распространенным вариантом использования данного тега является вывод листингов языков программирования (например, Java, C++, Pascal, C#, JavaScript и т.д.) на экран браузера без необходимости их переформатирования. При этом следует учитывать, что текст всегда будет выводиться моноширинным шрифтом, что снижает гибкость при форматировании общего вида документа. Кроме того, внутри контейнера <pre> нельзя использовать теги уровня блоков <big>, , <object>, <small>, <sub> и <sup>. Также тег <pre> сохраняет исходное число пробелов между словами, в отличие от вывода текста вне этого блока, когда несколько подряд идущих пробелов всегда заменяются одним.

1.3.7. Теги логического уровня форматирования текста

Тег <code>

Данный тег рекомендуется использовать для выделения фрагментов программных кодов в HTML-документе. Как правило, такой текст отображается моноширинным шрифтом. Следует отметить, что в отличие от тега <pre>, данный тег не является тегом уровня блока и может включаться в строки обычного текста. Пример:

```
<p>Задание целочисленной переменной на языке C++: <code>int a = 0;</code>
```

Теги и <ins>

С помощью тега-контейнера `` рекомендуется отмечать текст как удаленный. Здесь предполагается, что по каким-либо причинам разработчику документа необходимо оставить удаленный текст на экране браузера, но пометить его как удаленный. Обычно такой текст браузеры отображают как перечеркнутый. Данный тег имеет два необязательных параметра:

`cite` – URL-адрес документа, поясняющего причину удаления текста;

`datetime` – дата редактирования в формате YYYY-MM-DDThh:mm:ssTZD.

```
<del cite="http://mysite.ru" datetime="2014-05-22T17:54:05+0.00">Это текст помечен как удаленный</del>
```

Аналогично используется тег-контейнер `<ins>`, который отмечает текст как добавленный (вставленный). Обычно такой текст помечается браузерами как подчеркнутый. Данный тег имеет те же необязательные параметры, что и тег ``:

```
<ins cite="http://mysite.ru" datetime="2014-05-22T17:54:05+0.00">Это добавленный текст</ins>
```

Тег ``

Данный тег используется для выделения важных фрагментов текста. Браузеры обычно отображают такой текст курсивом:

```
<em>Выделенный</em> фрагмент текста
```

Тег `<kbd>`

Тег-контейнер `<kbd>` служит для выделения текста, введенного с клавиатуры. Такой текст обычно отображается моноширинным шрифтом:

Для загрузки яндекс-поиска введите `<kbd>yandex.ru</kbd>`

Тег `<samp>`

Тег-контейнер `<samp>` отмечает текст как образец и обычно выводит моноширинным шрифтом и используется для форматирования текста, содержащих результаты действия программ, например:

```
Моя первая программа на C++ выводит текст <samp>Hello World!</samp>
```

Тег ``

Тег-контейнер `` служит для выделения важных фрагментов текста. Текст, выделенный данным тегом считается более важным, чем текст

выделенный тегом ``. При выводе браузеры обычно выделяют такой текст жирным шрифтом:

```
<strong>Пример текста, помеченный как важный</strong>
```

Тег `<var>`

Данный тег предназначен для выделения переменных программ и, как правило, отображается браузерами курсивом:

```
Переменная <var>a</var> имеет значение 5
```

Рассматривая описанные выше теги, можно заметить, что результат действия некоторых тегов совпадает и дает один и тот же результат в окне браузера при отображении. Напрашивается вопрос: для чего необходимо такое разнообразие тегов, если они по результатам работы дублируют друг друга? Дело в том, что разработчики языка HTML дали возможность составителям HTML-документов корректно выделить те или иные информационные части в расчете на то, что будущее поколение браузеров и поисковых систем будут анализировать их и предоставлять большие возможности «конечным» пользователям для восприятия корректно отформатированного текста. Поэтому уже сейчас рекомендуется применять эти и другие теги логического форматирования текста для улучшения конкурентоспособности таких HTML-документов в будущем.

1.3.8. Теги физического уровня форматирования текста

Тег ``

С помощью данного тега-контейнера выполняется выделение полужирным шрифтом части текста в HTML-документе, например:

```
<p>Это <b>текст с полужирным шрифтом</b> и обычный текст
```

На данный момент вместо тега `` рекомендуется использовать тег ``.

Тег `<i>`

Данный тег служит для выделения текста курсивом следующим образом:

```
<p><i>Фрагмент текста, выделенный курсивом</i> и обычный текст
```

Тег `<u>`

Выполняет подчеркивание части текста в окне браузера:

```
<p><u>Подчеркнутый текст</u>
```

Теги <strike> и <s>

Данные теги-контейнеры позволяют выводить перечеркнутый текст в окне браузера. Обычно вместо этих тегов используется тег , если зачеркнутый текст следует представлять как результат редактирования. Если же это просто элемент оформления, то следует использовать <strike> или <s>:

```
<p>Текст <strike>перечеркнутый</strike>
```

Теги <sub> и <sup>

С помощью данных тегов выполняется сдвиг текста относительно вниз и вверх соответственно. Это бывает удобно при оформлении математических выражений при простановке индексов или степеней:

```
<p>Оформление <sub>нижнего индекса</sub>  
<p>Оформление <sup>верхнего индекса</sup>
```

Тег

Данный тег-контейнер позволяет задавать свойства шрифта при отображении текста в окне браузера. В настоящий момент не рекомендуется использовать тег для форматирования документа, т.к. он нарушает принцип спецификации HTML 4.0 по разделению вида документа от его содержания. Вместо него лучше применять стилевое оформление, заданные в каскадных таблицах стилей (css), о которых речь пойдет ниже. Тем не менее рассмотрим данный тег, т.к. он бывает весьма полезен в простых документах и дает представление о свойствах шрифта.

Тег относится к последовательным элементам и не может включать в себя теги уровня блоков, например, <p>, <table>, <div> и т.п. Тег имеет следующие необязательные параметры:

Параметр	Пример	Описание
color="цвет"		Устанавливает цвет текста
face="шрифт1, шрифт2,..."		Устанавливает шрифт выводимого текста
size="число"		Устанавливает размер шрифта (целое число от 1 до 7)

```
<html>  
<head>  
<title>Пример тега font</title>
```

```

<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>

<body>
<p><font color="#0000CC">Синий текст</font>
<p><font color="#CC0000" face="Arial">Красный текст со шрифтом
Arial</font>
<p><font color="#00CC00" face="Courier New" size=4>Зеленый текст
со шрифтом Courier New и размером 4</font>
</body>
</html>

```

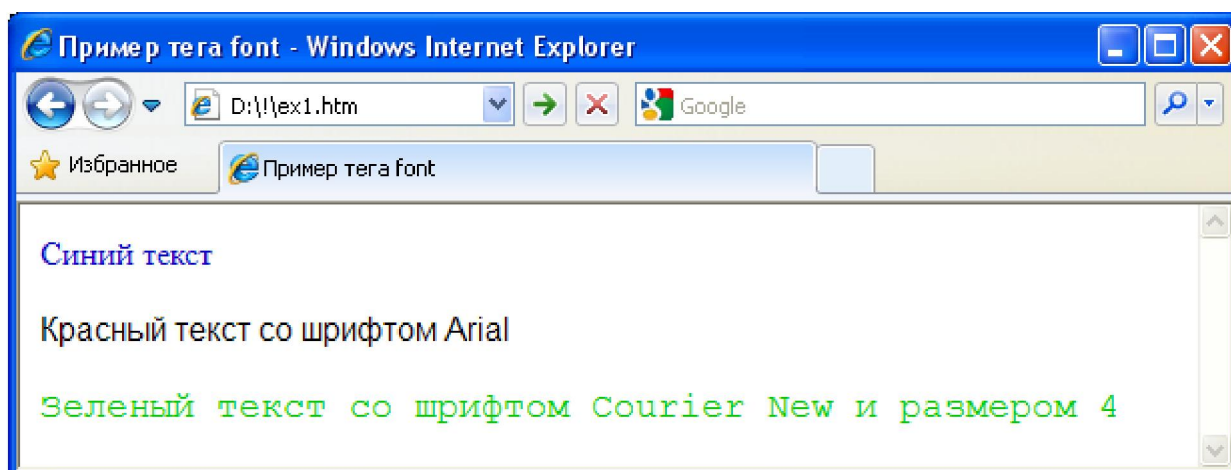


Рис. 1.9. Демонстрация работы тега

Тег <basefont>

Тег <basefont> задает значение свойств шрифта для всего текста документа и имеет те же свойства, что и тег , т.е. color, face и size. Данный тег не нуждается в закрывающем теге и может располагаться в любой части документа, включая раздел <head> и повторяться сколько угодно раз.

Следует отметить, что значения тега <basefont> могут переопределяться тегом и восстанавливаться после закрывающего тега , таким образом комбинируя возможные оформления в тексте HTML-документа.

Тег

С помощью данного тега выполняется вывод изображений в окно браузера. Тег имеет следующие наиболее распространенные необязательные параметры:

Параметр	Описание
align="bottom left middle right top"	Способ выравнивания изображения в документе
alt="текст"	Задает альтернативное описание изображение (отображается при

	неудачной загрузке изображения)
border="толщина рамки"	Устанавливает толщину рамки вокруг изображения. Чтобы убрать рамку нужно поставить значение 0
height="высота"	Высота рисунка при отображении (в пикселах или процентах)
hspace="отступ по горизонтали"	Задаёт величину отступа по горизонтали
longdesc="URL"	Указывает адрес страницы с аннотацией к изображению
src="URL"	Адрес загружаемого рисунка
vspace="отступ по вертикали"	Задаёт величину отступа по вертикали
width="значение"	Ширина рисунка при отображении (в пикселах или процентах)

Приведем пример тега `` для отображения графического файла `logotk.jpg`, находящегося по адресу `http://its.alnam.ru/images/`:

`<p>Пример логотипа: `



Рис. 1.10. Пример работы тега ``

Как видно из примера изображение вставляется в строку текста, т.е. является объектом `inline` уровня. При этом, используя параметр `align` можно устанавливать способ выравнивания изображения по отношению к тексту, например:

`<p>align=middle: `
`<p>align=top: `

```
<p>align=bottom: 
```

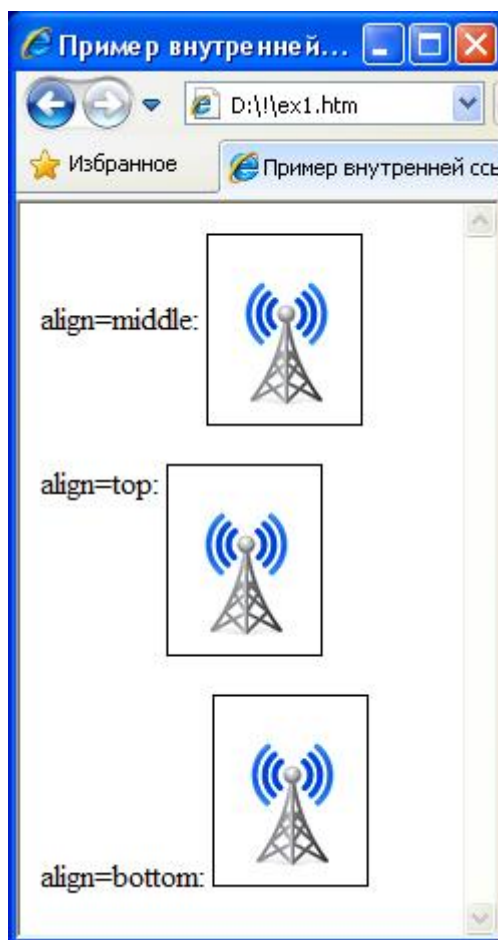


Рис. 1.11. Примеры разного способа выравнивания изображения

Если же значение параметра align установить равным left или right, то изображение окажется «прижатым» к соответствующему краю окна браузера, а текст будет обтекать это изображение (рис. 1.12). При всех других способах выравнивания изображение является «встроенным» в строку текста и составляет с ним единое целое.

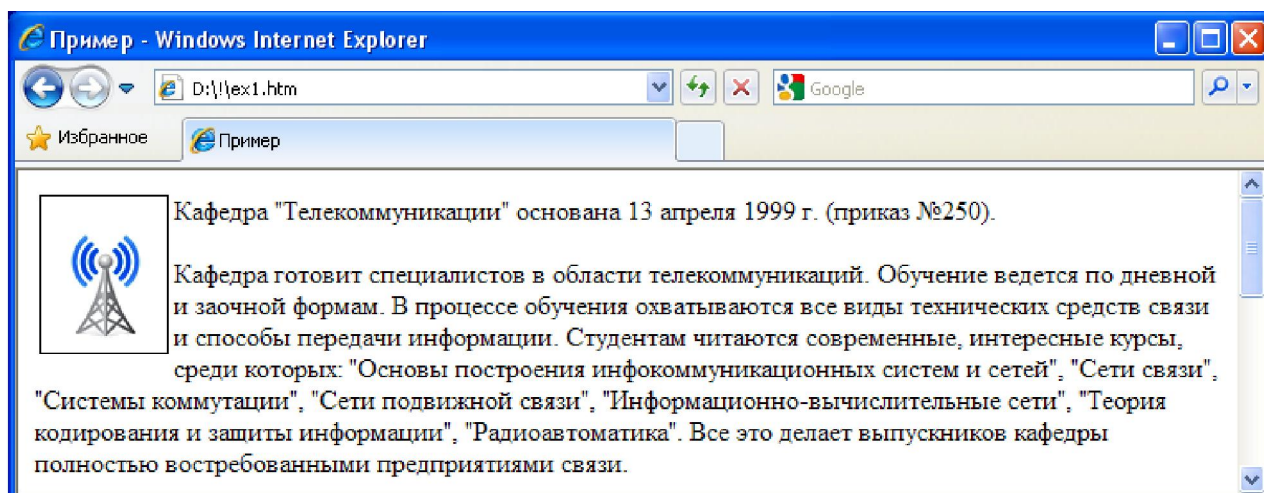


Рис 1.12. Пример работы параметра align=left тега

В приведенных примерах использовались изображения в форматах JPEG (расширение jpg) и GIF (расширение gif). Также допускаются форматы PNG, BMP. Другие форматы изображений поддерживаются не всеми браузерами. Учитывая разнообразие графических форматов, возникает вопрос: какой формат лучше всего выбирать для хранения графической информации? Как правило, руководствуются следующими соображениями: во-первых, чем меньше размер сжатого изображения, тем быстрее оно будет загружаться, и, во-вторых, чем детальнее изображение, тем лучше оно воспринимается визуально «конечным» пользователем. Эти два условия взаимоисключающие. Очевидно, что при увеличении детализации будет увеличиваться объем файла и наоборот. Поэтому на практике поступают следующим образом. Если изображение соответствует фотографии с большим числом цветов, переходов яркостей, то такое изображение лучше хранить в формате JPEG с некоторыми потерями в качестве, которые, как правило, практически не заметны для глаза. Если же изображение содержит четко структурированную информацию, например, скан текста, простые графические схемы, черно-белые (монохромные) рисунки, то такое изображение лучше хранить в формате PNG или GIF, которые не приводят к потерям в качестве при восстановлении, если изображение содержит не более 256 различных цветов.

Тег ``

Тег-контейнер `` предназначен для установки определенного стиля (формата) строчному фрагменту текста. Это тег inline уровня не приводит к разрыву текста и обычно используется для настройки собственных стилей, часто применяемых в HTML-документе. Для этого в таблице стилей (css) определяется нужный стиль и затем подключается через необязательный параметр `class` к тегу ``. (О том, как работать с каскадными таблицами стилей речь пойдет ниже).

```
<p>Пример работы <span style="color:red">тега span</span>
```

Тег `<div>`

Тег-контейнер `<div>` работает по тому же принципу, что и тег ``, но в отличие от него является элементом уровня блока, т.е. по умолчанию никакая другая информация справа или слева от него не располагается. Данный тег используется для назначения общих свойств отображаемому блоку информации, например, для задания общего цвета фона, шрифта, размера, цвета текста и т.п. Как правило, все свойства прописываются в каскадной таблице стилей (css) и подключаются через параметр `class` тега `<div>`.

```
<p>Пример работы <div style="color:red">тега div</div> - тега уровня блока.
```

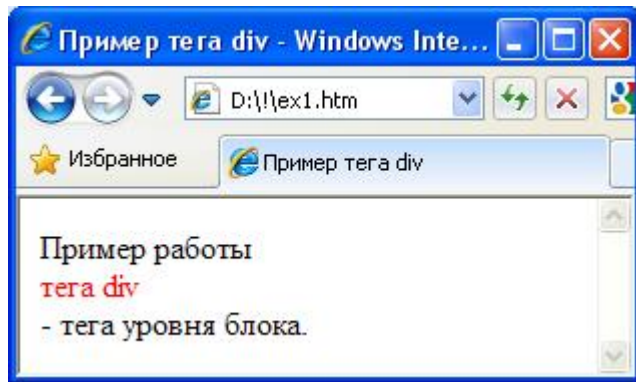


Рис. 1.13. Пример работы тега <div>

1.3.9. Специальные символы

Предположим в HTML-документе необходимо написать строку

Скалярное произведение двух векторов $\langle a, b \rangle$ равно нулю.

в которой используются символы \langle и \rangle . Проблема в том, что эти символы служат для записи тегов документа, и в частности, запись $\langle a$ будет интерпретироваться браузером как начало тега $\langle a \rangle$. В результате в окне браузера отобразится следующая информация:

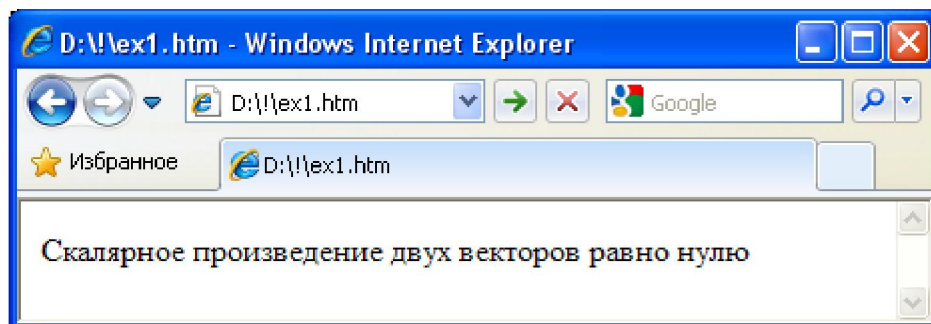


Рис. 1.14. Особенность использования скобок \langle и \rangle

Как видно из рис. 1.14 фрагмент строки $\langle a, b \rangle$ оказался скрыт браузером, из-за использования угловых скобок в тексте HTML-документа. Чтобы разрешить эту ситуацию в языке HTML применяются специальные символы для описания угловых скобок, пробелов, амперсандов и других символов, являющиеся служебными для HTML-документа.

Запись спецсимволов начинается с символа $\&$, затем идет имя символа (или его десятичное или шестнадцатиричное значение) и заканчивается запись точкой с запятой. В табл. 1.1 приведен список наиболее распространенных спецсимволов языка HTML.

Таблица 1.1. Список спецсимволов языка HTML

Запись спецсимвола	Результат вывода	Описание
$\<$	\langle	Знак «меньше»

>	>	Знак «больше»
 		Пробел
©	©	Знак copyright
&	&	Амперсанд
"	"	Кавычки

Таким образом, используя спецсимволы из табл. 1.1, строку начального примера следует оформлять в виде

Скалярное произведение двух векторов $\langle a, b \rangle$ равно нулю

для получения требуемого результата вывода.

1.3.10. Ссылки в HTML-документе

Отличительной особенностью HTML-страниц по отношению к стандартным текстам, напечатанным в книгах, газетах, журналах и т.п., является наличие ссылок на другие фрагменты этой же страницы или на сторонние документы. Наличие ссылок на страницах сайтов создают не просто текст, а гипертекст, т.е. текст, в котором пользователь может получить дополнительную информацию, кликнув мышкой на соответствующей ссылке. В результате грамотная организация текста и ссылок значительно упрощает пользователю навигацию и поиск нужной информации в сети Интернет.

Ссылка в HTML-документах состоит из двух частей: видимой в окне браузера частью, называемой **указателем ссылки** (или **анкором** от англ. anchor – якорь) и адресной частью, URL-адресом, говорящим браузеру куда следует перенаправить пользователя при использовании данной ссылки.

Создание ссылок в документе

Ссылка задается с помощью тега <a>, у которого имеется основной параметр href="URL-адрес", определяющий URL-адрес страницы перенаправления:

<p>Пример ссылки в HTML-документе.

