Кириченко А. В.,
Хрусталев А. А.

HTML5+CSS3

ОСНОВЫ СОВРЕМЕННОГО WEB-ДИЗАЙНА

НиТ
издательство

СЕРИЯ — **САМ СЕБЕ ПРОГРАММИСТ!** — СЕРИЯ

"Наука и Техника"

Санкт-Петербург


HTML
5
CSS
3

Кириченко А.В., Хрусталев А.А.

HTML5 + CSS3.

Основы современного web-дизайна

"Наука и Техника"

Санкт-Петербург

УДК 681.3.068; 004.738
ISBN 978-5-94387-750-6

Кириченко А.В., Хрусталев А.А.

HTML5+CSS3. ОСНОВЫ СОВРЕМЕННОГО WEB-ДИЗАЙНА — СПб.: "Наука и Техника", 2018 г. — 352 с., ил.

Серия "Сам себе программист!"

С помощью нашей книги вы сможете легко и непринужденно освоить базовый курс HTML5 и CSS3. Уровень подаваемого материала книги позволит эффективно использовать полученные знания как новичку, так и более опытному программисту, желающему освоить основы или улучшить свои навыки web-программирования и web-дизайна. Помимо базового синтаксиса обоих языков, вы узнаете: как редактировать и работать с текстом в HTML5; как использовать списки, таблицы, скрипты и ссылки; как размещать мультимедиа-объекты и создавать макет web-страницы, как пользоваться формами и фреймами; как форматировать и видоизменять блоки и структуру документов при помощи CSS3, что из себя представляют визуальные функции CSS3 и многое другое.

Каждый теоретический отрезок сопровождается практическим примером, наглядно демонстрирующем пройденный материал.

Книга подойдет для всех желающих освоить или начать лучше ориентироваться в HTML5 и CSS3, которые являются на данным момент основными инструментами современного Web-дизайна.

ISBN 978-5-94387-750-6

Контактные телефоны издательства:
(812) 412 70 26

Официальный сайт: www.nit.com.ru

Содержание

HTML5

Глава 1.

Введение в HTML. Структура документа HTML

В этой главе вы узнаете:

- *Что такое HTML?*
- *Какова структура документа html?*
- *Как создать HTML-файл?*

1.1. Что такое HTML?

Сеть Интернет – сеть глобального пользования, и поэтому вся информация в сети должна быть представлена на универсальном языке, который понимали бы все пользователи. Языком публикации, используемым в World Wide Web, является HTML (Hiper Text Markup Language – язык разметки гипертекста). Техническую информацию об этом языке (его спецификацию) на английском языке можно найти на сайте World Wide Web Consorcium – w3c.com

HTML – язык разметки, предоставляющий разработчикам следующие возможности:

- Представлять информацию в сети в виде электронных документов, с информационным содержимым в виде форматированного текста, таблиц, списков, фотографий;
- Включать в документы звуковые фрагменты, видеоклипы, электронные таблицы и другие приложения, и элементы мультимедиа;
- Осуществлять загрузку документов посредством активизации гипертекстовой ссылки
- Разрабатывать формы для осуществления взаимодействия с удаленными службами (поисковыми роботами, on-line магазинами и т.п.)

Разметка документов заключается в том, что документ представляется в виде последовательности элементов. Например, чтобы отобразить в окне

браузера простейшую текстовую информацию HTML-документ должен иметь следующий вид:

```
<html>
  <head>
    <title> Борис Бурда -  Эти смешные музыканты </title>
  </head>

  <body>
    <h1> Эти смешные музыканты </h1>
     <p> В музыке каждый понимает по мере веса слона,
         в свое время наступившего ребенку на ухо.
         Президент США Улисс Симпсон Грант, тот, который
         с полтинника, вообще говорил, что знает только
         две мелодии, одна из которых - "Янки дудл". А
         на вопрос, какая же вторая, гордо отвечал: "Не
         "Янки дудл". Впрочем, лучше, когда руководитель
         так разбирается в музыке, чем, скажем, как
         Жданов, который учил композиторов, какая должна
         быть музыка - чтоб ему, Жданову, было приятно ее
         напевать. Лучше бы Жданов пил, что ли - на Гранта
         еще в бытность его генералом за алкоголизм самому
         Линкольну стучали. Тот, правда, отреагировал
         весьма своеобразно - велел разузнать, какой
         сорт виски Грант пьет, чтоб и прочим генералам
         послать за свой счет по бочке, ибо воевал-то
         Грант неплохо. Так что лучше, чтоб руководитель
         своих музыкальных вкусов не демонстрировал.
         Разве что как Елизавета Английская, даровавшая
         некому доктору Джону Булю (интересное, кстати,
         совпадение) герб с надписью "Sol, mi, re, fa"
         - за сочинение английского гимна "Боже, храни
         королеву". А то вот Николай I не любил Глинку
         и разрешал заменять провинившимся офицерам
         гауптвахту... посещением оперы "Руслан и Людмила".
         Его бы в современную Россию - государственный гимн
         послушать. Отомстили бы за декабристов! </p>
     <p> Еще Наполеон говорил, что в российской кампании
         у его войска было два главных врага - морозы и
         русская военная музыка. Да и генералы Великой
         французской революции, прося подкреплений, писали:
         "Пришлите два полка солдат или тысячу экземпляров
         "Марсельезы". Музыка - дело смертоносное, причем
```

```
        не только для людей. Недавно в одном старинном
        английском замке состоялся рок-концерт, после
        которого из замка начисто исчезли крысы. Ни
        кошки, ни отрава не смогли добиться того же, что
        пленительные звуки современной музыки. Это еще раз
        доказывает, что люди сильнее крыс и слухи о их
        живучести несколько преувеличены. </p>
  </body>
</html>
```

Рис. 1.1. Отображение простейшей текстовой информации в браузере

На экране монитора это будет выглядеть так:

Каждое объявление тега обычно включает три части:

1. *начальный (открывающий) тег;*
2. *содержимое;*
3. *конечный (закрывающий) тег.*

Имя тега отображается в начальном (<имя тега>) и конечном (</имя тега>) тегах. Сами теги на экране не отображаются.

Все теги языка HTML выделяются символами-ограничителями (<и>), между которыми прописывается имя тега и, возможно, его атрибуты. Об атрибутах мы поговорим в следующих уроках.

Единственным исключением являются теги комментария. Для них предусмотрено следующее написание: <! --Текст комментария -->

Имена тегов и их атрибутов не чувствительны к регистру, т.е. не имеет значения заглавными или прописными буквами они написаны. Например, записи <bOdY> и <BoDy> абсолютно идентичны с точки зрения HTML.

1.2. Какова структура документа html?

Документ в формате HTML состоит из трех основных частей:

- строки, объявляющей файл как документ на языке HTML5;
- заголовка, заключенного в тег HEAD;
- тела документа, представляющего собой тег BODY, если документ имеет классическое, однооконной представление. В теле документа, собственно, и содержится вся, предлагаемая пользователю, информация.

Заголовок и тело документа заключены в тег-контейнер HTML. Все остальные теги разметки располагаются либо в заголовке, либо теле документа.

Минимальная структура HTML-документа представлена ниже:

```
<! DOCTYPE html>
<html>
  <head>
    <title> Заголовок страницы </title>
  </head>
  <body>
    Здесь располагается текст страницы
  </body>
</html>
```

1.3. Как создать HTML-файл?

Для того, чтобы создать HTML-файл в операционной системе Windows не требуется установка никаких дополнительных программ. Достаточно использовать Блокнот (Notepad).

Рис. 1.2. Окно текстового редактора

Необходимо ввести некоторую информацию в текстовый редактор, сохранить её, а затем загрузить в свой браузер. Звучит очень просто, не правда ли? Давайте попробуем создать сайт-портфолио одной из команд-участников всероссийского технологического марафона TechRussia.

Шаг 1.

Введите текст в новый документ в текстовом редакторе в том же виде, в котором вы его видите на рисунке 1.3.

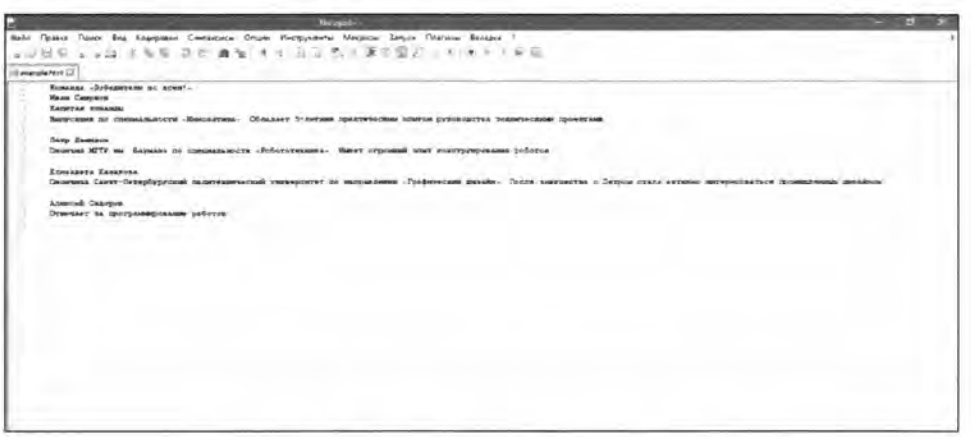


Рис. 1.3. Текст сайта в окне редактора

Шаг 2.