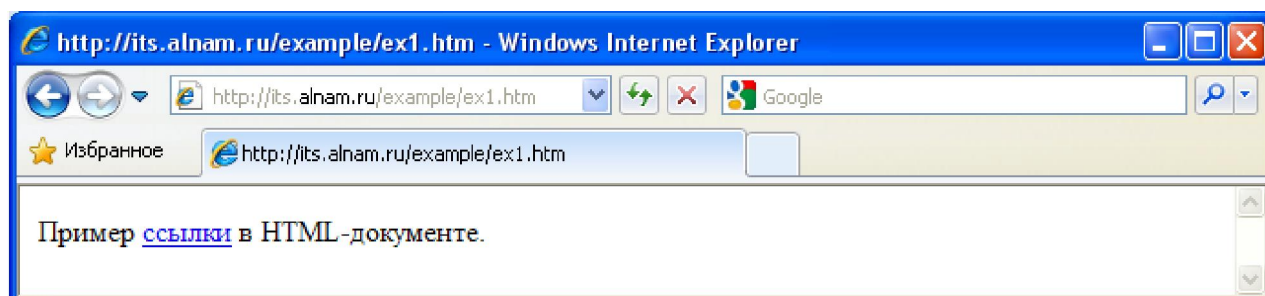


Рис. 1.15. Пример отображения ссылки в окне браузера документа, расположенного по адресу: http://its.alnam.ru/example/ex1.htm

Здесь `mypage.htm` – URL-адрес страницы, на которую перейдет пользователь при нажатии на ссылку, а слово «ссылки» - анкор ссылки, т.е. видимая ее часть в окне браузера. В качестве анкора ссылки чаще всего выступают фрагменты текста или изображения.

В приведенном примере используется относительный URL-адрес страницы перенаправления, т.е. указывается только часть пути, где расположен требуемый документ (в данном случае имя документа `mypage.htm`), остальная часть пути к документу достраивается браузером автоматически, используя базовый адрес страницы. По умолчанию перед именем `mypage.htm` добавится домен сайта, с которого был загружен документ (`http://its.alnam.ru`) плюс набор каталогов, в котором находится текущий документ (в данном примере каталог `example`). В итоге полный путь будет восстановлен в виде:

```
http://its.alnam.ru/example/mypage.htm
```

В случаях, когда страница `mypage.htm` находится по другому пути, то следует либо скорректировать относительный путь, либо указать абсолютный URL-адрес. Например, если страница `mypage.htm` находится по адресу

```
http://its.alnam.ru/mypage.htm
```

то ссылку из документа `http://its.alnam.ru/example/ex1.htm` можно организовать так:

```
<p>Пример <a href="http://its.alnam.ru/mypage.htm">ссылки</a> в HTML-документе.
```

или

```
<p>Пример <a href="../mypage.htm">ссылки</a> в HTML-документе.
```

В первом случае используется абсолютный URL-адрес страницы: `http://its.alnam.ru/mypage.htm`, а во втором – относительный: `../mypage.htm`. Здесь `‘../’` означает родительский каталог, т.е. каталог, в котором находится каталог `example`. По сути это означает исключение каталога `example` из пути URL-адреса страницы, что приводит к тому же адресу, что и в первом варианте записи ссылки.

Здесь следует отметить, что относительные URL-адреса страниц могут быть достроены браузером, исходя из базового адреса, указанного в теге `<baseurl>`. Данный тег располагается в разделе `<head>` текущего документа и имеет параметр `href="базовый адрес"`. Например, для корректной переадресации на страницу `mypage.htm`, можно создать следующий HTML-документ:

```

<!-- документ http://its.alnam.ru/example/ex1.htm -->
<html>
<head>
<title>Пример тега baseurl</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
<baseurl href="its.alnam.ru">
</head>

<body>
<p>Пример <a href="mypage.htm">ссылки</a> в HTML-документе.
</body>
</html>

```

В итоге относительный адрес будет иметь вид не
<http://its.alnam.ru/example/mypage.htm>,

а

<http://its.alnam.ru/mypage.htm>,

благодаря переопределению базового пути с помощью тега `baseurl`.

Внутренние ссылки

Помимо ссылок на другие страницы или сторонние ресурсы можно создавать ссылки на разные разделы текущего документа. Например, большой документ может содержать оглавление, состоящий из таких ссылок, щелкая по которым, пользователь автоматически переходит к выбранной части документа.

Для организации таких ссылок на странице создаются «маркеры», указывающие, куда должен быть перенаправлен пользователь при выборе той или иной ссылки. «Маркеры» создаются с помощью того же тега `<a>` с параметром `name="имя маркера"`, а ссылка на этот маркер имеет вид

```
<a href="#имя маркера">Внутренняя ссылка</a>
```

Следует обратить внимание, что значение параметра `href` в этом случае должно начинаться с префикса `#`, говорящий что это внутренняя ссылка страницы.

Ниже представлен HTML-документ с внутренней ссылкой.

```

<!-- документ http://its.alnam.ru/example/exlink.htm -->
<html>
<head>
<title>Пример внутренней ссылки</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>

<body>

```

```

<h1>Оглавление</h1>
<a href="#chapter1">Глава 1</a>

<h2>Введение</h2>
<p>Введение в основы языка HTML.
<a name="chapter1"></a>
<h2>Глава 1</h2>
<p>Первая глава книги
</body>
</html>

```

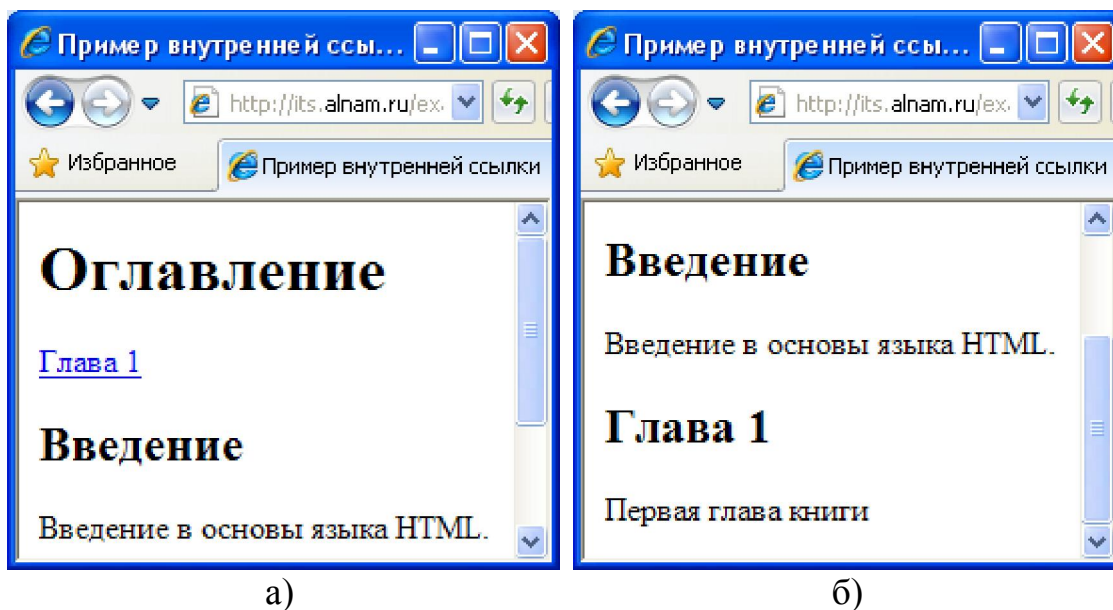


Рис. 1.16. Пример организации внутренней ссылки в HTML-документе: а – начальное отображение страницы; б – результат перехода по ссылке к главе 1.

Ссылки на другие ресурсы сети Интернет

В стандартной реализации ссылки, заданные в теге <a>, указывают на HTML-страницы в сети Интернет или на локальных ресурсах. Вместе с тем, существуют различные сервисы, имеющие иной формат передачи, приема, отображения данных отличный от HTML. Например, это может быть почтовый клиент, либо ftp-сеть передачи данных, либо telnet и т.п. Все эти и другие сервисы могут быть интегрированы в HTML-документ через ссылки. В этом случае также используется тег <a>, но в параметре href сначала указывается сервис, который подключается к документу, а затем, через двоеточие необходимые параметры этого сервиса. Например, если необходимо автоматизировать написание почтовых сообщений владельцу сайта, то на странице (например, контакты) следует создать следующую ссылку:

```

<a href="mailto:tkkaf@ulstu.ru">Написать письмо</a>

```

Здесь префикс `mailto:` говорит браузеру, что следует открыть почтовый клиент, а параметр `tkkaf@ulstu.ru` указывает по какому адресу следует отправить письмо.

Ниже в таблице приведены общеизвестные сервисы для интеграции их в HTML-документ.

Таблица 1.2. Ссылки на ресурсы сети Интернет

Название ресурса	Формат ссылки	Пример записи ссылки
Web-страница	<code>http://sitename</code>	<code>http://its.alnam.ru/mypage.htm</code>
e-mail	<code>mailto:address</code>	<code>mailto:tkkaf@ulstu.ru</code>
Newsgroup	<code>news:newsgroupname</code>	<code>news:news.questions</code>
FTP	<code>ftp://sitename</code>	<code>ftp://its.alnam.ru</code>
Gopher	<code>gopher://sitename</code>	<code>gopher://gopher.mysite.com</code>
WAIS	<code>wais://sitename</code>	<code>wais://wais.mysite.com</code>
TelNet	<code>telnet://sitename</code>	<code>telnet://its.alnam.ru</code>

СПИСКИ

В HTML-документах часто используется конструкция, представляющая собой список тех или иных данных. Например, это может быть список марок автомобилей, или знаков зодиаков, или рецептов блюд и т.п. Организовать такие списки можно, используя уже известные (из гл. 1) теги HTML. Однако гораздо более удобным и гибким инструментом являются специальные теги для представления списков в документе. В данной главе будут рассмотрены основные теги, позволяющие задавать списки различных типов.

2.1. Маркированный список

Наиболее простым в представлении является маркированный список, задаваемый тегом-контейнером ``. С его помощью формируются списки, у которых в качестве маркеров используются «отвлеченные» обозначения в виде квадратов, кружков, треугольников и др.

Для создания элементов списка используется тег ``, записанный внутри тегов ` ... `. Тег `` не требует закрывающего тега ``, но его использование не возбраняется спецификацией языка HTML. Обычно браузер сам определяет где заканчивается один тег `` и начинается другой. Кроме того, при определении элементов списка нет необходимости выполнять принудительный перенос строк с помощью тега `
` или тега `<p>`, эту задачу выполняет тег ``, который четко структурирует элементы, образуя из них вид списка в окне браузера. Ниже представлен пример HTML-документа, использующий маркированный список для представления знаков зодиаков.

```
<!--пример маркированного списка знаков зодиаков -->
<html>
<head>
<title>Пример маркированного списка знаков зодиаков</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>

<body>
<p>Знаки зодиаков:
<ul>
<li>Овен
<li>Телец
<li>Близнецы
<li>Рак
</ul>
</body>
</html>
```

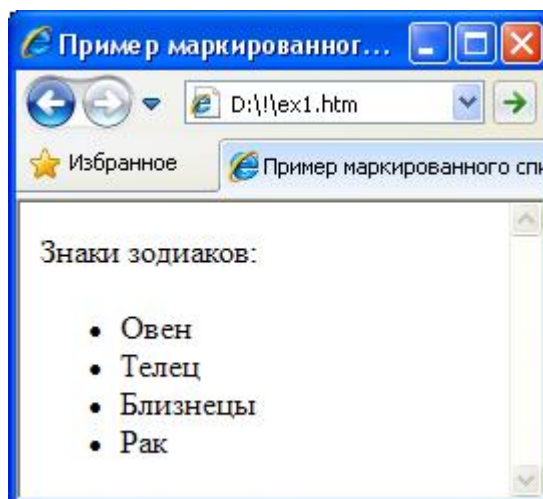



Рис. 2.1. Пример маркированного списка

Как видно из рис. 2.1. тег `` осуществляет отступ сверху и снизу, а также слева от края документа, выделяя таким образом список в тексте документа. Тег `` в свою очередь обеспечивает вывод каждого элемента с новой строки, образуя список.

Тег `` имеет необязательный параметр `type`, позволяющий задавать тип маркеров перед элементами списка. По умолчанию, в качестве маркеров используются закрашенные кружки (некоторые браузеры используют другие маркеры по умолчанию). Для изменения маркера по умолчанию, необходимо задать параметр `type` одним из следующих значений:

- disk – закрашенные кружки;
- circle – незакрашенные кружки;
- square – закрашенные квадратики.

Например, если в записанном выше примере переписать тег `` в виде

```
<ul type=square>
```

то получим следующий вид списка (рис. 2.2):

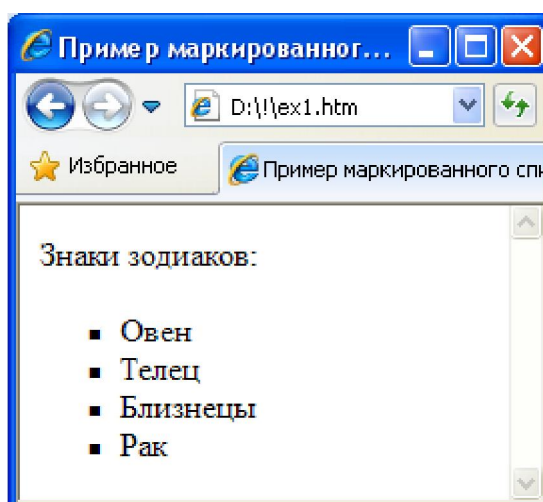


Рис. 2.2. Маркированный список с параметром `type=square`

Следует отметить, что внутри тега `` могут находиться любые другие теги языка HTML, в том числе и вложенные теги ``, образуя таким образом более сложный вид списков. Например, можно создать список сотрудников, с вложенным списком возраста, места работы, образования:

```
<html>
<head>
<title>Пример вложенных списков</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>

<body>
<ul>
<b>Список сотрудников</b>
<li>Иванов Иван Иванович
  <ul>
    <li>Возраст: 22
    <li>Место работы: УлГТУ
    <li>Образование: высшее
  </ul>
<li>Петров Петр Петрович
  <ul>
    <li>Возраст: 27
    <li>Место работы: УлГУ
    <li>Образование: высшее
  </ul>
</ul>
</body>
</html>
```

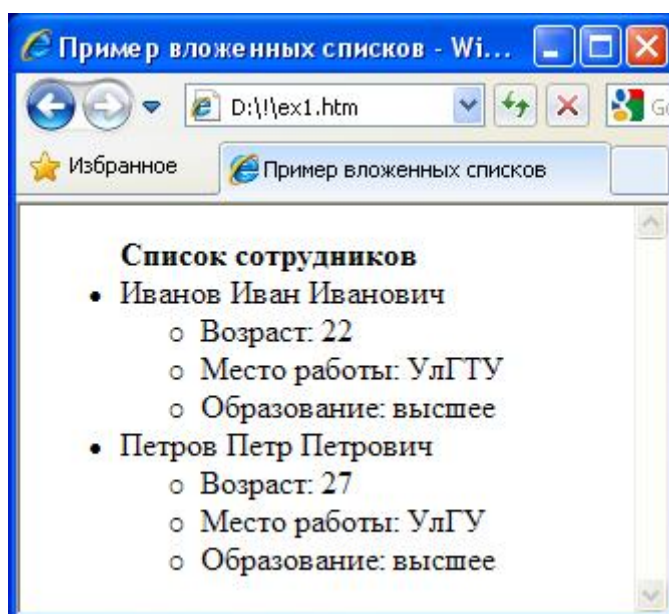


Рис. 2.3. Пример вложенных списков

Обратите внимание, что при отображении вложенных списков браузер автоматически меняет тип маркера и делает необходимые отступы для удобного визуального восприятия таких списков.

2.2. Нумерованный список

Данный тип списка отличается от маркированного тем, что в качестве маркера использует тот или иной вид перечисления элементов списка (по умолчанию в качестве перечисления используются арабские цифры, начиная с 1). Во всем остальном нумерованный список похож на маркированный.

Для создания нумерованного списка используется тег-контейнер ``, внутри которого располагаются элементы, заданные тегом ``, например:

```
<p>Наиболее яркие звезды, видимые с Земли:<br><ol><br><li>Сириус<br><li>Канопус<br><li>Арктур<br><li>Альфа Центавра<br></ol>
```

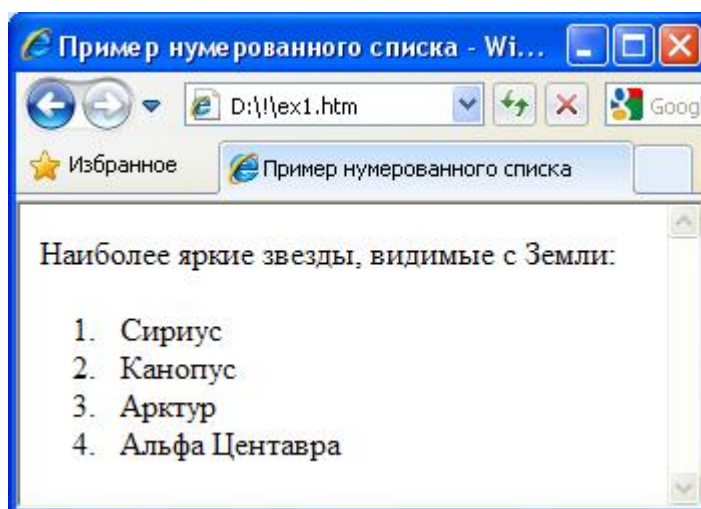


Рис. 2.4. Пример нумерованного списка

Тип перечисления можно менять с помощью параметра `type` тега ``. Параметр `type` может принимать следующие значения:

`type=A` – задает маркер в виде прописных латинских букв;

`type=a` – задает маркер в виде строчных латинских букв;

`type=I` – задает маркер в виде больших римских цифр;

`type=i` – задает маркер в виде малых римских цифр;

`type=1` – задает маркер в виде арабских цифр.

По умолчанию все браузеры используют маркер `type=1`.

Другой необязательный параметр тега `` `start` позволяет задавать начальное значение для нумерации списка, например,

```
<ol start=5>
```

при типе нумерации `type=1` начнет нумерацию с 5, при `type=I` – начнет нумерацию с V, при `type=A` – начнет нумерацию с символа E и т.д.

Тег `` в нумерованном списке может использовать параметр `value` для определения текущего номера элемента списка. При этом идущие следом за ним элементы будут увеличивать свой номер относительно него, например,

```
<ol>
<li>Сириус
<li>Канопус
<br>...
<li value=5>Арктур
<li>Альфа Центавра
</ol>
```

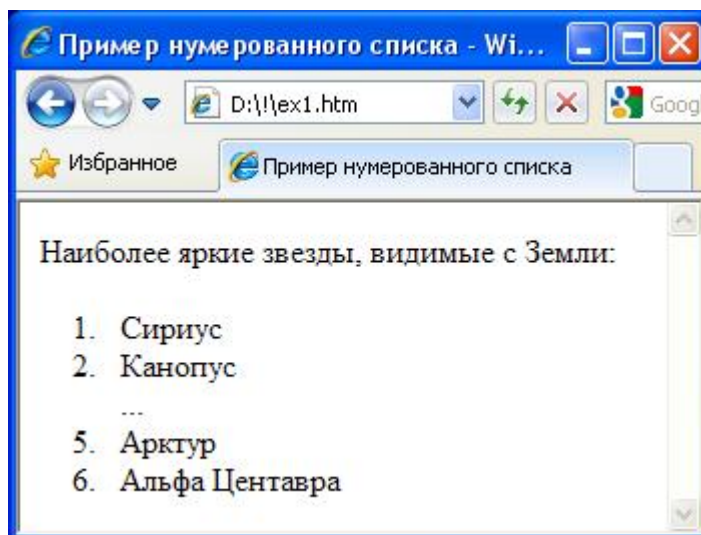


Рис. 2.5. Пример изменения нумерации при отображении списка

2.3. Список определений

Список определений служит для представления словарной информации в HTML-документах и по способу записи отличается от маркированного и нумерованного списков. Для создания списка определений используется тег-контейнер `<dl>`, внутри которого располагается тег `<dt>`, задающий термин, и тег `<dd>`, определяющий описание термина. Теги `<dl>` и `<dd>` не требуют закрывающих тегов. Ниже приведен пример списка определений:

```
<html>
<head>
<title>Пример списка определений</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>
```

```
<body>
<dl>
<dt>HTML
<dd>HTML (от англ. HyperText Markup Language — «язык
гипертекстовой разметки») — стандартный язык разметки документов
во Всемирной паутине
<dt>Тег
<dd>Тег (от англ. tag, читается; более правильное название —
дескриптор) — в SGML (в HTML, WML, AmigaGuide, языках семейства
XML) — элемент языка разметки гипертекста
</dl>
</body>
</html>
```

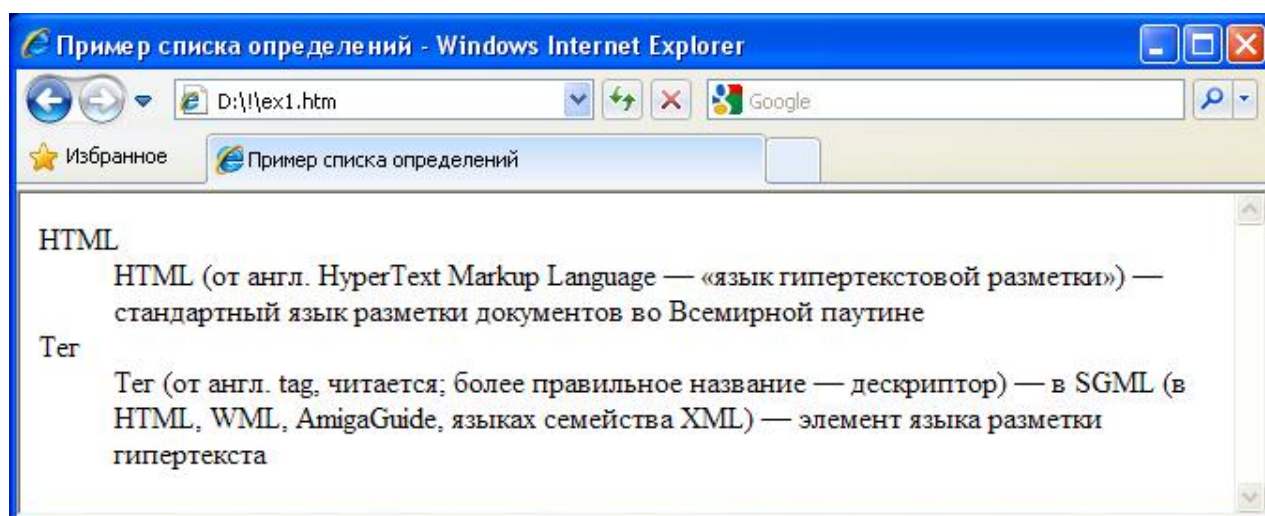


Рис. 2.6. Пример списка определений

Следует отметить, что внутри тега `<dt>` не допускается применение тегов уровня блоков, таких как `<div>`, `<p>`, `<h1>` – `<h6>` и др. Однако внутри тега `<dd>` такие теги допустимы, что позволяет создавать вложенные списки для списка определений.

ТАБЛИЦЫ

Таблицы обеспечивают важную часть в представлении информации. Разделение данных на столбцы и строки имеет широкое применение в практике оформления текстовых документов. Не являются исключением из этого правила и HTML-документы, которые имеют специальный тег `<table>` для создания таблиц.

В HTML-документах таблицы применяют не только для представления табличных данных, но и для создания разметки HTML-страницы. Например, многие страницы сайта имеют «шапку» (верхняя часть страницы), непосредственно информацию и «подвал» (нижняя часть страницы). Для разбивки HTML-документа на эти составные части хорошо подходят таблицы, у которых скрывается рамка и визуально создается впечатление единства всех составляющих частей. Таблицы удобны для создания разнообразного разбиения, т.к. ячейки таблицы могут содержать в себе и другие таблицы. В результате Web-мастеру не составляет большого труда организовать требуемую структуру документа и, затем, наполнить ее необходимым содержимым.

В данной главе речь пойдет о способах создания таблиц в HTML-документах, будут рассмотрены наиболее используемые свойства таблиц, представлены примеры разметки страницы с помощью таблиц.

3.1. Создание простейших таблиц в HTML-документах

Создание таблицы в HTML-документе начинается с тега `<table>` и заканчивается закрывающим тегом `</table>`. Таблицы всегда должны располагаться в разделе `<body>` документа. За тегом `<table>` обычно следует тег `<tr>`, описывающий строку таблицы. Затем, внутри тега `<tr>` располагается либо тег `<th>`, описывающий ячейку-заголовок таблицы, либо (и чаще всего) тег `<td>`, описывающий ячейку с данными таблицы. По сути, тег `<th>` или `<td>` разбивают строку таблицы на ячейки. Если нужно создать две ячейки в строке, то соответственно внутри тега `<tr>` должно быть два тега `<td>` (или `<th>`). Число строк в таблице определяется числом тегов `<tr>` внутри текущей таблицы. При этом общее число таблиц в документе может быть произвольным. По умолчанию каждая строка таблицы `<tr>` должна содержать одинаковое число ячеек. Если данное условие не будет выполнено, то браузер автоматически добавит недостающее число ячеек и выровнит таблицу. Теги `<tr>`, `<th>` и `<td>` могут не иметь соответствующих закрывающих тегов, но их применение рекомендуется для лучшей ориентации Web-мастеру в структуре таблицы.

Приведем пример простой таблицы, содержащей список студентов с экзаменационными отметками по разным дисциплинам.

```

<html>
<head>
<title>Пример простой таблицы</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>

<body>
<table border=1>
<tr>
    <th>Ф.И.О.</th>
    <th>Информатика</th>
    <th>Математика</th>
    <th>Физика</th>
    <th>Химия</th>
</tr>
<tr>
    <td>Иванов И.И.</td>
    <td>4</td>
    <td>3</td>
    <td>5</td>
    <td>4</td>
</tr>
<tr>
    <td>Петров П.П.</td>
    <td>3</td>
    <td>5</td>
    <td>4</td>
    <td>4</td>
</tr>
<tr>
    <td>Наместников С.М.</td>
    <td>5</td>
    <td>5</td>
    <td>5</td>
    <td>5</td>
</tr>
</table>
</body>
</html>

```

В приведенном примере у тега `<table>` использовался необязательным параметр `border` со значением 1, который задает толщину рамки вокруг таблицы. Данный параметр может принимать и большие значение: 2,3,..., однако толщина будет влиять только на рамку вокруг таблицы, на толщину рамки вокруг отдельных ячеек это не оказывает никакого влияния. Отличие составляет только значение `border=0`. В этом случае рамок ни вокруг таблицы, ни вокруг ячеек не будет.

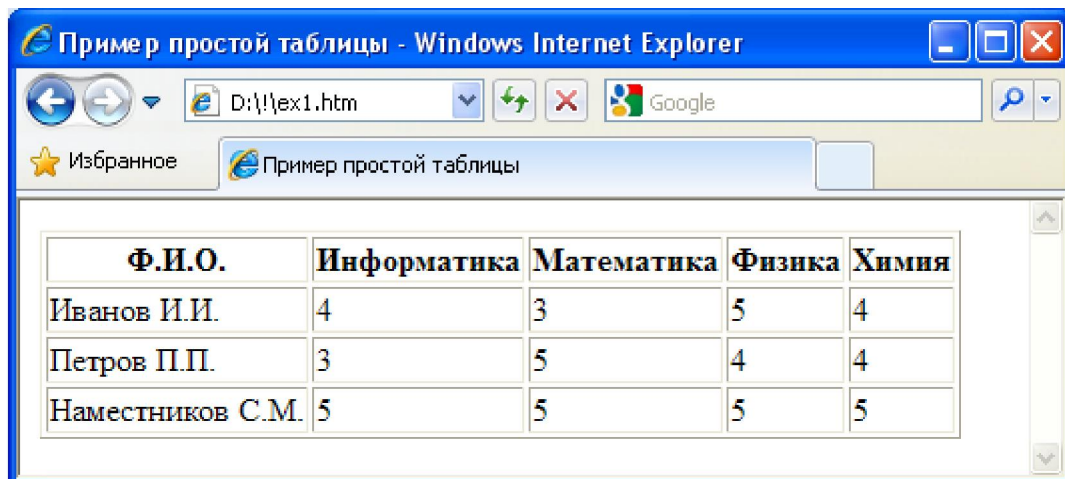


Рис. 3.1. Пример отображения простой таблицы

Из приведенного примера видно, что текст в ячейках заголовка таблицы (заданных тегом `<th>`) выделены полужирным и выровнены по центру. Текст в ячейках данных (заданных тегом `<td>`) не выделен полужирным и выровнен по левому краю ячейки. Также следует отметить, что по умолчанию вертикальное выравнивание во всех ячейках таблицы выполняется по центру.

Тег `<caption>`

Таблица в HTML-документе может иметь описание, располагающееся в необязательном теге `<caption>`. Данный тег обязательно должен идти сразу после тега `<table>` и имеет необязательный параметр

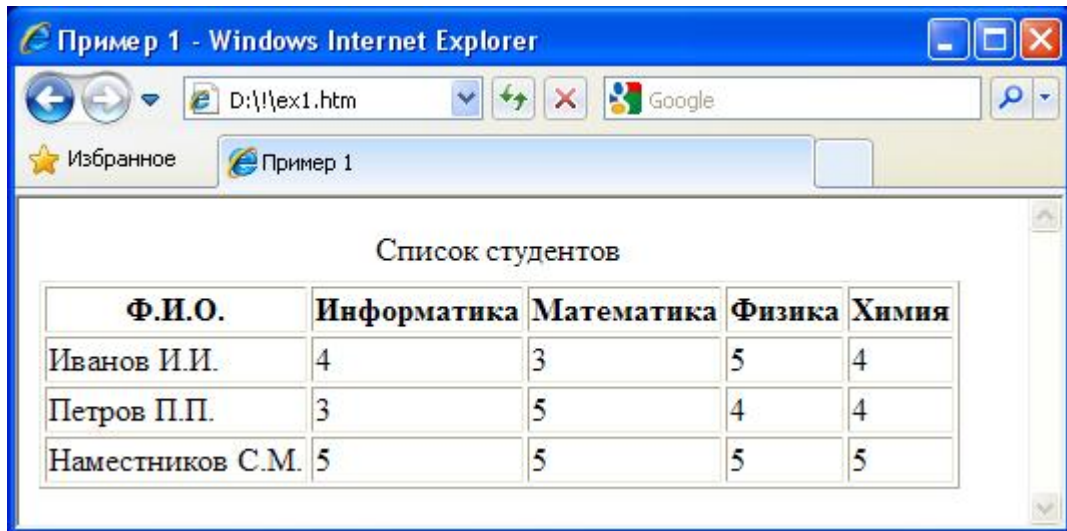
`align=left | right | top | bottom`

для выравнивания заголовка таблицы по левому, правому краю таблицы, либо определяет способ вывода сверху или снизу таблицы. Если параметр `align` не задан, то он принимается равным `top`. Ниже представлены примеры использования тега `<caption>` с различным набором параметров.

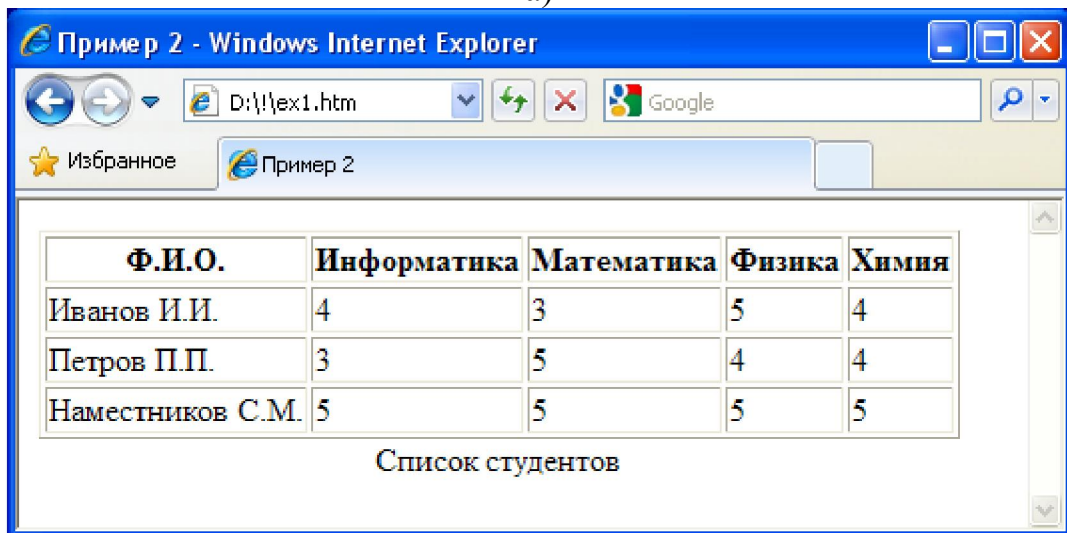
```
<!-- Пример 1 -->
<table border=1>
<caption>Список студентов</caption>
...
</table>
```

```
<!-- Пример 2 -->
<table border=1>
<caption align=bottom>Список студентов</caption>
...
</table>
```

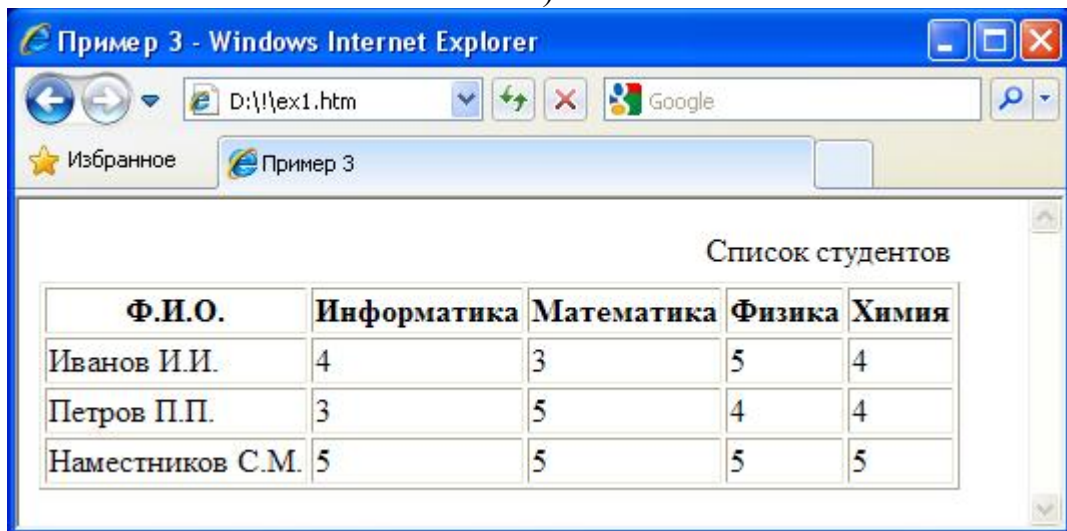
```
<!-- Пример 3 -->
<table border=1>
<caption align=right>Список студентов</caption>
...
</table>
```

а)



б)



в)

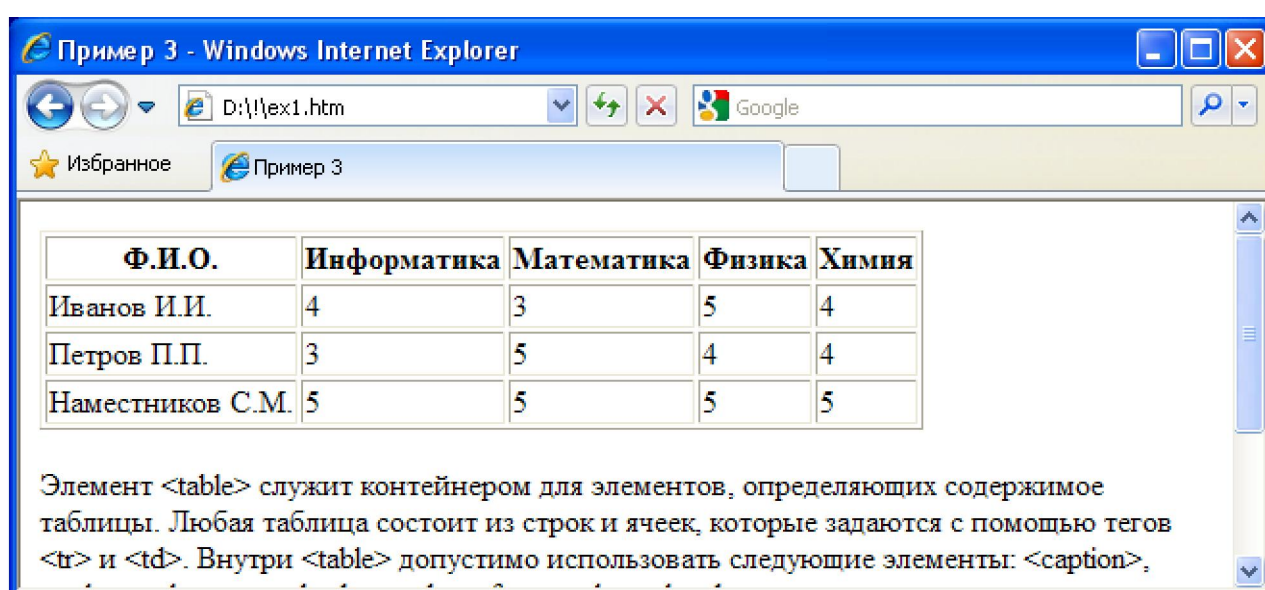
Рис. 3.2. Примеры использования тега <caption> с различными свойствами параметра align.

3.2. Параметры тега <table>

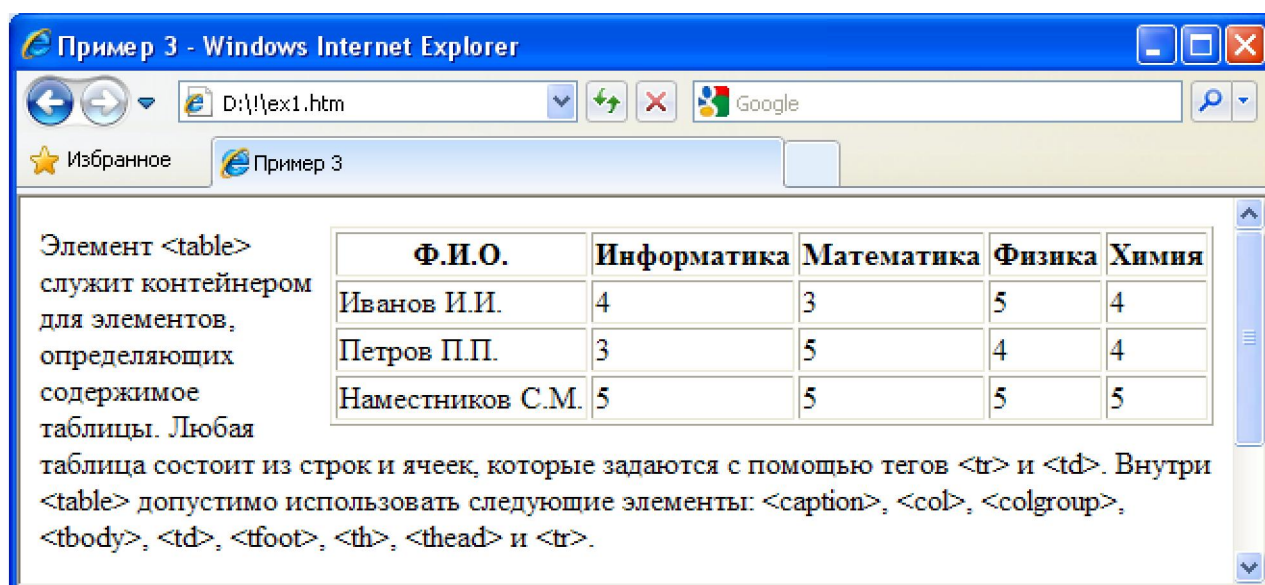
Тег <table> имеет следующие основные необязательные параметры:

align="left | center | right"

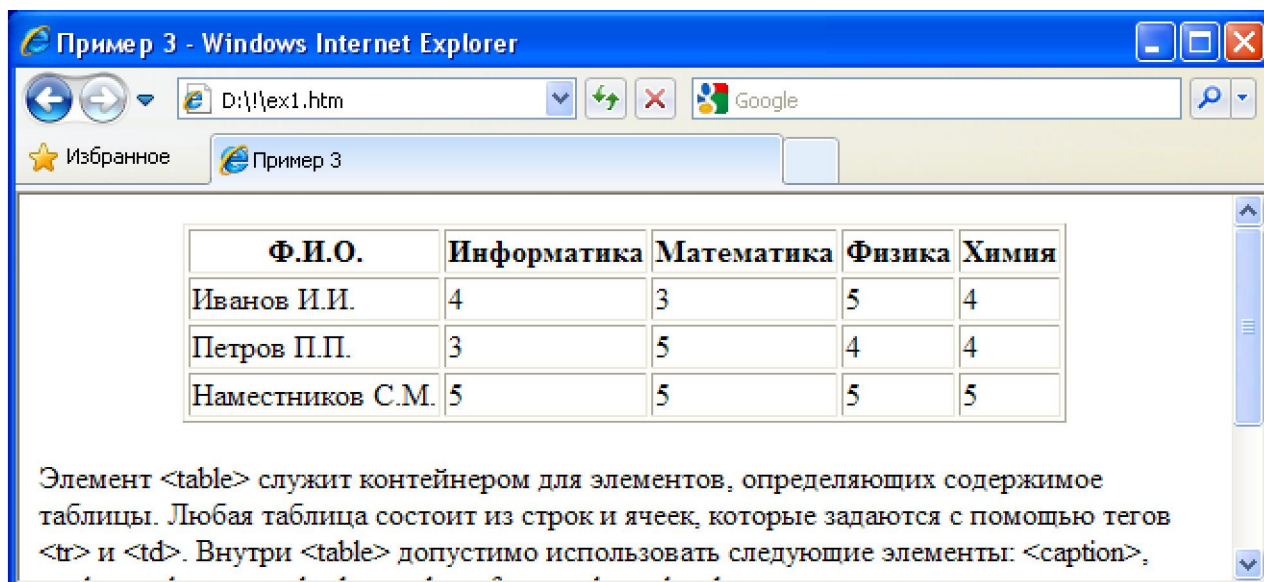
Выполняет горизонтальное выравнивание таблицы на странице документа. Если параметр align опускается при определении таблицы, то таблица располагается по левому краю и ее обтекание текстом не происходит (рис. 3.3, а). Если устанавливается значение align=left или right, то таблица ориентируется по левому или правому краю с обтеканием текста (рис. 3.3, б). Если же выравнивание выполняется по центру, то обтекание таблицы текстом не происходит (рис. 3.3, в).