Выберите команду "Файл->Сохранить как..." и сохраните ваш документ в формате html. Для этого выберите место сохранения файла (Совет: создавайте отдельную папку под каждый создаваемый вами сайт!) и измените тип файла, как показано на рисунке 1.4.

Внимание! Не забудьте выбрать пункт UNF-8 в разделе "Кодировка". Это необходимо для корректного отображения кириллических символов.

Рис. 1.4. Сохранение HTML-файла

Шаг 3.

Откройте сохраненный HTML-файл в браузере и убедить, что он выглядит так же, как показано на рисунке 1.5.

Как вы наверняка заметили, отображение текста на сайте сильно отличается от того, как он выглядел в окне текстового редактора. Это означает, что браузер самостоятельно не может определить *структуру* документа. Для этого и используется язык HTML.

Рис. 1.5. Вид страницы в браузере без определения структуры

Шаг 4.

Добавьте структуру к HTML-документу и содержимому страницы с помощью разметки таким образом, чтобы страница в браузере выглядела как на рисунке 1.6. Подробное описание тегов, которые вам понадобятся, представлено в памятке. Распечатайте её и постарайтесь как можно скорее запомнить.

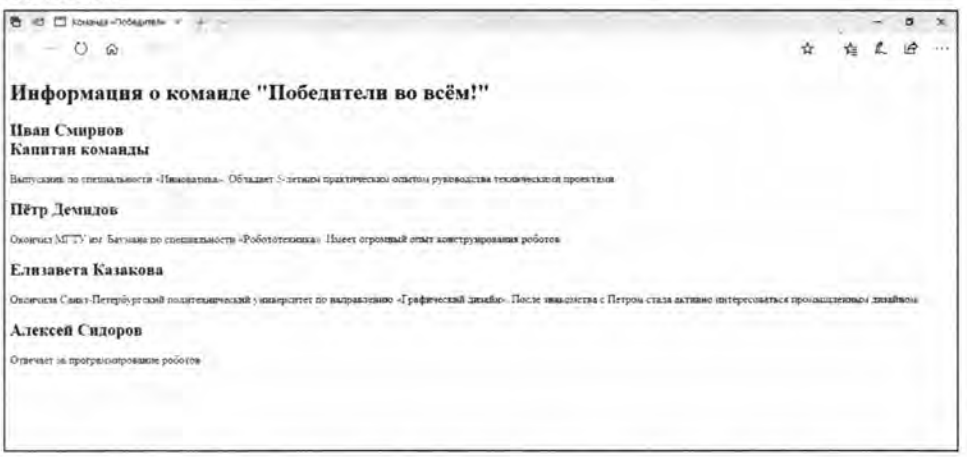


Рис. 1.6. Вид страницы в браузере после определения структуры

Памятка

1. Окружите свой код тегами <html> и </html>. Это позволит браузеру понять, что в файле содержится HTML-код.
2. Затем добавьте теги <head> и </head>. Этот раздел "шапка" содержит описание вашей веб-страницы. Заголовок обычно появляется вверху окна браузера. "Шапка" состоит из тегов <head> и </head> и всего, что находится между ними.
3. Основная часть страницы состоит из тегов <body> и </body> и всего, что находится между ними.
4. Теги <h1> и </h1> используются для выделения заголовка. Весь текст внутри этих тегов – заголовок.
5. Теги <h2> и </h2> окружают подзаголовок. Важно понимать, что <h2> - это подзаголовок заголовка <h1>.
6. Теги и окружают часть текста, которая является отдельным абзацем. Это может быть одно или несколько предложений.

HTML5

Глава 2.

Служебная информация web-страницы. Данные для поисковиков. Тег HEAD

В этой главе вы узнаете:

- *Как создать заголовок документа*
- *Как задать название документа? Тег TITLE*
- *Как задать URL-адрес документа? Тег BASE*
- *Как создать ссылку? Тег LINK*
- *Как задать свойства документа? Тег META*
- *Как изменить стиль документа? Тег STYLE*
- *Как добавить скрипт? Тег SCRIPT*

2.1. Как создать заголовок документа?