а)



б)



в)

Рис. 3.3 Взаимное расположение таблицы и текста: а – параметр align отсутствует; б – параметр align=right; в – параметр align=center

background="URL"

Данный параметр служит для задания фонового изображения таблицы. Следует отметить, что изображение, указанное в значении URL, не масштабируется под высоту и ширину таблицы и будет отображена только соответствующая часть изображения. Если изображение меньше по ширине и/или высоте таблицы, то оно повторяется, заполняя соответственно всю ее площадь.

bgcolor="цвет"

Устанавливает цвет фона таблицы. В качестве значения данного параметра допустимо использовать как заданные цветовые константы, типа red, green, blue и др., так и шестнадцатиричные значения цветов в формате #RRGGBB, например,

```
<table bgcolor="#CC0000">...</table>
```

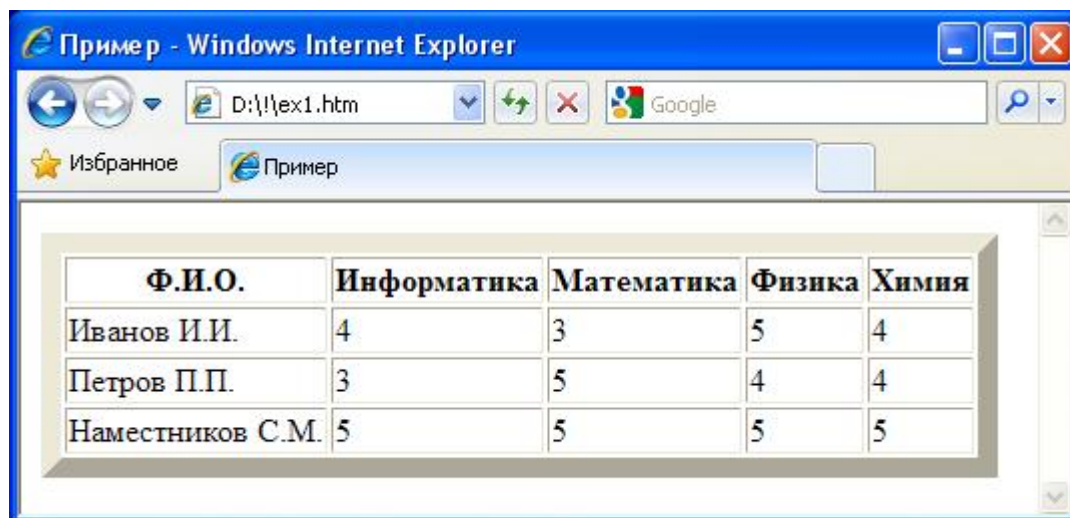
соответствует красному цвету фона таблицы. Часто цвет фона таблицы делают близким к цвету фонового изображения, т.к. он отображается, пока фоновое изображение не загрузится с сервера, либо в случаях ошибки загрузки фонового изображения.

border="толщина"

С помощью параметра border устанавливается толщина рамки вокруг таблицы. По умолчанию border равен 0 и рамки в таблице не рисуются.

Отображение таблиц без рамок бывает весьма полезным при создании разметки HTML-страницы, когда содержимое ячеек воспринимается пользователем как единый документ. Также это полезно при создании многоколоночных текстов или списков в тексте и т.п.

Следует отметить, что параметр `border` устанавливает толщину рамки только вокруг таблицы, но не вокруг ячеек. Например, при определении значения `border=10`, приведет к следующему визуальному эффекту (рис. 3.4).

The image shows a screenshot of a Windows Internet Explorer browser window. The title bar reads "Пример - Windows Internet Explorer". The address bar shows "D:\|ex1.htm" and the search bar contains "Google". Below the browser interface, a table is displayed with a thick, dark border. The table has five columns: "Ф.И.О.", "Информатика", "Математика", "Физика", and "Химия". There are three rows of data below the header.

Ф.И.О.	Информатика	Математика	Физика	Химия
Иванов И.И.	4	3	5	4
Петров П.П.	3	5	4	4
Наместников С.М.	5	5	5	5

Рис. 3.4. Таблицы с параметром `border=10`

`cellpadding="число"`

Данный параметр определяет расстояние в пикселах между границей ячейки и ее содержимым. По умолчанию значение параметра равно 1. Этот атрибут добавляет пустое пространство к ячейке, увеличивая тем самым ее размеры. Без `cellpadding` текст в таблице «налипает» на рамку, снижая тем самым его восприятие. Добавление же `cellpadding` позволяет улучшить читабельность текста. При отсутствии границ особого значения этот атрибут не имеет, но может помочь, когда требуется установить пустой промежуток между ячейками.

`cellspacing="число"`

Задает расстояние в пикселах между смежными ячейками таблицы (точнее между рамками ячеек) по горизонтали и вертикали одновременно. При отсутствии рамки, данный параметр аналогичен по действию параметру `cellpadding`. Если в таблице используется набор параметров

```
<table border=0 cellspacing=0 cellpadding=0>...</table>
```

то расстояния между ячейками будут отсутствовать и они сливаются, образуя единое неразрывное пространство. Это свойство таблиц очень удобно, например, для отображения графического меню, где в ячейках располагаются

фрагменты единого изображения, и, сливаясь вместе, они образуют естественное визуальное восприятие такой информации в целом.

cols="число"

Атрибут cols задает количество столбцов в таблице, помогая браузеру в подготовке к ее отображению. Без этого атрибута таблица будет показана только после того, как все содержимое таблицы будет загружено в браузер и проанализировано. Использование атрибута cols позволяет несколько ускорить отображение содержимого таблицы.

frame="значение"

Определяет способ рисования рамки вокруг таблицы. Данный параметр может принимать следующие значения:

- void – без рамки;
- border – рисование рамки вокруг таблицы;
- above – рисование рамки сверху таблицы;
- below – рисование рамки снизу таблицы;
- hsides – рисование рамки только сверху и снизу таблицы;
- vsides – рисование только вертикальных границ;
- rhs – рисование границы справа;
- lhs – рисование границы слева.

По умолчанию данный параметр имеет значение border.

rules="значение"

Сообщает браузеру, где отображать границы между ячейками. При этом толщина внешней рамки таблицы устанавливается с помощью атрибута border. По умолчанию рамка рисуется вокруг каждой ячейки, образуя тем самым сетку; толщина таких линий составляет 1px.

Данный параметр может принимать следующие значения:

- all – линия рисуется вокруг каждой ячейки таблицы;
- groups – линия отображается между группами, которые образуются тегами <thead>, <tfoot>, <tbody>, <colgroup> или <col>;
- cols – линия отображается между колонками;
- none – все границы скрываются;
- rows – граница рисуется между строками таблицы, созданных через тег <tr>.

width="значение"

Параметр width определяет желаемую ширину таблицы. По умолчанию браузеры вычисляют ширину и высоту таблиц автоматически, исходя из содержимого ячеек. Принцип формирования ширины исходит из того, что при

чтении документа его удобно просматривать сверху вниз, не выполняя дополнительный скроллинг по горизонтали документа. Таким образом, браузеры пытаются установить ширину таблицы не превышающую ширину HTML-документа, чтобы исключить горизонтальный скроллинг. Однако бывают ситуации, когда Web-разработчику необходимо указать строго определенное значение ширины. Для этого можно использовать параметр width, который определяет ширину таблицы в пикселах, либо в процентах, например, так:

```
<table width=200>...</table>  
<table width="100%">...</table>
```

В первом случае ширина будет установлена в 200 пикселей, во втором – она будет масштабироваться по всей ширине документа. Соответственно при изменении ширины документа, таблицы также будет менять свою ширину.

При использовании данного параметра всегда следует помнить, что задается именно желаемая ширина таблицы. То есть, если браузер не сможет уменьшить ширину до величины, указанной в width, то таблица будет отображена более широкой. Например, если в ячейках таблицы находятся изображения и их общая ширина больше ширины, указанной в width, то таблица будет увеличена по ширине для корректного отображения всех данных.

3.3. Создание более сложных таблиц в HTML-документах

Язык HTML предоставляет возможности для создания более сложного вида таблиц, чем просто разбиение всего пространства на строки и столбцы. В частности можно выполнять объединение ячеек в строках и столбцах, что приводит к более гибкому представлению данных.

Объединение ячеек выполняется с помощью параметров colspan и rowspan, которые задаются в тегах <td> и <th>. Параметр colspan определяет число объединяемых ячеек в строке, а rowspan – число объединяемых ячеек в строках. Например, если требуется создать таблицу вида:

то этого можно достичь с помощью следующего кода:

```
<table border=1>  
<tr>  
  <td>ячейка 1</td>  
  <td colspan=2> ячейка 2</td>  
</tr>  
<tr>
```



```

        <td rowspan=2>ячейка 3</td>
        <td> ячейка 4</td>
        <td> ячейка 5</td>
</tr>
<tr>
        <td> ячейка 6</td>
        <td> ячейка 7</td>
</tr>
</table>

```

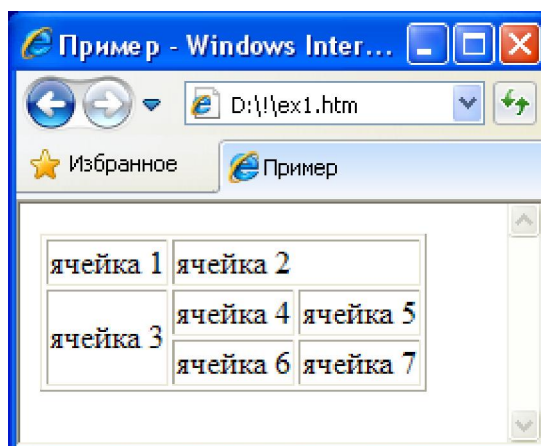


Рис. 3.5. Пример объединения ячеек по горизонтали и вертикали

3.4. Форматирование данных в ячейках таблиц

Теги `<tr>`, `<td>` и `<th>` имеют набор необязательных параметров, позволяющие форматировать данные внутри ячеек таблицы. Так, параметры `align` и `valign` задают способ выравнивания содержимого внутри ячеек по горизонтали и вертикали соответственно. Значения данных параметров следующие:

`align=left | right | center;`

`valign=top | bottom | middle.`

По умолчанию `align=left`, а `valign=middle`, т.е. выравнивание выполняется по левому краю ячейки и центрирование по вертикали.

Следует учитывать, если данные параметры установлены для тега `<tr>`, задающего строку таблицы, то выравнивание распространяется на все ячейки данной строки, если они не переопределены внутри тегов `<td>` этой строки. Например,

```

<table border=1 width="100%">
<tr align=right>
        <th>Заголовок 1</th>
        <th align=left>Заголовок 2</th>
        <th>Заголовок 3</th>
</tr>
<tr align=center>
        <td>Ячейка<br>1</td>
        <td valign=top>Ячейка 2</td>

```

```

        <td>Ячейка 3</td>
    </tr>
</table>

```

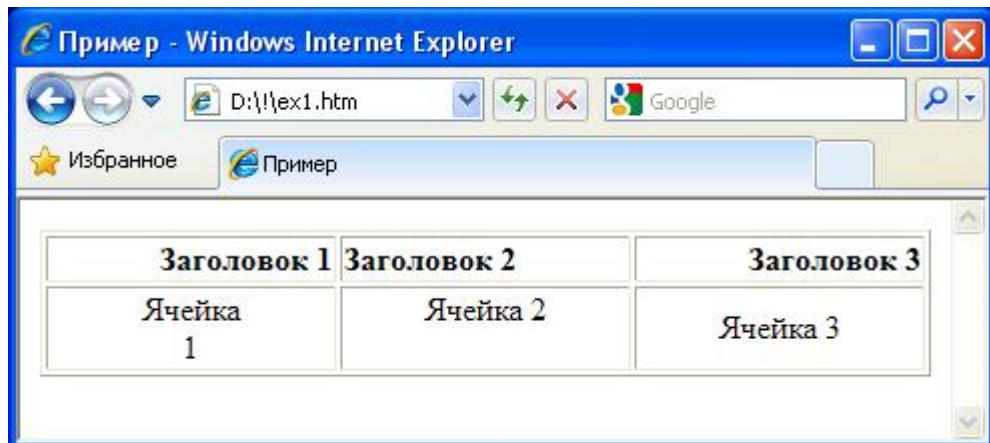


Рис. 3.6. Пример выравнивания данных внутри ячеек таблицы

Для определения цвета фона ячейки используется атрибут bgcolor со значением цвета. Например,

```

<table border=1 width="100%">
<tr align=center bgcolor="#f0f0f0">
    <th>Заголовок 1</th>
    <th>Заголовок 2</th>
    <th>Заголовок 3</th>
</tr>
<tr align=center>
    <td bgcolor="#CC0000">Ячейка<br>1</td>
    <td bgcolor="#00CC00" valign=top>Ячейка 2</td>
    <td bgcolor="#0000CC">Ячейка 3</td>
</tr>
</table>

```

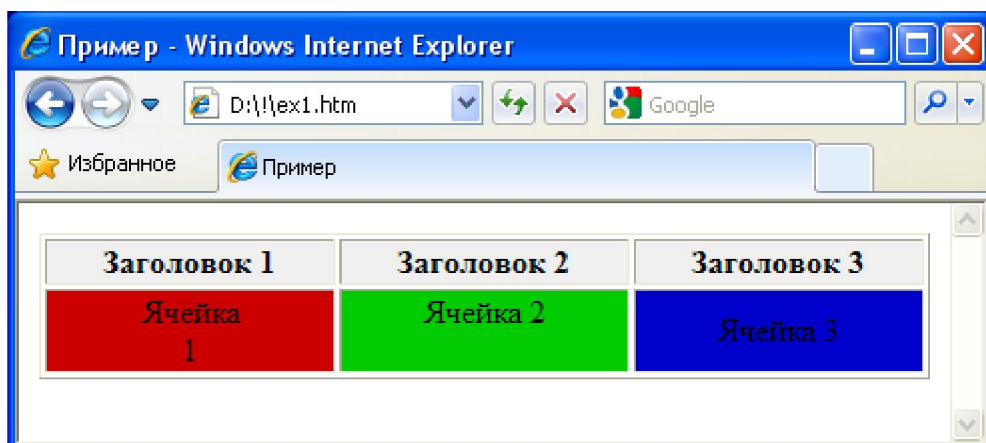


Рис. 3.7. Пример использования атрибута bgcolor

Следует отметить, что внутри ячеек таблицы могут использоваться любые теги языка HTML, включая и другие таблицы, создавая, таким образом,

вложенные таблицы. Следовательно, для более детального форматирования информации внутри ячеек можно использовать, например, теги ``, ``, `` и др.

КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ

Современные принципы создания HTML-документов подразумевают четкое разделение содержания документа от его вида (стилей форматирования). Это упрощает его редактирование, уменьшает объем загружаемой страницы, сокращает время создания документа, представление одного и того же документа с различными стилями. Отделение стилей форматирования от содержания осуществляется при помощи каскадных таблиц стилей (от англ. Cascading Style Sheets, сокращенно CSS). Здесь слово «каскадные» означает возможность использования набора таблиц стилей для представления одного документа. Браузер объединяет наборы таблиц по установленным правилам, в соответствии с заданными приоритетами и отображает документ в своем окне на основе полученных стилей.

Каскадные таблицы стилей представляют собой текстовый файл, в котором определены **селекторы** и их **определения**. В качестве селектора могут выступать теги языка HTML, классы, идентификаторы. Определение селектора задается внутри фигурных скобок парой: <свойство>: <значение>. Например, чтобы задать красный цвет шрифта для тегов <h1> следует записать такое определение:

```
h1 {color: red}
```

Здесь h1 – это селектор, color – свойство, red – его значение. Ниже будут более подробно рассмотрены различные селекторы и наборы свойств, с помощью которых можно манипулировать видом документа в окне браузера.

4.1. Встраивание CSS в HTML-документ

Каскадные таблицы стилей применяются к документу только в том случае, если они подключены к нему. Это можно сделать различными способами, которые могут комбинироваться в пределах одного документа. Распространенным вариантом является подключение таблиц в разделе head, записанных в текстовых файлах с расширением css:

```
<head>  
<title>Пример подключения css</title>  
<link rel="stylesheet" type="text/css" href="css/styles.css" >  
</head>
```

При этом число подключаемых таблиц неограниченно. В данном примере подключается одна таблица из файла styles.css, находящаяся в каталоге css, т.е. общий путь доступа к этому файлу имеет вид: http://домен/css/styles.css.

Таблицы стилей можно внедрить непосредственно в тело документа. Обычно это выполняется в разделе <head>, но может быть описано и в любой

другой части документа. Главное, чтобы определение таблицы стилей предшествовало их использованию. Для внедрения CSS в тело документа используется следующая запись:

```
<head>
<title>Пример внедрения css</title>
<style type="text/css">
h1 {color: red}
</style>
</head>
```

Следующим вариантом является импортирование таблиц стилей в HTML-документ с помощью свойства

```
@import: url(URL на css-файл);
```

Данное свойство может быть записано внутри тега `<style>`, либо внутри css-файла, образуя, таким образом, каскад из таблиц. Причем свойство `@import` обязательно должно идти сразу после тега `<style>` или являться самой первой строкой в css-файле. При этом допускается использование нескольких подряд идущих свойств `@import`:

```
@import: url(URL на 1-й css-файл);
@import: url(URL на 2-й css-файл);
```

Например,

```
<html>
  <head>
    <meta charset="windows-1251">
    <title>Импорт стиля</title>
    <style>
      @import url("/style/main.css");
      @import url("/style/palm.css");
    </style>
  </head>
  <body>
    <p>...</p>
  </body>
</html>
```

Следует отметить, что использовать свойство `@import` следует крайне осторожно, т.к. в ряде случаев такой способ подключения может приводить к ошибочным ситуациям, например, при использовании JavaScript совместно с CSS может случиться, что документ будет загружен раньше, чем необходимая импортируемая таблица. В результате сценарий не сможет отработать так, как это задумывалось разработчиком. Без крайней необходимости рекомендуется

вместо @import использовать подключение через тег <link> в разделе <head> (первый описанный вариант).

Последний вариант подключения таблиц стилей осуществляется через атрибут style, присутствующий в каждом теге языка HTML. Например, можно указать синий цвет шрифта у конкретного тега <p> следующим образом:

```
<p style="color: blue;">Синий цвет</p>
```

Однако такой вариант не отвечает требованиям к разделению содержания документа и его вида, поэтому его следует применять в особых, частных случаях.

4.2. Единицы измерения в таблицах стилей

Чтобы корректно задавать значения свойств в таблицах стилей необходимо знать, какими единицами измерения можно оперировать при их описании. В табл. 4.1. приведены основные единицы измерения, применяемые в CSS.

Таблица 4.1. Единицы измерения в CSS

Единицы измерения	
Относительные	Абсолютные
em – высота шрифта элемента	in – дюйм (1 in = 2.54 см)
ex – высота буквы x	cm – сантиметр
px – пиксел	mm – миллиметр
% - процент	pt – пункт (1 pt = 1/72 in)
	pc – пика (1 pc = 12 pt)

Помимо указанных единиц можно использовать числа как целочисленные, так и с плавающей точкой. Дополнительно, при описании цветов, в CSS предусмотрены специальные константы:

aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow

и шестнадцатиричная форма записи числа:

#RRGGBB

Например, чтобы задать красный цвет шрифта, можно воспользоваться следующими значениями:

```
p {color: red} или p {color: #CC0000}
```

Также допускается задавать цвет с использованием функциональных записей вида rgb и rgba:

```
p {color: rgb(255,255,255)}  
p {color: rgba(200,200,200, 0.4)}
```

Во втором случае задается серый цвет с прозрачностью 40% единиц (1 – отсутствие прозрачности, 0 – полностью прозрачный).

4.3. Селекторы-теги и формат записи свойств в CSS

В предыдущем параграфе был рассмотрен селектор `h1`, для которого назначался цвет шрифта. Аналогичным образом в качестве селектора можно указать любой другой тег языка HTML, например,

```
p {color: #000000;}  
p {font-size: 14px;}  
p {background: #ffffff;}
```

Здесь для тега абзаца `<p>` заданы наборы свойств: цвет (`color`) – черный; размер шрифта (`font-size`) – 14 пунктов; фон (`background`) – белый. Теперь при использовании любого тега `<p>` в документе тексту будут назначены перечисленные свойства.

Указывать каждый раз один и тот же тег, для которого задаются различные свойства, не очень удобно, поэтому в CSS допускается группировать свойства, относящиеся к одному тегу, следующим образом:

```
p {  
    color: #000000;  
    font-size: 14pt;  
    background: #ffffff;  
}
```

и это будет аналогично записи выше.

Если требуется к разным тегам применить один и тот же набор свойств, то в этом случае можно указать теги через запятую, а затем определить набор свойств, как это показано ниже:

```
body, p, b {  
    font-size: 14pt;  
    font-weight: normal;  
    font-family: Arial;  
}
```

В приведенном примере тегам `<body>`, `<p>` и `` назначаются свойства шрифта (`font`): размер – 14 пунктов; стиль шрифта (`font-weight`) – нормальный; имя шрифта (`font-family`) – Arial. Все эти свойства в CSS можно задать одной строкой с помощью свойства `font`, объединяющий в себе все свойства шрифта:

```
body, p, b {
    font: normal 14pt Arial;
}
```

и их порядок должен быть только таким: стиль, размер, имя шрифта.

Часто в HTML-документах теги вложены один в другой. Например, в тег абзаца `<p>` может быть вложен тег `` для выделения полужирным фрагмента текста. В этом случае тег `<p>` называется родителем, а тег `` - потомком:

`<p>Пример текста с вложенным тегом`

При этом возникает вопрос: что будет происходить, если для тега `<p>` определены свойства в таблице стилей, а для тега `` нет? В этом случае теги-потомки наследуют свойства тега-родителя, т.е. если указан стиль

```
p {color: blue}
```

то на экране браузера будет отображена информация в таком виде (рис. 4.1):

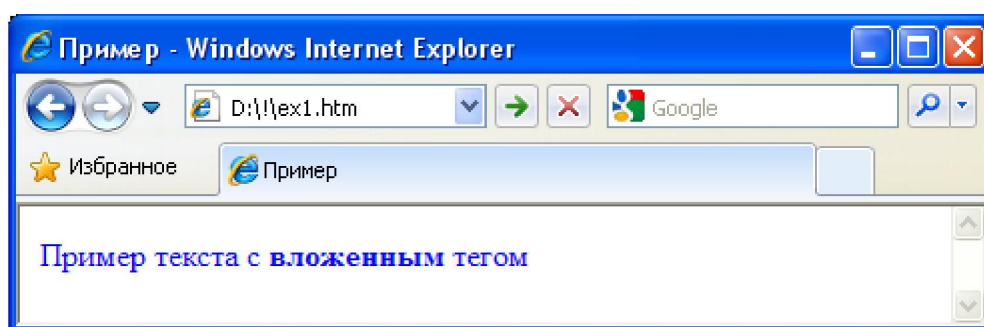


Рис. 4.1. Пример наследования свойств тегом-потомком

По этому правилу применяются все свойства из заданной таблицы CSS к тегам-потомкам.

Наследование создает естественность при использовании стилей. Однако встречаются ситуации, когда необходимо, чтобы вложенный тег имел свойства отличные от наследуемых, например, чтобы в вышеприведенном примере текст в теге `` не менял своего цвета. Этого можно добиться, используя контекстные селекторы, т.е. указать браузеру применять заданный стиль для тега, только тогда когда он вложен в один или несколько других тегов, например, так:

```
p b {color: black}
```

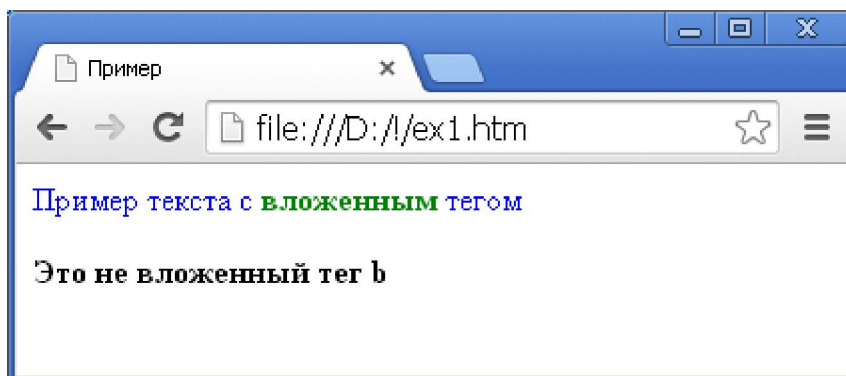


Рис. 4.2. Пример использования контекстных селекторов

4.4. Селектор CLASS

В приведенных выше примерах, в качестве селекторов использовались теги языка HTML и, соответственно, в HTML-документе они применялись с указанными стилями. Но что делать, если нужно применить тот или иной стиль лишь к некоторому числу тегов на странице, например к первому тегу `<p>`, а ко всем остальным тегам `<p>` не применять? Этого можно достичь несколькими способами. В самом простом случае использовать атрибут `style` у первого тега `<p>` и задать нужное свойство, например, так:

```
<p style="color:#00cc00">Это первый тег p</p>
<p>Это второй тег p</p>
```

Однако такая запись стилей непосредственно внутри тега приводит к объединению содержания страницы и ее виду, что в итоге усложняет последующее редактирование такого документа. Лучше задать необходимый стиль в таблице CSS и подключить его только к нужным тегам на странице. Для этого используется атрибут `class`, существующих у всех тегов языка HTML. Через этот атрибут можно указывать стили по их имени, используемые в данном теге. Это выглядит следующим образом. В таблице CSS задается стиль в виде:

```
.mystyle {
    color: black;
    font-size: 14px;
    background: #f0f0f0;
}
```

и подключается к первому тегу `<p>` с помощью атрибута `class`:

```
<p class="mystyle">Это первый тег p</p>
<p>Это второй тег p</p>
```

в итоге стиль `mystyle` будет применен только к первому тегу `<p>` HTML-документа. Аналогичным образом с помощью атрибута `class` можно подключить этот стиль только к выбранным тегам на странице.

Здесь следует обратить внимание, что перед именем стиля в CSS стоит точка. Точка означает, что данное имя является именем класса и будет подключаться к тегам через атрибут `class`. Если убрать точку перед именем, то документ будет отображен со стилями по умолчанию, т.к. класс `mystyle` не будет идентифицирован браузером и соответственно не подключится к тегу. При этом никаких сообщений об ошибке выдаваться не будет.

Если требуется определить класс исключительно для одного определенного тега языка HTML, то его следует указать перед именем класса, как это показано ниже:

```
p.mystyle {
    color: black;
    font-size: 14px;
    background: #f0f0f0;
}
```

и подключить в HTML-странице:

```
<p class="mystyle">Это первый тег p</p>
<p>Это <span class="mystyle">второй</span> тег p</p>
<b class="mystyle">Это тег b</b>
```

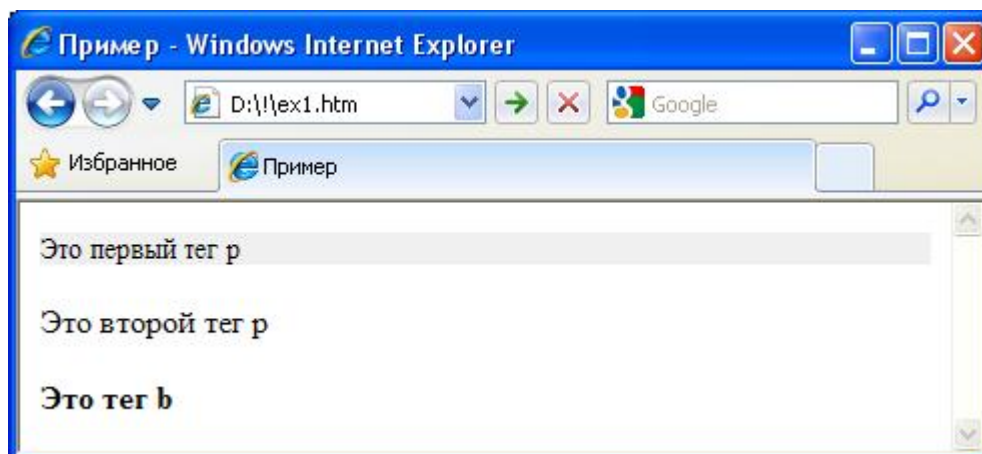


Рис. 4.3. Пример задания класса для тега `<p>`

Из рис. 4.3. видно, что заданные стили применяются только к тегу `<p>`, хотя подключаются и в тегах `` и ``. Это свойство удобно использовать для селекции тегов, к которым применяются стили.

Еще одной полезной особенностью селектора `class` является возможность создавать иерархию классов, когда дополнительные классы расширяют общий заданный стиль базового класса. Для создания иерархических классов, сначала описывается базовый класс (при необходимости):

```
.basestyle {
```



```
font-family: Arial;
color: #000000;
font-size: 14px;
}
```

а затем задается дополнительный к нему класс в виде:

```
.basestyle.extended {
    background: #f0f0f0;
}
```

Обратите внимание, что между определением классов `basestyle` и `extended` записана точка без пробелов! Наличие пробела в данном случае приведет к заданию двух разных классов с одним свойством `background`.

Класс `extended` используется только в том случае, если в текущем теге указан класс `basestyle` в атрибуте `class`:

```
<p class="basestyle">Текст со стилем basestyle</p>
<p class="basestyle extended">Текст со стилем
basestyle.extended</p>
<p class="extended">Текст со стилем extended</p>
```

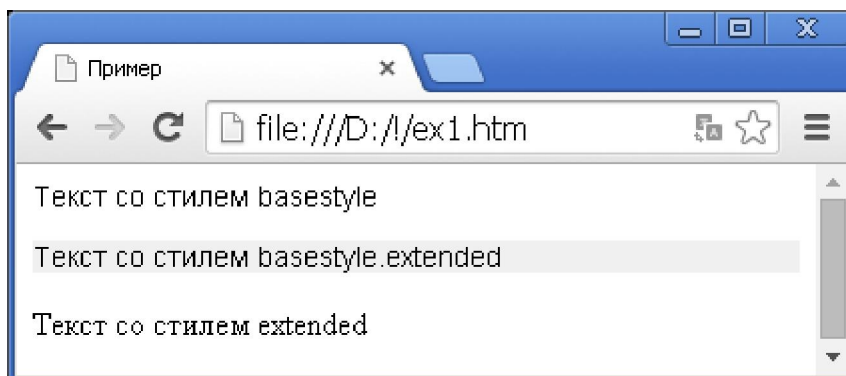


Рис. 4.4. Пример использования иерархических классов в CSS

Из рис. 4.4. видно, что класс `extended` применен только ко второму тегу `<p>` и проигнорирован третьим тегом `<p>`, т.к. в нем отсутствует базовый класс `basestyle`. При этом стоит иметь в виду, что вложенные классы корректно обрабатываются только современными браузерами, а устаревшие будут считать, что это два разных класса и независимо применять их к тегам. Это, в частности, касается браузера Internet Explorer 6.0, поддержка которого давно прекращена и который считается устаревшим.

Используя классы при описании стилей можно задавать контекстные стили по аналогии с ранее рассмотренными контекстными селекторами на базе тегов языка HTML. Например, можно задать стиль

```
.style {
    padding: 5px;
}
```

и определить контекстный селектор для тега ``:

```
.style b {  
    color: green;  
}
```

в результате зеленый цвет шрифта будет присвоен только тем тегам, которые находятся внутри тега с параметром `class="style"`, например,

```
<b>Это не вложенный тег b</b>  
<div class="style">  
    <b>Это вложенный тег b в тег div, имеющий class="style"</b>  
</div>
```

В итоге в окне браузера будет отображена информация в следующем виде (рис. 4.5):

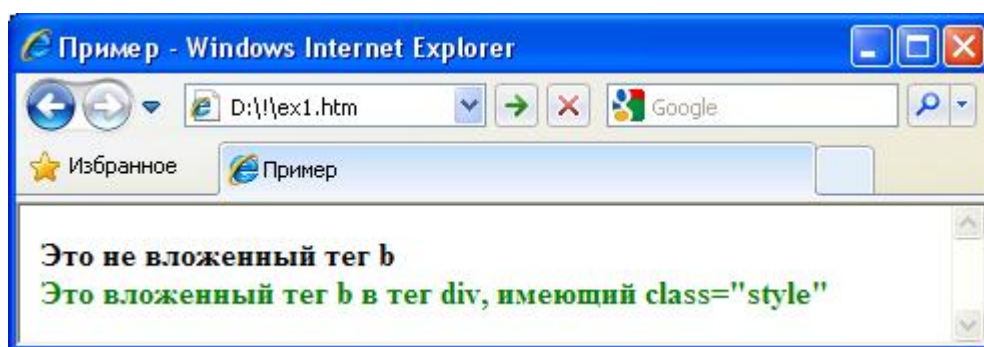


Рис. 4.5. Пример контекстного селектора на базе класса

Подобным образом можно определять и более сложные контексты, например, так:

```
.style p b {  
    color: blue;  
}
```

или так:

```
.style span, p b {  
    color: gray;  
}
```

в этих примерах, для применения стиля к тегу `` необходимо, чтобы он находился внутри тега с параметром `class="style"` и дополнительно был вложен в тег `<p>`, т.е.

```
<div class="style">  
<p><b>Это тег b, вложенный в тег p</b></p>  
<b>Это тег b, не вложенный в тег p</b>  
<span>Это тег span</span>
```

```
</div>
```

В итоге получим, что первая строка будет отображена черным шрифтом, а вторая – синим.

Аналогично можно задавать не только контекстные теги, но и контекстные классы, т.е. когда стиль одного класса будет применен только в том случае, если он вложен в другой, например,

```
.style1 .style2 {  
    background: #a0a0a0;  
}
```

здесь style2 вложен в style1, который становится активным только при вложении в style1 на странице HTML-документа.

4.5. Селектор ID

Селектор ID (от англ. identifier) позволяет задавать стили для тега с соответствующим id. У каждого тега языка HTML имеется атрибут id, задающий его уникальный идентификатор в HTML-странице. То есть, в отличие от атрибута class, значение атрибута id не может повторяться для нескольких тегов в пределах одного документа. Если же это происходит, то браузер, как правило, оставляет один последний неуникальный идентификатор. Следующий пример демонстрирует использование атрибута id внутри тегов:

```
<div id="main">  
    <p id="title">Заголовок</p>  
    <p id="text">Текст</p>  
</div>
```

Обычно идентификаторы служат для манипулирования содержимым тегов, например, для передачи данных серверу, или при обработке содержимого тегов с помощью JavaScript-сценариев. Однако, используя идентификаторы тегов также можно определить для них свойства в таблице стилей. Это выполняется также как и для классов, только перед именем стиля идентификатора ставится символ #:

```
#main {  
    padding: 10px;  
    border: 1px solid #000;  
}  
p#title {  
    font-size: 18px;  
}  
p#text {  
    font-size: 14px;
```

}

Обратите внимание, что перед именем двух последних идентификаторов стоят теги `p`. Это означает, что стиль для идентификатора `title` будет применен браузером только тогда, когда он определен в теге `<p>`. Аналогично и для идентификатора `text`.

В отличие от классов для идентификаторов нельзя создавать иерархии стилей, т.к. значение идентификатора строго определено и не должно изменяться при отображении HTML-страницы. Однако для них можно использовать контекстные селекторы, также как и для классов.

4.6. Выбор первого дочернего элемента

Рассмотренные выше примеры контекстных селекторов позволяют выбирать все дочерние элементы относительно указанного родителя. Вместе с тем на практике часто возникают ситуации, когда необходимо выбрать только первый дочерний элемент. Для этого используется специальный тип селектора, имеющий следующий синтаксис:

```
родитель > потомок
```

Здесь в качестве родителя может выступать любой тег языка HTML, любой класс, идентификатор, контекстные селекторы и т.п. В качестве потомка указывается тег, который требуется выбрать. Следующий пример демонстрирует возможность выбора первого абзаца в теге `div`:

```
div > p {color: #a0a0a0;}
```

В результате на странице

```
<div>  
<p>Это первый абзац</p>  
<p>Это второй абзац</p>  
</div>
```

первая строка будет отображена серым цветом, а вторая – черным.

Селектор выбора первого дочернего элемента часто уточняют с использованием классов, например, так:

```
div.menu > p.main {background: red;}
```

Такой селектор выберет все блоки `div`, у которых `class="menu"` и в них присвоит указанный стиль только первому найденному параграфу с атрибутом `class="main"`.

4.7. Выбор сестринских элементов

В каскадных таблицах стилей существуют селекторы, позволяющие выбирать следующий элемент за текущим, имеющего того же родителя. Такие элементы называются сестринскими. Например, если нужно выделить первый абзац, следующий за заголовком h1, то можно применить такой селектор:

```
h1 + p {padding: 10px; background: f7f7f7;}
```

Такая запись будет указывать браузеру найти на странице все заголовки с тегом <h1> и применить стиль только для первого найденного абзаца <p>, следующего за этим заголовком:

```
<h1>Заголовок 1</h1>  
<p>Первый абзац, следующий за заголовком 1<p>  
<p>Второй абзац, следующий за заголовком 1<p>
```

В итоге первый абзац будет иметь указанный стиль, а второй отобразится со стилем по умолчанию.

Обратите внимание, что в отличие от выбора первого дочернего элемента, сестринские элементы находятся на одном уровне иерархии, т.е. имеют единого родителя. В предыдущем примере заголовков h1 и параграф p имеют единого родителя body. Если это условие не выполняется, то соответствующий сестринский элемент найден не будет.

В CSS допускается комбинировать селекторы разных типов, например, выбор дочерних и сестринских элементов. Следующая запись позволяет выбрать сестринский элемент span у первого дочернего элемента p тега div:

```
div > p + span {color: red}
```

Следует отметить, что селекторы выбора дочерних и сестринских элементов поддерживаются браузером Internet Explorer, начиная с версии 7.0

4.8. Псевдоклассы

С помощью селекторов псевдоклассов можно задавать стили для элементов в зависимости от их текущего состояния в HTML-документе. Проще всего это можно объяснить на примере ссылок. Начиная с самой первой версии, браузеры различали ссылки на посещенные и не посещенные пользователем ресурсы. И в зависимости от их состояния отображали их разным цветом. Например, браузер Internet Explorer по умолчанию отображает ссылки на не посещенные ресурсы синим цветом, а на посещенные – фиолетовым. При этом ссылка на ранее не посещенный ресурс может со временем изменить свое состояние и стать ссылкой на посещенный ресурс. Такое динамическое изменение состояния объекта и описывается с помощью селекторов, называемые псевдоклассами.

Рассмотрим вначале псевдоклассы для манипулирования отображением ссылок в окне браузера (табл. 4.2):

Таблица 4.2. Псевдоклассы для ссылок

Обозначение	Описание
:link	псевдокласс не посещенной ссылки, имеющий атрибут href
:visited	псевдокласс посещенной ссылки
:active	псевдокласс активной ссылки (та которая выбрана, но еще не посещена)
:hover	псевдокласс ссылки, над которой расположен курсор мыши

Теперь зададим стили для отображения ссылки, используя псевдоклассы табл. 4.2:

```
a { text-decoration: none }
a:link {
    color: green;
}
a:visited {
    color: red;
}
a:active {
    color: black;
}
a:hover {
    color: black;
    text-decoration: underline;
}
```

После подключения этих стилей к странице, получим следующий результат отображения ссылки:

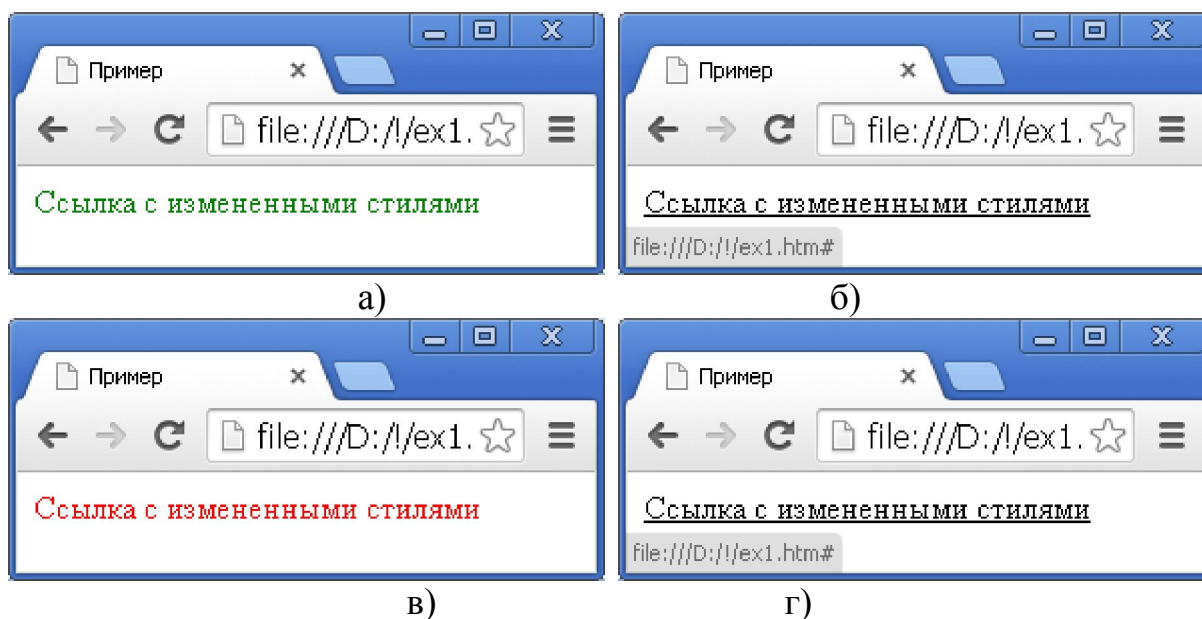


Рис. 4.6. Изменение стиля отображения ссылки с использованием псевдоклассов: а – вид непосещенной ссылки; б – вид ссылки при наведении курсора; в – вид посещенной ссылки; г – вид посещенной ссылки при наведении на нее курсора

Как видно из приведенного примера, стиль ссылки динамически меняется при изменении состояния самой ссылки.