Тег HEAD (элемент заголовка) предназначен для содержания в нем информации о документе, а именно:

- название документа,
- точный адрес (URL) документа,
- параметры, управляющие загрузкой и обработкой документа
- ключевые слова, по которым осуществляется поиск поисковыми роботами и т.п.

В заголовок документа также могут быть помещены описание стиля документа и скрипты.

Начальный и конечный теги HEAD являются необязательными. Это значит, что если присутствует начальный тег BODY (тела документа), то конечный тег HEAD можно не указывать, и наоборот.

Заголовок может содержать в себе следующие элементы:

- название документа (тег TITLE)
- полный URL документа (тег BASE)
- управляющую информацию (тег META)
- список ссылок (тег LINK)
- описание стилей (тег STYLE)
- задание скриптов (тег SCRIPT)

2.2. Как задать название документа? Тег TITLE

Тег TITLE задает название документа. Этот тег разметки, строго говоря, не является обязательным, однако его использование настоятельно рекомендуется. Он присваивает документу название, независимое от имени файла, которое отображается в строке заголовка браузера. Также это имя используется по умолчанию при добавлении документа в папку "Избранное". Документу, тег TITLE, которого не задан, браузер в качестве его имени будет использовать надпись "NoTitle" либо полный адрес документа.

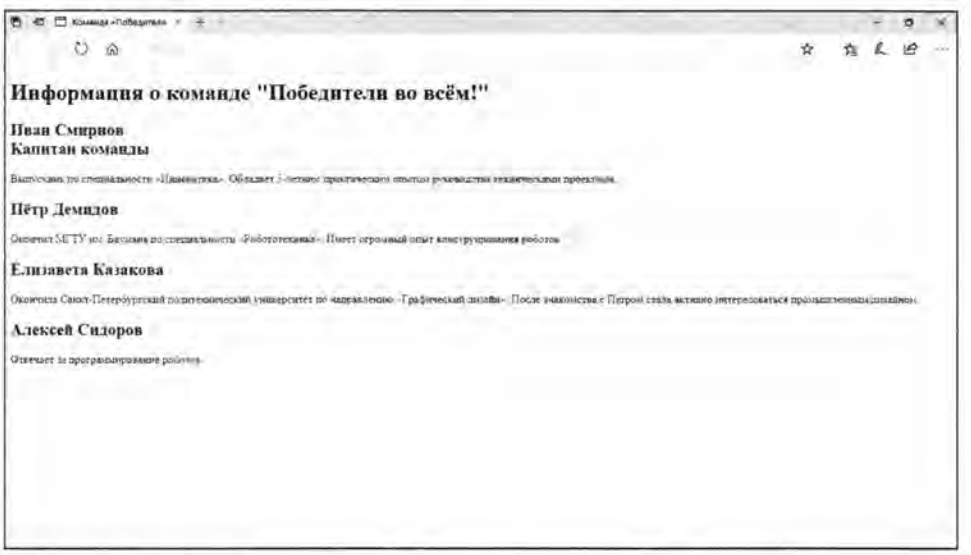


Рис. 2.1. Демонстрация результата использования тега TITLE

Открывающий и закрывающий теги TITLE являются обязательными. Его содержимое представляет текстовую строку неограниченной длины.

Название документа должно кратко характеризовать его содержание. Учитывая это, а также возможную минимизацию окна браузера, рекомендуется в названии документа ограничиться 50-60 символами. Содержание тега TITLE не должно включать в себя других тегов разметки. Например, нельзя с помощью тега <i> вывести название документа курсивом, т.е. запись

```
<title><i> название документа </i></title>
```

недопустима.

Запись

```
<i><title> название документа </title></i>
```

также неприемлема, так как тег <i> не является элементом заголовка.

2.3. Как задать URL-адрес документа? Тег BASE

Тег BASE используется для явного задания полного URL-адреса документа. Это бывает полезно ввиду того, что общепринятым стилем задания гипертекстовых ссылок является относительная их адресация. То есть при задании ссылки на документ указывается не полный его URL-адрес, а задается его месторасположение относительно текущего адреса. Так вот тег BASE как раз и задает адрес, относительно которого и будут браться относительные ссылки.

Пример относительной ссылки:

```
<a href="/BOOK1/chapter1.html">
```

Такой стиль полезен тем, что при переносе всего дерева документов в другое место, не требуется изменять ссылки в самих документах.

Использование тега BASE позволяет реализовывать относительные ссылки в том случае, когда HTML-документ перемещен (или скопирован), а все остальное дерево документов, на которые он ссылается, нет. Адрес его поменялся (например, документ лежал на http://techrussia.org, а теперь у Вас дома, на диске C), однако при активизации относительной ссылки, она будет взята браузером относительно исходного адреса, прописанного в теге BASE.

Тег BASE имеет один обязательный атрибут **href**, значением которого является полный URL документа.

В листинге показано применение тега BASE и относительных ссылок:

```
<!DOCTYPE html><html>
<head>
   <title> Документ с использованием тега BASE</title>
   <base href= "http://techrussia.org">
</head>
```

```
<body>
        ..... текст документа.....
        ..... текст документа.....
   <p><a href="/assets/img/Robots/Design.png">
          Ссылка на изображение Design.png </a></p>
          ..... текст документа.....
</body>
</html>
```

В этом примере переход по относительным ссылкам задается относительно URL - адреса http://techrussia.org. Таким образом, заданные в этом документе ссылки в абсолютном варианте всегда (независимо от месторасположения документа) будут иметь следующий вид: http://techrussia.org/assets/img/Robots/Design.png. Если бы базовый адрес задан бы не был, то эти ссылки интерпретировались бы относительно каталога, в котором находится данный документ. Соответственно, при перемещении документа изменялись бы цели относительных ссылок.

Вид страницы в браузере представлен на рисунках 2.2 и 2.3.

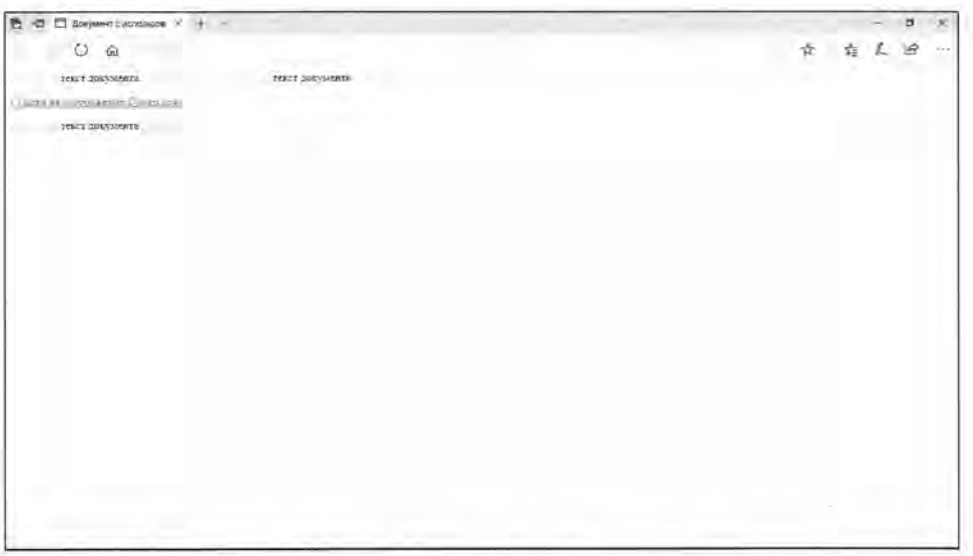


Рис. 2.2. Вид HTML-страницы с тегом BASE в окне браузера

Начальный тег BASE обязателен, конечный тег запрещен.

Необязательный атрибут **target** – задает целевую ссылку на фрейм в данном документе.

Рис. 2.3. Вид HTML-страницы после нажатия ссылки

Значением его является имя фрейма.

Таким образом, общий вид задания тега BASE выглядит так:

```
<BASE href=URL target=frame_name>
```

Тег BASE часто используется для организации узлов, которые имеют так называемые "зеркала". Часть документов основного сервера переносится на "зеркальный". Обычно это делается для повышения скорости взаимодействия с удаленными пользователями и применяется в совокупности с запретом на кэширование перенесенных документов (как пользовательскими браузерами, так и proxy-серверами).

Тег BASE можно использовать и в заголовке, и в теле документа, причем несколько раз. Область действия тега BASE определяется от места его задания и до конца документа, или до следующего объявления тега BASE, если таковой имеется.

2.4. Как создать ссылку? Тег LINK

Данный элемент указывает на связь, отношение между содержащим его документом и другим ресурсом сети. На данный момент единственное его практическое применение заключается в подключении внешних каскадных

таблиц стилей (CSS), ответственных за визуальное представление документа.