Псевдоклассы можно применять совместно любыми рассмотренными выше селекторами. Например, если требуется определить разные стили для внутренних и внешних ссылок сайта, то это можно легко сделать с использованием классов. Для этого достаточно для внешних ссылок указать некий класс, например, класс с именем `external` и задать стили следующим образом:

```
a.external:link {font-weight: bold;}
a.external:visited {font-weight: bold;}
a.external:active {font-weight: bold;}
a.external:hover {font-weight: bold; text-decoration: underline;}
```

а в HTML-странице написать

```
<p><a href="#">Внутренняя ссылка</a>
<p><a href="http://yandex.ru" class="external">Внешняя ссылка</a>
```

в результате получим (рис. 4.7):

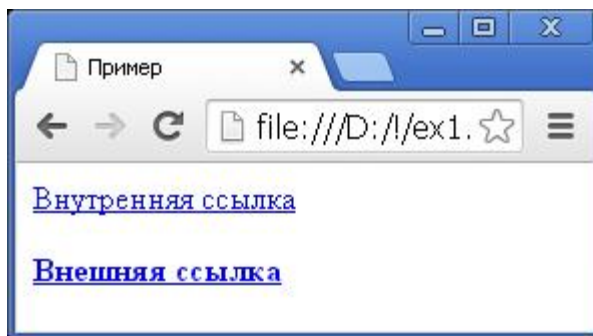


Рис. 4.7. Использование селекторов совместно с псевдоклассами

В CSS2.1 были введены три псевдокласса, позволяющие задавать стиль элемента на события от пользователя. Они также называются динамическими псевдоклассами (табл. 4.3).

Таблица 4.3. Динамические псевдоклассы

Обозначение	Описание
:focus	псевдокласс элемента с активным фокусом (т.е. элемент, выбранный пользователем в настоящий момент)
:active	псевдокласс элемента, активизированный пользователем

:hover	псевдокласс элемента, над которым расположен курсор мыши
--------	--

В приведенных выше примерах уже использовались два динамических псевдокласса: `active` и `hover`. Продемонстрируем работу третьего псевдокласса `focus` на примере элемента ввода текста с клавиатуры:

```
input:focus {background: silver; font-weight: bold;}
```

и определим тег `input` в HTML-странице:

```
<input type="text" value="Поле ввода" />
```

в результате в окне браузера будет отображено (рис. 4.8):

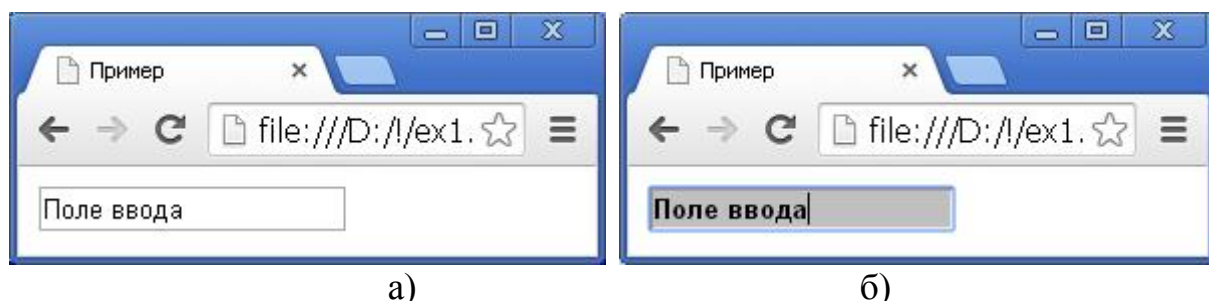


Рис. 4.8. Стиль поля ввода: а – без фокуса; б – с фокусом

При использовании псевдоклассов `focus` и `hover` следует иметь в виду, что они поддерживаются современными браузерами и могут игнорироваться более старыми. Например, Internet Explorer 6.0 не воспринимает псевдокласс `focus`, а псевдокласс `hover` применяет только для ссылок. В IE7 `hover` стал применяться ко всем элементам, но нет поддержки `focus`.

Каскадные таблицы стилей предоставляют еще один полезный псевдокласс для работы с элементами `first-child`, который служит для указания первого дочернего элемента. Ниже представлен пример задание стилей с использованием псевдокласса `first-child`:

```
p:first-child {text-transform: uppercase;}
```

HTML-страница:

```
<h1>Заголовок страницы</h1>
<div>
  <p>Первый абзац страницы</p>
  <p>Второй абзац страницы</p>
</div>
```

результат (рис. 4.9):

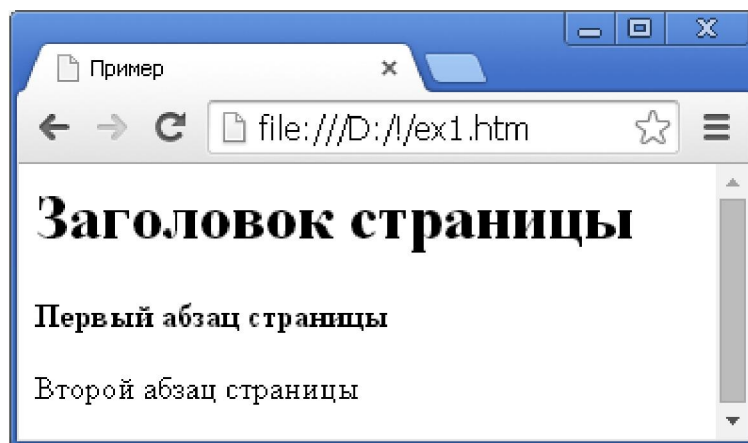


Рис. 4.9. Пример работы псевдокласса first-child

В приведенном примере параграфы заключены в тег `<div>` для того, чтобы на странице существовали дочерние элементы. Если теги `<p>` не помещать внутрь тега `<div>`, то на странице не будет дочерних элементов и соответствующий псевдокласс не будет применен ни к одному тегу.

4.9. Селекторы псевдоэлементов

Псевдоэлементы, в отличие от псевдоклассов, позволяют назначать стили отдельным элементам HTML-страницы. В CSS2.1 определены четыре псевдоэлемента:

- `:first-letter` – первая буква элемента;
- `:first-line` – первая строка в элементе;
- `:before` – стиль перед элементом;
- `:after` – стиль после элемента.

Порядок использования псевдоэлементов такой же как и для псевдоклассов. Ниже приведены примеры использования всех четырех псевдоэлементов.

```
<html>
<head>
<title>Пример</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
<style type="text/css">
h1:first-letter {color: red;}
p:first-line {background: gray;}
</style>
</head>
```

```
<body>
<h1>Заголовок страницы</h1>
<p>Каскадные таблицы стилей позволяют задавать псевдоклассы и
псевдоэлементы для более гибкого назначения стилей при
отображении HTML-страницы в окне браузера.
</body>
```

</html>

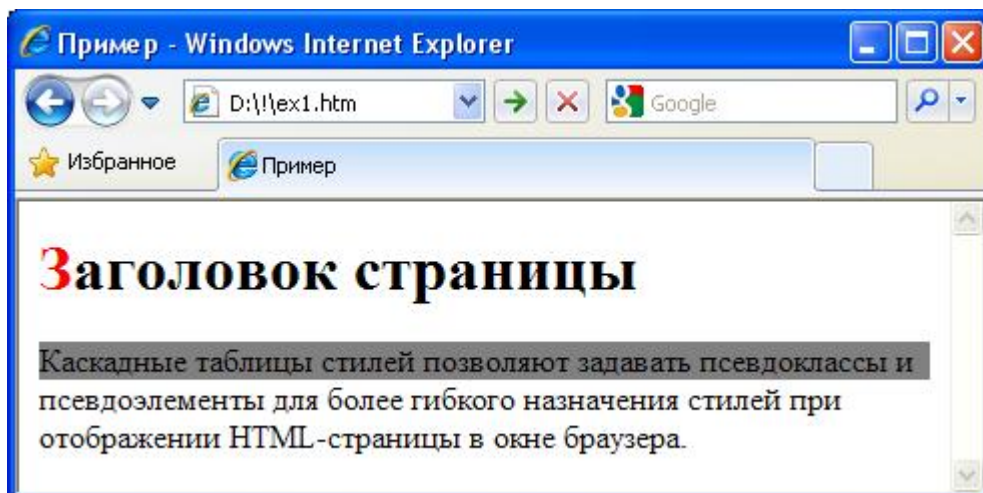


Рис. 4.10. Пример использования псевдоэлементов first-letter и first-line

Следует отметить, что псевдоэлементы first-letter и first-line применяются только к тегам уровня блоков, таких как <p>, <div>, <h1> - <h6> и др., но не могут быть применены к строчным элементам, например, таких как ссылки <a>. Кроме того, при определении в таблице стилей все псевдоэлементы должны размещаться в конце селектора, т.е. запись вида p:first-letter b неверна, т.к. за псевдоэлементом следует тег .

Вторая пара псевдоэлементов before и after позволяет добавлять содержимое до и после элемента, например,

```
h2:before {content: '['; color: gray;}  
h2:after {content: ']'; color: gray;}
```

будет заключать все заголовки уровня h2 в квадратные скобки:

```
<h2>Заголовок страницы</h2>
```

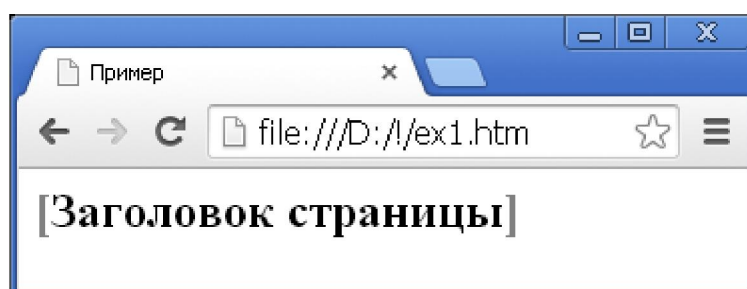


Рис. 4.11. Пример использования псевдоэлементов before и after

Последние два псевдоэлемента воспринимаются только современными браузерами. Например, браузер IE6 не работает с псевдоэлементами before и after и будет их игнорировать при отображении страницы.

ПРИМЕРЫ РАЗМЕТОК СТРАНИЦ С ИСПОЛЬЗОВАНИЕМ CSS

В данной главе рассмотрим примеры разметок страниц, примененные в реальных сайтах сети Интернет. Это позволит объединить вышеизложенную информацию для лучшего ее понимания и усвоения. Кроме того, при рассмотрении каскадных таблиц стилей, читатель сможет познакомиться с наиболее распространенными свойствами CSS и увидит способы их определения.

Для лучшего понимания, при описании структуры страниц, будут представлены общие правила их форматирования на уровне HTML и CSS, несущественные детали, относящиеся к особенностям построения и функционирования конкретного сайта, будут опущены.

5.1. Разметка страниц сайта <http://selrecipes.ru>

Сайт «Избранные кулинарные рецепты» представляет собой шаблон страниц с фиксированной шириной, расположенных по центру окна браузера. На фоне отображен рисунок, размером 1920x1000 пикселей, сохраненный в формате JPEG. Фоновый рисунок имеет фиксированное положение, поэтому, когда пользователь прокручивает содержимое страницы, рисунок остается на месте. Это создает красивый эффект фона и подчеркивает прозрачность некоторых элементов дизайна сайта.

Первой строкой в HTML-странице этого сайта идет тег `<!doctype>`, указывающий браузеру правила интерпретации тегов и их свойств в документе. Например, запись

```
<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
```

означает интерпретацию тегов в соответствии со спецификацией HTML 4.01. Эта строка необходима для верного представления страницы в окне браузера.

Следующие теги указывают на начало описания HTML-документа и определяют раздел `head` следующим образом:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link rel="shortcut icon" href="http://selrecipes.ru/images/fav.ico"
type="image/x-icon"/>
<link type="text/css" href="http://selrecipes.ru/css/styles.css"
rel="stylesheet" />
<title>Избранные кулинарные рецепты</title>

<base href="http://selrecipes.ru/" />
</head>
```

Здесь в первой строке раздела `head` задается кодировка страницы UTF-8. Во второй строке определяется фавикон страниц сайта (изображение, обычно 16x16 пикселей, отображаемое в закладке окна браузера перед именем страницы). Затем подключается таблица стилей `style.css`, в разделе `title` задается название страницы «Избранные кулинарные рецепты», и, наконец, определяется базовый адрес сайта в теге `<base>`.

Непосредственно разметка страницы описывается в разделе `body`, который определен в данном сайте как

```
<body class="body_style">
...
</body>
```

где `body_style` – класс из таблицы стилей `style.css`, имеющий вид:

```
.body_style {
    min-width: 1100px;
    min-height: 900px;
    height: 100%;
    margin: 0px;
    background: #fff url('../images/page.jpg') no-repeat 50% 0
fixed;
}
```

Свойства **`min-width`** и **`min-height`** определяют минимальную ширину и высоту страницы сайта в окне браузера. То есть, если окно браузера будет меньше этих значений, то появятся полосы прокрутки, сохраняя указанные минимальные значения. Значение **`height: 100%`** указывает браузеру создавать страницу на всю высоту окна, даже если ее содержимое отсутствует. Это нужно, например, для отображения информации по центру окна браузера. Если фактическая высота страницы не будет соответствовать высоте окна, то элемент не будет отцентрирован по высоте. Свойство **`margin`** позволяет задавать смещения блоковых элементов на HTML-страницы. По умолчанию раздел `body` имеет небольшие отступы слева и сверху страницы и если их не приравнять нулю, то изображение «шапки» сайта будет отображено с этими отступами, что в данном случае не соответствует дизайну страницы. Поэтому свойство `margin` было установлено равным 0. Наконец, последнее свойство

```
background: #fff url('../images/page.jpg') no-repeat 50% 0 fixed;
```

задает фон страниц сайта в виде изображения, находящегося по адресу:

<http://selrecipes.ru/images/page.jpg>

которое не должно дублироваться (повторяться) на странице по-`repeat`, быть выровненным по центру (50%), отображаться сверху (0) и не прокручиваться вместе с содержимым документа (`fixed`).

Непосредственно разметка страницы задается с помощью таблицы, которая записана в виде:

```
<table border=0 cellspacing="0" cellpadding="0" style="width:1100px;height:100%;" align=center>
<tr><td valign=top class="header">
    <div class="header_text">Избранные рецепты</div>
    <div class="header_text2">простые и вкусные блюда</div>
</td></tr>
<tr><td class="shadow"></td>
</tr>
<tr><td>
Содержимое страницы
</td></tr>
<tr><td valign=top style="height:60px;">
<div class="footer">
    <p>&copy; SelRecipes.Ru 2013
</div>
</td></tr>
</table>
```

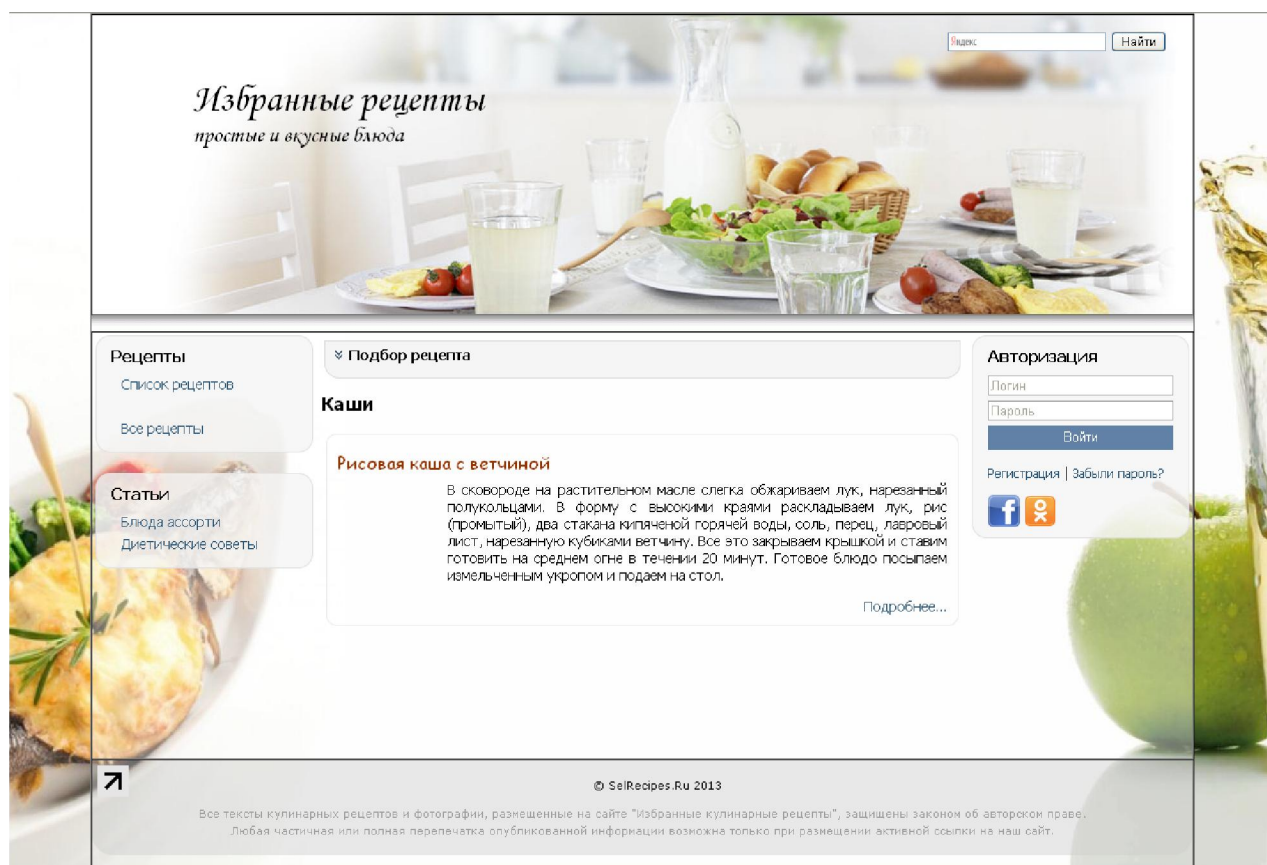


Рис. 5.1. Разметка страницы сайта таблицей (здесь временно был установлен параметр таблицы border=1)

Из рис. 5.1. видно, что таблица позиционируется по центру окна браузера и содержит три ячейки, расположенных вертикально. Самая верхняя ячейка содержит «шапку» сайта, в которой отображен логотип, текст «Избранные

рецепты» и «простые и вкусные блюда», а также поле для поиска по сайту. Во второй ячейке находится непосредственно содержимое страницы: меню, список и описания рецептов, окно авторизации. Последняя самая нижняя ячейка служит для отображения «подвала» (от англ. footer) сайта. В нем обычно размещают знак соруригит с именем сайта и текущим годом. Также может быть любой дополнительный текст общий для всех страниц. В данном случае идет предупреждение об авторских правах.

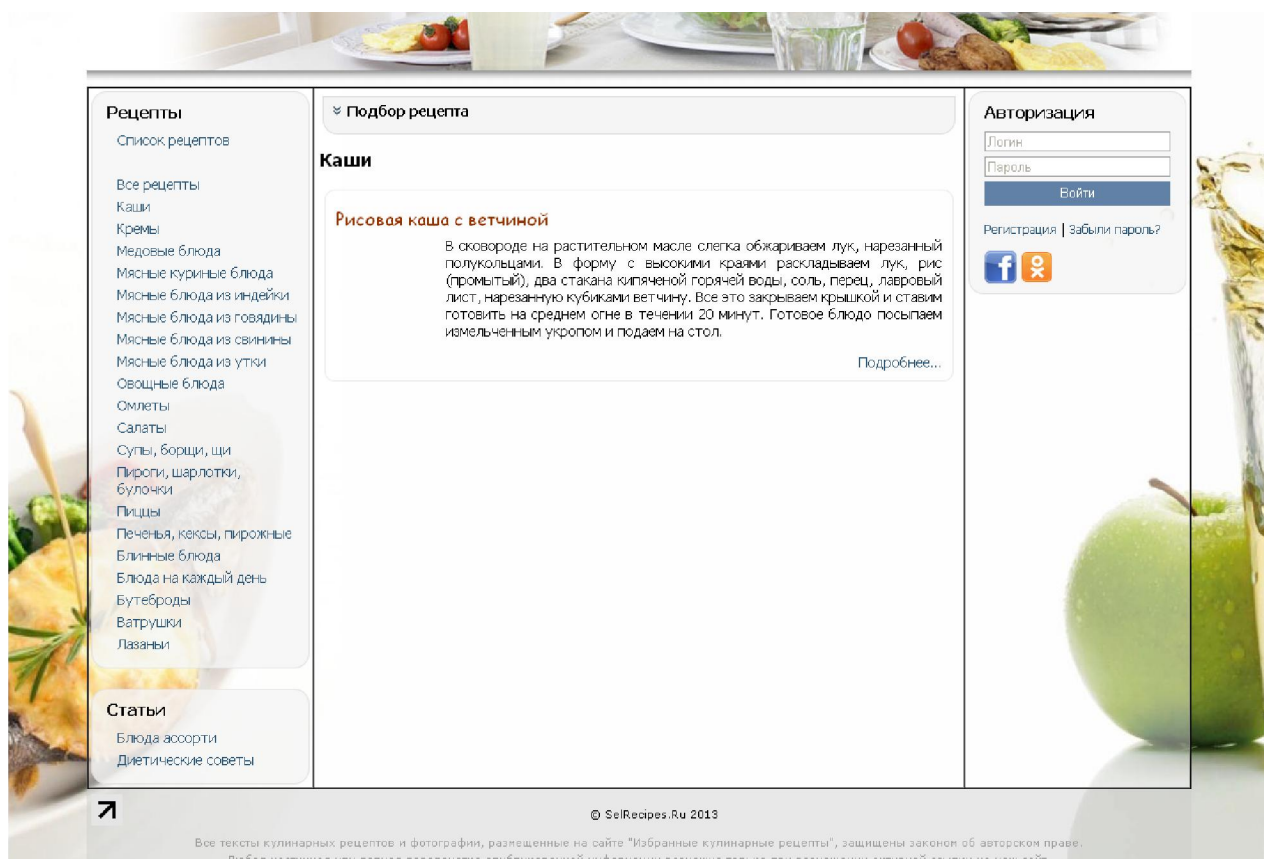


Рис. 5.2. Разметка основной части страницы (здесь временно был установлен параметр таблицы border=1)

Ячейка, в которой находится основное содержание страницы сайта, дополнительно разбита еще на 3 ячейки, расположенных в горизонтальном порядке (рис. 5.2). Они были образованы с помощью вложенной таблицы в соответствующую ячейку первой таблицы. Вторая таблица имеет вид:

```
<table border=0 cellspacing="0" cellpadding="0" class="main-page">
<tr><td valign=top>
Меню сайта
</td><td valign=top class="content">
Содержимое страницы сайта
</td><td valign=top>
Форма авторизации
</td></tr>
</table>
```

соответствующие стили описаны следующим образом:

```
.main-page {
    background: #a0a0a0;
    background: rgba(255, 255, 255, 0.4);
    height: 100%;
}

.content {
    padding: 5px;
    width: 100%;
    font-family: Tahoma, Arial;
    font-size: 14px;
}

.content table {
    font-family: Tahoma, Arial;
    font-size: 14px;
}

.content h2 {
    font-size: 18px;
}
```

Стиль таблицы **main-page** содержит два определения свойства **background**. Это сделано для отображения желаемого фона в браузерах разных версий. Дело в том, что значение **rgba** поддерживается только современными браузерами и игнорируется старыми. Поэтому для прежних версий свойство **background** описано как **#a0a0a0**, а для более современных браузеров оно будет заменено значением **rgba(255, 255, 255, 0.4)**. Такой прием часто используется при описании стилей под разные версии браузеров. Свойство **height** со значением **100%** соответственно вытягивает страницу на всю высоту ячейки таблицы, в которую она вложена. В результате в современных браузерах будет отображена таблица с белым полупрозрачным фоном, а в устаревших – с серым непрозрачным.

Стиль **content** средней ячейки содержит свойство **padding:5px** задает отступ от границ внутри блоковых элементов (в данном случае внутри элемента ячейки таблицы **td**). Свойство **width:100%** устанавливает ширину средней ячейки на максимально возможное значение (в данном случае на всю ширину страницы за вычетом ширины содержимого первой и третьей ячеек). Свойства **font-family: Tahoma, Arial** и **font-size: 14px** определяют параметры шрифта для отображения текста на странице.

На основе класса **content** образовано два контекстных селектора: **.content table** и **.content h2**. Они служат для задания свойств таблицы и заголовка второго уровня внутри блока с классом **content**, т.е. внутри средней ячейки второй таблицы. Селекторы имеют наборы свойств, которые уже были описаны выше.

Описанные таблицы и их свойства образуют общую разметку страниц сайта <http://selrecipes.ru> таким образом, чтобы они корректно отображались в

браузерах с разным размером окна и разными версиями. При этом содержимое основной страницы может быть сколь угодно большим, оно будет корректно отображено в окне браузера в пределах дизайна страницы.

5.2. Разметка страниц сайта <http://math.sernam.ru>

Сайт <http://math.sernam.ru> служит для выполнения вычислений в дробях и состоит из трех страниц. Общий дизайн всех трех страниц подобен друг другу, поэтому рассмотрим подробно структуру первой (главной) страницы этого сайта.

В первой строке каждой страницы идет объявление:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

для интерпретации тегов в соответствии со спецификацией XHTML 1.0. Эта строка необходима для верного представления страницы в окне браузера.

В самом верху страницы расположено меню: «Научная библиотека», «Клуб читателей», «Вычисления в дробях», «Информационный ассистент» и справа указан e-mail: sc-lib@list.ru. Ниже идет непосредственно содержание страницы. Она разделена на две части: слева располагается меню: «Калькулятор», «Матрицы», «Тригонометрия», а справа – заголовок страницы «Вычисления в дробях» и строка ввода формулы с кнопкой «Вычислить» (рис. 5.3).

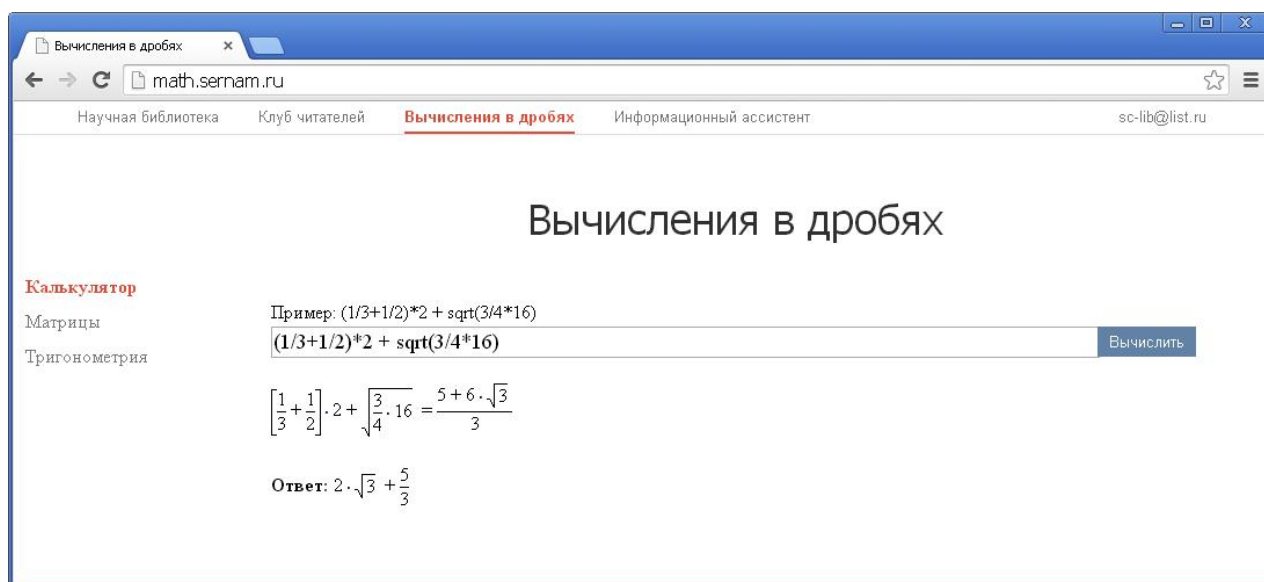


Рис. 5.3. Вид первой страницы сайта math.sernam.ru

Для представления верхнего меню сайта был использован блок `div`, в который вложены пять других блоков `<div>`, таким образом, чтобы они образовывали элементы меню:


```

<div class="topmenu">
  <div class="topmenu-item"><a href="http://sernam.ru">Научная
библиотека</a></div>
  <div class="topmenu-item"><a href="http://reeders.ru">Клуб
читателей</a></div>
  <div class="topmenu-item topmenu-active">Вычисления в
дробях</div>
  <div class="topmenu-item"><a href="http://abcn.info"
target="_blank">Информационный ассистент</a></div>
  <div class="topmenu-item last"><a href="mailto:sc-
lib@list.ru">sc-lib@list.ru</a></div>
</div>
<div style="clear: both"></div>

```

Класс `topmenu` определяет стиль всего верхнего меню и имеет вид:

```

.topmenu {
  font: normal normal 400 13px/normal Arial, Tahoma, sans-
serif;
  min-width: 1000px;
  border-bottom: 1px solid #e0e0e0;
  background: #fff;
  padding: 0 40px 0 40px;
}

```

Здесь свойство *font* определяет размер и имя шрифта пунктов меню, свойство *min-width: 1000px* задает минимальную ширину верхнего меню, *border-bottom: 1px solid #e0e0e0;* указывает браузеру нарисовать линию в нижней части меню шириной 1 пиксел, непрерывную (solid) с цветом #e0e0e0. Свойство *background: #fff;* определяет белый цвет фона меню, и свойство *padding: 0 40px 0 40px;* определяет отступ в 40 пикселей от левой и правой границ блока `div` и 0 пикселей сверху и снизу блока.

Так как пункты меню представляют собой ссылки на другие ресурсы, то дополнительно для них задаются контекстные селекторы:

```

.topmenu a {
  color: #777;
  text-decoration: none;
}
.topmenu a:hover {
  color: #000;
}

```

Первый селектор определяет цвет и оформление ссылки при ее отображении в окне браузера внутри класса `topmenu`. Второй селектор использует псевдокласс `:hover` для изменения цвета ссылки на черный, при наведении на нее курсора.

Для описания стилей пунктов меню используются классы `topmenu-item` и `topmenu-active`, имеющие вид:

```

.topmenu-item {
    display: inline-block;
    padding: 5px 0 5px 0;
    margin: 0 15px 0 15px;
}
.topmenu-item.last {
    float: right;
}

.topmenu-active {
    color: #dd4b39;
    font-weight: bold;
    cursor: pointer;
    border-bottom: 2px solid #dd4b39;
}

```

Первое свойство класса `topmenu-item` ***display: inline-block*** указывает браузеру отображать элемент `div` не как блочный (`block`), а как строчный элемент (`inline-block`). Это свойство CSS позволяет блочные элементы превращать в строчные элементы и является полезной особенностью каскадных таблиц стилей. Второе свойство ***padding: 5px 0 5px 0;*** задает отступы в 5 пикселей сверху и снизу пункта меню. Это сделано для того, чтобы пункт меню не сливался с верхней границей окна браузера, а внизу – с разделительной линией меню. Третье свойство ***margin: 0 15px 0 15px;*** смещает элемент пункта меню на 15 пикселей справа и слева так, чтобы элементы меню визуально были отделены друг от друга. Обратите внимание на отличие свойств `padding` и `margin`: `padding` добавляет пустое пространство внутрь блока, а `margin` смещает сам блок на странице.

Составной класс **`.topmenu-item.last`** используется для отображения последнего элемента в меню, в данном случае для отображения ссылки на email. Свойство ***float: right;*** указывает браузеру расположить данный пункт от правой границы верхнего меню, которая определяется первым элементом `<div class="topmenu">`. В результате ссылки на другие ресурсы и ссылка на email визуально четко разделены между собой (рис. 5.3).

Последний класс **`.topmenu-active`** задает стиль для выбранного пункта меню. Он отображается красным цветом ***color: #dd4b39***, полужирным ***font-weight: bold***, вид курсора в виде указателя ***cursor: pointer*** и снизу подчеркивается красной чертой толщиной 2 пиксела ***border-bottom: 2px solid #dd4b39*** (рис. 5.3).

Основное содержание страницы описывается в виде блока меню, стоящего слева и таблицы, находящейся по центру оставшейся ширины страницы. В терминах HTML-тегов это записано следующим образом:

```

<!-- меню слева на странице -->

<div class="menu-bar">
<ul class="menu-bar-list">

```

```

    <li class="selected">Калькулятор</li>
    <li><a href="matrix.php">Матрицы</a></li>
    <li><a href="trigonom_index.php">Тригонометрия</a></li>
</ul>
</div>

```

<!--таблица с основным содержимым страницы -->

```

<table border="0" cellspacing="0" cellpadding="0" class="calc-
main" align=center>
<tr><td>
    <div class="calc-title">Вычисления в дробях</div>
    Строка ввода с кнопкой
</td></tr>
<tr><td valign=top>
    <div class="calc-output">
        <div id="loading-status" style="display:none;"></div>
        <div id="solution"></div>
    </div>
</td></tr>
</table>

```

Блок меню создается с помощью блочного элемента `<div>` со стилями, заданными в классе `menu-bar`:

```

.menu-bar {
    float: left;
    margin-top: 100px;
    padding-left: 10px;
    font: normal normal 400 16px/normal Times New Roman, Arial;
}

.menu-bar a {
    color: #777;
    text-decoration: none;
}

.menu-bar a:hover {
    color: #000;
}

```

Обратите внимание, что первым свойством класса `menu-bar` идет ***float: left***. Оно необходимо, чтобы блок меню располагался слева, а таблица «обтекала» его справа. Если это свойство убрать, то блок меню также будет расположен слева, но таблица будет следовать на следующей строке за ним, т.е. под меню, что не будет согласоваться с дизайном сайта. Свойства ***margin-top: 100px*** и ***padding-left: 10px*** задают смещение сверху на 100 пикселей и определяют пустое пространство слева в 10 пикселей. Последнее свойство ***font*** определяет размер и название шрифта элементов меню. Определение свойств для тега `<a>` должно быть уже понятно из предыдущего изложения.

Непосредственно элементы меню выполнены в виде маркированного списка `` со следующими стилями:

```
.menu-bar-list {
    list-style: none;
    padding: 0px;
}
.menu-bar-list li {
    min-width: 150px;
    padding: 5px 0 5px 0;
}
.menu-bar-list .selected {
    color: #DD4B39;
    font-weight: bold;
}
```

Сначала задаются общие стили для всего списка в классе **menu-bar-list**. Свойство **list-style: none** означает отсутствие маркеров в списке, а свойство **padding: 0px** удаление каких-либо специальных смещений для элементов списка (по умолчанию присутствует смещение элементов списка слева).

Для определения свойств непосредственно элементов списка, заданных в теге ``, создается контекстный селектор **.menu-bar-list li**, в котором определяется **min-width: 150px** – минимальная ширина элементов меню, и отступы сверху и снизу **padding: 5px 0 5px 0**, чтобы визуально разделить пункты меню. Наконец, определен контекстный селектор в виде класса **.selected**, в котором задаются свойства для выбранного пункта меню: **color: #DD4B39** – красный цвет и **font-weight: bold** – полужирный шрифт.

Для представления основного содержимого страницы используется таблица, состоящая из двух ячеек, расположенных вертикально друг за другом (рис. 5.4) и центрированной по горизонтали с помощью атрибута **align=center**.

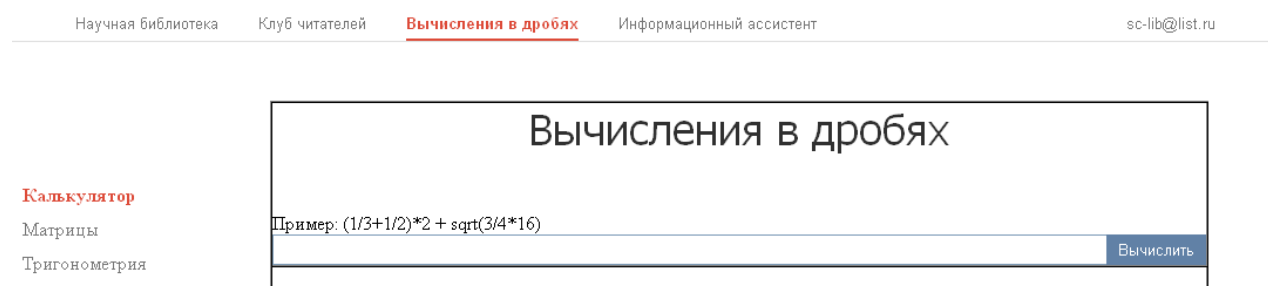


Рис. 5.4. Расположение таблицы для разметки основного содержимого страницы (здесь временно установлен атрибут `border=1` для отображения границ таблицы)

В первой (верхней) ячейки находится название страницы «Вычисления в дробях» со стилями, заданными в классе **calc-title**:

```
.calc-title {
    text-align: center;
```

```
font: normal normal 400 36px/normal Tahoma, Arial;
padding-bottom: 50px;
color: #303030;
}
```

а во второй ячейке записаны два блока `<div>`: в первом блоке отображается анимированное изображение процесса вычисления и загрузки решения (изначально оно скрыто `style="display: none;"`), а во втором – полученное решение при вычислении указанных дробей. Второй блок всегда отображается на экране, но т.к. изначально в нем ничего нет, то создается впечатление, что он как будто отсутствует. При нажатии на кнопку «Вычислить» происходит процесс вычисления, с помощью JavaScript решение записывается во второй блок `<div>` и оно отображается на экране.

5.3. Разметка страниц сайта <http://reeders.ru>

Сайт «Клуб читателей» имеет некоторые общие стили для описания изображений и ссылок:

```
img {border:0px;}

a:link {
    color: #2B587A;
    text-decoration: none;
}
a:visited {
    text-decoration: none;
    color: #2B587A;
}
a:hover {
    text-decoration: underline;
    color: #2B587A;
}
a:active {
    text-decoration: none;
    color: #2B587A;
}
```

Эти стили определены в файле `style.css` и подключаются через тег `<link>` в разделе `<head>`.

В первой строке каждой страницы идет объявление:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

для интерпретации тегов в соответствии со спецификацией XHTML 1.0. Эта строка необходима для верного представления страницы в окне браузера.

Для разметки страниц сайта <http://reeders.ru> также как и в предыдущих примерах использовалась таблица (рис. 5.5). Чтобы таблица плотно прилегала к границам окна, для тега `<body>` были установлены стили, записанные в классе `body_style`:

```
.body_style {  
    min-width: 900px;  
    min-height: 900px;  
    height: 100%;  
    margin: 0px;  
}
```

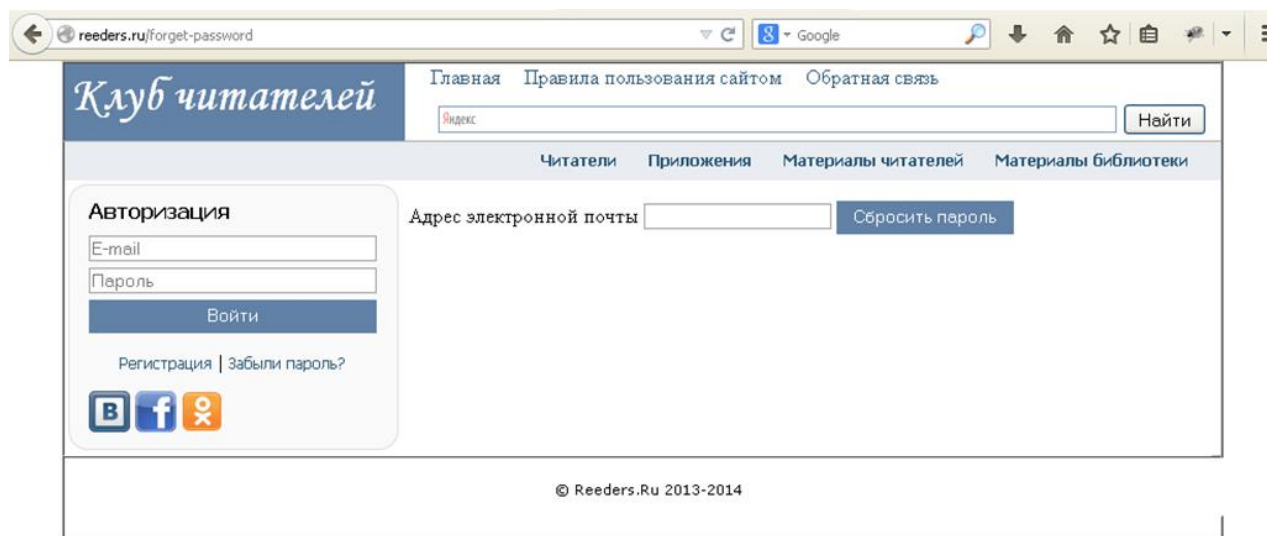


Рис. 5.5. Разметка страницы сайта <http://reeders.ru> (здесь временно установлен атрибут `border=1` для отображения границ таблицы)

Таблица представляет собой три ячейки, расположенные вертикально друг за другом. Верхняя ячейка содержит «шапку» сайта, вторая – основное содержание страницы, а третья – «подвал» (footer) сайта:

```
<table id="site-table" border=0 cellspacing="0" cellpadding="0"  
style="width:900px;height:100%;" align=center>  
<tr><td valign=top align=left>  
    <div class="header">  
        <div class="logo">Клуб читателей</div>  
        <div class="logo_line">  
            <a href="http://reeders.ru">Главная</a>  
            <a href="terms">Правила пользования сайтом</a>  
            <a href="mailto:reeders@bk.ru">Обратная связь</a>  
        </div>  
        <div class="logo_line">  
            <input type="text" class="top_search"/>  
            <input type="submit" value="Найти" />  
        </div>  
    </div>  
</td></tr>
```

```

<tr><td align=left>
Основное содержание страницы
</td></tr>
<tr><td valign=top style="height:40px;">
<div class="footer">
    <p>&copy; Reeders.Ru 2013-2014
</div>
<div class="separator"></div>
</td></tr>

</table>

```

Для создания разметки «шапки» сайта используется блоковый тег-контейнер `<div>` с классом стилей `header`

```

.header {
    width: 900px;
    height: 60px;
}

```

который задает ширину и высоту «шапки» сайта. Внутри этого блока находятся три блока `<div>`, описывающие детали «шапки». Первый вложенный блок

```
<div class="logo">Клуб читателей</div>
```

отображает надпись «Клуб читателей» белыми буквами на синем фоне:

```

.logo {
    width:260px;
    height:55px;
    background: #6181A6;
    padding: 0 0 5px 5px;
    margin-right: 20px;
    font-family: Monotype Corsiva, Tahoma, Arial;
    font-size: 38px;
    color: #fff;
    float:left;
}

```

Свойства класса **logo** определяют ширину и высоту блока логотипа, цвет фона, смещение и размер надписи в этом блоке, цвет шрифта и положение блока `div` слева относительно родительского блока `<div class="header">`. Последнее свойство ***float:left*** необходимо, чтобы вложенный блок `<div>` располагался слева, а последующие блоки `<div>` его обтекали справа. Если это свойство убрать, то по умолчанию блоки `<div>` располагаются друг за другом по вертикали, что не будет соответствовать дизайну сайта.

Следующие два вложенных блока

```

<div class="logo_line">
  <a href="http://reeders.ru">Главная</a>
  <a href="terms">Правила пользования сайтом</a>
  <a href="mailto:reeders@bk.ru">Обратная связь</a>
</div>
<div class="logo_line">
  <input type="text" class="top_search"/>
  <input type="submit" value="Найти" />
</div>

```

образуют две горизонтальные линии справа от логотипа (первого блока div) благодаря стилям класса `logo_line`:

```

.logo_line {
  height:25px;
  float: left;
  width: 610px;
}
.logo_line a {
  padding-right: 15px;
}

```

Свойства `height` и `width` подобраны таким образом, чтобы блоки `div` занимали всю ширину и половину высоты блока логотипа. Свойство ***float:left*** обеспечивает расположение этих блоков друг за другом по горизонтали, но т.к. их ширина не позволяет им располагаться на одной линии, то они выравниваются по вертикали, образуя две линии общей высотой, равной высоте блока логотипа. В результате получается заголовок сайта, показанный на рис. 5.6.

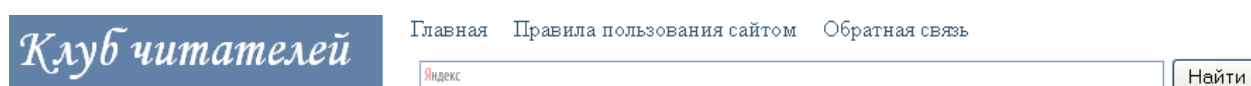


Рис. 5.6. Вид «шапки» сайта <http://reeders.ru>

Для разметки основного содержания страницы используется таблица, вложенная во вторую ячейку первой таблицы. Вложенная таблица имеет две ячейки, расположенные горизонтально друг за другом (рис. 5.7). Содержимое каждой ячейки выравнивается по верхнему краю, т.е. ячейки `<td>` имеют атрибут `valign=top`. В первой ячейке располагается форма авторизации, а после авторизации – профиль пользователя. Во второй ячейке – содержание страницы. В результате получается разметка, универсальная для всех страниц сайта <http://reeders.ru>.

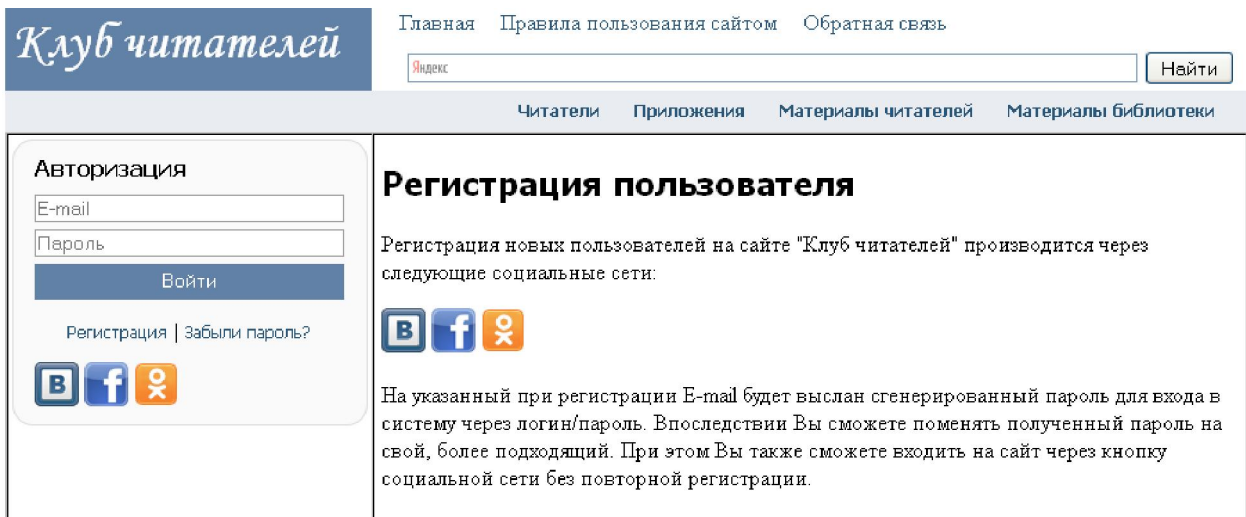


Рис. 5.7. Разметка основного содержания страницы (здесь временно установлен атрибут border=1 для отображения границ таблицы)

ФОРМЫ В HTML

Создание стандарта сети Интернет Web 2.0 позволило пользователю не только принимать информацию, но и активно взаимодействовать с другими пользователями и интернет-сервисами. Для организации такой обратной связи в язык HTML были введены дополнительные теги, позволяющие отправлять на сервер запрашиваемую информацию. Например, это может быть форма регистрации, или форма добавления комментария, или настройка личного аккаунта (страницы социальной сети) и т.п. В данной главе рассмотрим набор тегов, позволяющих организовывать взаимодействие пользователя с сайтом.

6.1. GET и POST-запросы

Для организации взаимодействия пользователя с сетью Интернет, разработчик сайта должен предусмотреть передачу запросов от пользователя сайта на сервер, где расположен данный сайт. Запросы бывают двух видов: GET- и POST-запросы.

GET-запросы

На раннем этапе развития сети Интернет существовали только GET-запросы. Они представляют собой передачу данных непосредственно в адресной строке браузера, имеющих следующий синтаксис:

```
http://домен/страница?[параметр1=значение1][&параметр2=значение2]...
```

Здесь набор передаваемых данных на сервер начинаются с символа '?' и разделяются символом '&'. Сами данные представляют собой пары

```
параметр=значение
```

Например, если требуется передать имя и фамилию пользователя на странице регистрации (например, register.php) сайта mysite.com, то это будет выглядеть так:

```
http://mysite.com/register.php?fname=Иван&lname=Иванов
```

Следует обратить внимание, что браузеры устаревших версий могут некорректно воспринимать кириллицу и передача русских букв будет осуществляться неверно. Лучше в GET-запросах передавать исключительно служебную информацию в виде чисел и слов на латинском языке.

Недостатком GET-запросов является ограниченность передаваемых данных. На стороне сервера строка запроса ограничивается некоторым

максимальным значением. Например, если максимальный размер запроса может составлять 1024 символа, то все что превышает это значение будет удалено и тогда часть передаваемой информации не будет обработана указанной страницей сайта. Вторым существенным ограничением является возможность передачи строго определенных наборов символов. Например, символы ? и & уже зарезервированы и их передавать как значения параметров нельзя. Однако это правило можно обойти, если в строке запроса передавать не сам символ, а его кодовое значение. Для этого используется символ '%', за которым следует код символа, например, так:

```
http://mysite.com/register.php?fname=%CC%DF%AD%1F%DS&lname=%DD
```

Здесь кодовые значения указаны в шестнадцатиричном виде с целью экономии длины запроса.

Несмотря на указанные недостатки, создание сайтов без GET-запросов было бы крайне затруднительно. Например, они незаменимы в случаях начальной инициализации страницы сайта для конкретного пользователя, когда в запросе указывается не только сайт и текущая страница, но и его id, как это сделано в социальной сети Вконтакте:

```
http://vk.com/profile.php?id=12345678
```

Также GET-запросы часто применяются для проверки корректности email-адреса при регистрации пользователя. В этом случае пользователю на указанный email приходит письмо со ссылкой активации и эта ссылка представляет собой GET-запрос.

POST-запросы

Для решения указанных недостатков GET-запросов были добавлены POST-запросы, позволяющие передавать большие объемы данных в бинарном виде, т.е. без искажений и изменений передаваемых данных. Такие запросы хорошо подходят для загрузки файлов и изображений на сервер. Например, когда пользователь загружает изображение в свой профиль социальной сети, то для этого используются POST-запросы. Более подробно об организации POST-запросов будет сказано ниже.

6.2. Создание HTML-форм

Чтобы позволить пользователю взаимодействовать с сайтом используют специальные формы, в которых вводится необходимая информация, а затем она передается серверу для обработки. Для создания таких форм используется тег-контейнер <form>, имеющий следующие основные необязательные атрибуты:

`action` – адрес скрипта, обрабатывающий передаваемые данные;
`method` – метод протокола HTTP (определяет тип запроса GET или POST).

Например, чтобы отправить GET-запрос сайту `http://mysite.ru` скрипту `getform.php` следует записать тег `<form>` в виде:

```
<form action="http://mysite.ru/getform.php" >  
<!-- поля ввода данных -->  
</form>
```

или так:

```
<form action="getform.php" >  
<!-- поля ввода данных -->  
</form>
```

Второй вариант записи подразумевает, что браузер достроит относительный адрес до абсолютного в виде `http://mysite.ru/getform.php`. Это будет корректно работать, например, когда страница загружена с этого же сайта `http://mysite.ru` или установлен тег `<baseurl url="http://mysite.ru">` (о теге `baseurl` читай выше).

Обратите внимание, если атрибут `method` не задан, то форма будет передавать GET-запрос. Если же нужно изменить способ передачи информации с GET на POST, то это следует явно указывать в атрибуте `method`:

```
<form action="getform.php" method="POST" >  
<!-- поля ввода данных -->  
</form>
```

Внутри тега `<form>` располагаются поля ввода информации. Чаще всего они создаются с помощью тегов `<input>`, `<select>` и `<textarea>`. Все перечисленные теги имеют необязательные атрибуты `class`, `style` и `id` для их связи с таблицами каскадных стилей, позволяющие создавать требуемое оформление элементов управления.

Тег `<input>`

Тег `<input>` отвечает за создание следующих элементов (табл. 6.1).

Таблица 6.1. Элементы, создаваемые с помощью тега `<input>`

Название элемента	Вид элемента	Тип
Поле ввода	<input type="text"/>	text
Кнопка	<input type="button" value="Кнопка"/>	button
Флажки	<input checked="" type="checkbox"/> Флажок	checkbox
Переключатели	<input type="radio"/> Пиво <input type="radio"/> Чай <input checked="" type="radio"/> Кофе	radio
Поле ввода пароля	<input type="password"/>	password

Скрытое поле		hidden
Сброс данных формы	<input type="button" value="Сбросить"/>	reset
Отправка данных серверу	<input type="button" value="Отправить"/>	submit

Например, для создания поля ввода в простейшем случае используется запись:

```
<input type="text" />
```

Если необходимо задать значение по умолчанию, которое будет изначально отображаться в поле ввода, то используется атрибут value:

```
<input type="text" value="Начальное значение" />
```

Если нужно изменить размер отображения поля ввода по горизонтали, то используется атрибут size:

```
<input type="text" value="Начальное значение" size=10 />
```

Наконец, чтобы указать имя параметра, который будет ассоциирован с введенным значением в поле ввода при передаче информации, следует использовать атрибут name:

```
<input type="text" name="fio" value="Значение" size=10 />
```

Обратите внимание, значение атрибута name должно быть уникальным в пределах одной формы, иначе возникнет неопределенность в сопоставлении имен и соответствующих данных формы.

Аналогичным образом создаются все вышеуказанные в табл. 6.1 элементы управления:

```
<input type="checkbox" checked />Флажок
```

Атрибут checked устанавливает флажок в состояние «отмечено».

```
<input type="radio" name="sex" checked>Мужской
<input type="radio" name="sex" >Женский
<br>
<input type="radio" name="drink" checked>Чай
<input type="radio" name="drink" >Кофе
```

Пример двух переключателей в одной форме: первый служит для выбора пола, второй – для выбора напитка. Переключатели группируются по атрибуту name, т.е. поля radio, имеющие одно и тоже значение атрибута name, определяют одну группу переключателей.

Для создания кнопки отправки данных формы серверу, используется специальный тип кнопки submit:

```
<input type="submit" value="Отправить" />
```

Тег <select>

С помощью тега <select> создаются списки на HTML-страницы. Они могут быть выпадающими и не выпадающими (рис. 6.1).

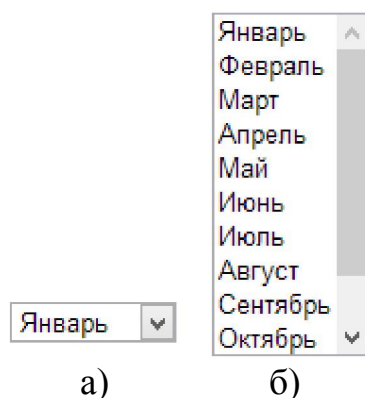


Рис. 6.1. Виды списков: а – выпадающий; б – не выпадающий

Создание списка выполняется с помощью тегов <option>, расположенных внутри тега <select>, например,

```
<select name="month">
<option value="1">Январь</option>
<option value="2">Февраль</option>
<option value="3">Март</option>
<option value="4">Апрель</option>
<option value="5">Май</option>
<option value="6">Июнь</option>
<option value="7">Июль</option>
<option value="8">Август</option>
<option value="9">Сентябрь</option>
<option value="10">Октябрь</option>
<option value="11">Ноябрь</option>
<option value="12">Декабрь</option>
</select>
```

создается выпадающий список с 12 месяцами. Атрибут value тега <option> определяет значение, передаваемое серверу при выборе соответствующего значения из списка.

Тег <select> имеет следующие необязательные атрибуты:

name – имя элемента при отправке соответствующих данных серверу (аналогично атрибуту name тега <input>);

size – число отображаемых строк списка (если size > 1), то список становится не выпадающим);

multiple – позволяет выбирать сразу несколько элементов в списке.

Пример. Не выпадающий список с множественным выбором

```
<select name="month" size=5 multiple>
<option value="1">Январь</option>
<option value="2">Февраль</option>
...
</select>
```

Пример. Выпадающий список с множественным выбором

```
<select name="month" size=1 multiple>
<option value="1">Январь</option>
<option value="2">Февраль</option>
...
</select>
```

В последнем примере, несмотря на то, что параметр `size=1`, будет отображен не выпадающий список, т.к. множественный выбор возможен только для таких типов списков и атрибут `multiple` имеет более высокий приоритет, чем атрибут `size`.

Тег <textarea>

Тег-контейнер `<textarea>` служит для ввода многострочного текста в поле ввода (рис. 6.2). В отличие от поля ввода, данный тег позволяет вводить полноценный текст больших размеров. Обычно максимальный размер текста ограничивают 64 Кб, но может быть и больше. В конечном итоге, максимальный объем поля `textarea` определяется разработчиком сайта и возможностями сервера.

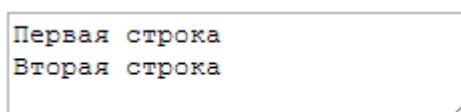


Рис. 6.2. Пример поля ввода текста `textarea`

Для создания поля ввода текста в HTML-документе используется запись:

```
<textarea>
Первая строка
Вторая строка
</textarea>
```

при этом переносы строк в тексте, заключенном в теге `<textarea>` будут сохранены (см. рис. 6.2).

Тег-контейнер `<textarea>` имеет следующие атрибуты:

cols – ширина поля в символах;
rows – высота поля в символах;
name – имя поля (действие аналогично такому же параметру в теге <input>);
maxlength – максимальное число вводимых символов (не поддерживается некоторым браузером);

Пример. Поле textarea с набором описанных атрибутов.

```
<textarea cols=40 rows=20 name="text" maxlength=10000>  
Начальный текст  
</textarea>
```

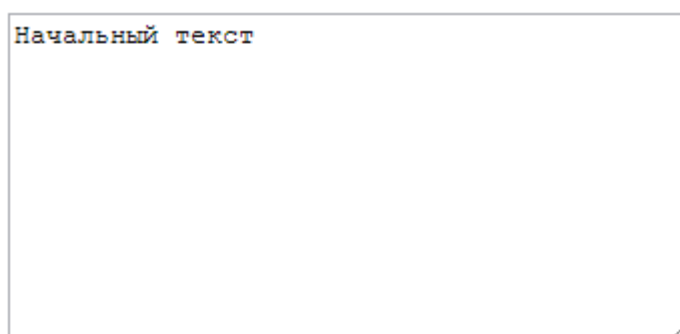
A screenshot of a web browser showing a text area. The text area is rectangular and contains the text "Начальный текст" in a blue font. The text area is surrounded by a thin border and has a small cursor icon in the bottom right corner.

Рис. 6.3. Вид поля textarea приведенного примера

6.3. Примеры HTML-форм

В данном пункте рассмотрим примеры использования HTML-форм, часто встречающихся на страницах сайтов. В качестве первого примера рассмотрим форму регистрации пользователя на сайте (рис. 6.4).

Ф.И.О.	<input type="text" value="Сергей"/>
E-mail	<input type="text" value="mail@mail.ru"/>
Пароль	<input type="password" value="....."/>
Повтор пароля	<input type="password" value="....."/>

Я принимаю пользовательское соглашение

Рис. 6.4. Пример формы регистрации пользователя

HTML-разметка данной формы имеет вид:

```
<form action="register.php" method="post">  
<table border=0>
```



```

<tr>
  <td>Ф.И.О.</td>
  <td><input type="text" name="fio" size=20 /></td>
</tr>
<tr>
  <td>E-mail</td>
  <td><input type="text" name="email" /></td>
</tr>
<tr>
  <td>Пароль</td>
  <td><input type="password" name="password" /></td>
</tr>
<tr>
  <td>Повтор пароля</td>
  <td><input type="password" name="confirm" /></td>
</tr>
</table>

<p><input type="checkbox" name="agree">Я принимаю
пользовательское соглашение
<p><input type="submit" value="Зарегистрироваться" />

</form>

```

Обратите внимание, что для работы данной формы необходимо создать скрипт register.php на сервере, к которому будет отправлен POST-запрос. На стороне скрипта введенные данные будут доступны по именам, указанных в атрибутах name соответствующих полей формы.

Во втором примере рассмотрим форму обратной связи пользователя с подразделениями сайта (рис. 6.5).

Ф.И.О.

▾

Скажите, как работает эта форма?

Рис. 6.5. Вид формы обратной связи

HTML-разметка формы:

```
<form action="support.php" method="post">
```

```

<p>Ф.И.О. <input type="text" name="fio" />
<p><select name="type">
<option value="1">Администрации сайта</option>
<option value="2" selected>Службе поддержки сайта</option>
<option value="3">Не определено</option>
</select>

<p><textarea cols=80 rows=10 name="message"></textarea>

<p><input type="submit" value="Отправить" />

</form>

```

Для формы обратной связи лучше использовать POST-запрос, т.к. размер текста в поле `textarea` может быть большим и GET-запрос отбросит часть информации. Кроме того, поле `textarea` может содержать символы, не допустимые в GET-запросе.

В списке тега `<select>` по умолчанию выбирается второй элемент. Это достигается с помощью атрибута `selected` тега `<option>`.

Последним примером рассмотрим форму добавления комментария на странице сайта (рис. 6.6).

Комментарии

- **Сергей** Первый комментарий
- **Алексей** Второй комментарий

Ф.И.О.

Рис. 6.6. Форма добавления комментариев на странице сайта

```

<h2>Комментарии</h2>
<ul>
<li><b>Сергей</b> Первый комментарий
<li><b>Алексей</b> Второй комментарий
</ul>

<hr>

```

```
<form action="comments.php" method="post">
<p>Ф.И.О. <input type="text" name="fio" />
<input type="hidden" name="id" value="10" />
<p><textarea cols=80 rows=10 name="comment"></textarea>
<p><input type="submit" value="Отправить" />
</form>
```

В приведенной форме используется скрытое поле `<input type="hidden">`, которое не отображается на экране, но передает скрипту информацию, в частности значение 10 идентификатора `id`. Данный идентификатор может быть использован для определения страницы сайта, к которой следует добавлять комментарий. Благодаря такому подходу можно создать универсальный скрипт `comments.php`, который будет «знать» в какой раздел добавлять комментарий. Все что нужно сделать – это разместить скрытое поле с нужным значением `id`.