Информация о документе, предоставляемая тегом LINK может быть использована некоторыми поисковыми машинами для оптимизации поиска.

Подробное описание этого тега разметки читайте в следующих уроках.

2.5. Как задать свойства документа? Тег META

Тег META используется для задания некоторых свойств документа (а именно: автора, списка ключевых слов, кодировки и т.п.), благодаря чему позволяет управлять обработкой HTML-документа. Этот тег, вместе с тегом TITLE является наиболее используемым при задании заголовка.

При задании META начальный тег обязателен, конечный тег запрещен.

Атрибуты:

- **name** – указывает имя свойства;
- **content** – задает значение свойства;
- **scheme** – указывает имя схемы, используемой для обработки значения свойства;
- **http-equiv** – используется вместо атрибутаname для указания имени http-сообщения;
- **lang** – информация о языке. Необязательный атрибут;
- **dir** – указывает направленность текста Необязательный атрибут.

Каждый тег META содержит в себе пару свойство-значение. Атрибут **name** (http-equiv) указывает свойство, атрибут **content** – значение.

Например:

```
<meta charset="UTF-8">

<META name="description" content="best techrussia team,
portfolio">
```

```
<META name="keywords" content="techrussia, marathon, team,
winners, portfolio">
```

В первом случая указана кодировка HTML-документа, во втором случае – описание документа, в третьем – ключевые слова. Оба последних тега META обычно имеют одинаковое значение в целях повышения эффективности обнаружения их поисковыми роботами, так как именно по этим параметрам в значительной степени осуществляется отбор нужных документов.

Атрибут **http-equiv** может использоваться вместо атрибута **name**, для задания свойств HTML-документа на уровне http-заголовка. Через атрибут **http-equiv** осуществляется доступ к полям HTTP-заголовка.

Первое полезное применение этого элемента заключалось в осуществлении принудительной перезагрузки документа браузером. Для этого с помощью атрибута **http-equiv** используется http-оператор **refresh**. Время, через которое надо произвести перезагрузку указывается через атрибут **content**, а адрес загружаемого документа – атрибутом **url** оператора **refresh**.

Пример написания:

```
<META http-equiv="refresh" content="5"; team.html">
```

При таком задании, через 5 секунд после загрузки текущего документа браузер автоматически перейдет к загрузке документа team.html.

Если **url** не задан, то происходит просто обновление документа.

Такое применение тега META позволяет выстроить автоматически перезагружаемую последовательность документов.

Практически во всех HTML-документах тег META используется для их описания посредством задания списка ключевых слов и краткой информации о содержимом документа. Ключевые слова, вместе с названием документа, помогают поисковым машинам найти документ. В своих отчетах они выдают название документа, прописанное в элементе TITLE, и краткое его описание, заданное через тег META.

Для указания списка ключевых слов и краткой информации о документе в заголовке используются два META тега:

```
<META name="description" content="портфолио команды победители
во всём, всероссийский технологический марафон, TechRussia">

<META name="keywords" content="команда, техраша, techrussia,
портфолио, победители">
```

В целях большей эффективности поиска краткая информация и ключевые слова обычно совпадают, как это было продемонстрировано выше в одном из примеров.

Через тег META можно указать кодировку содержимого документа. Тег META тогда принимает вид:

```
<meta charset="UTF-8">
```

С помощью тега META можно запретить кэширование документа, что бывает полезно при частом обновлении документа. Для осуществления этой операции, в тег META включается оператор **pragma**. Оператор устанавливается в положение no-cashe. META-тег тогда примет вид:

```
<META http-equiv="pragma" content="no-cache">
```

При кэшировании документа можно указать время, до которого имеет место соответствие закэшированного документа с его оригиналом на сервере. В данном случае используется HTTP-оператор **expires**, и тег META принимает вид:

```
<META http-equiv="expires" content="Monday, 18-May-2018
00:00:01">
```

Замечание: в данном случае дата задается для корректировки HTTP-заголовка и поэтому должна иметь следующий формат, указанный в примере.

2.6. Как изменить стиль документа? Тег STYLE

Тег STYLE предназначен для размещения в нем элементов описания стиля. При этом, согласно правилам каскадирования стилей, если какие из них совпадают по имени с элементами описания стиля, заданными во внешнем файле (подключенном через тег LINK), то они заменяют элементы из внешнего файла.

Начальный тег обязателен, конечный тег обязателен.

Атрибуты:

- **type** – обязательный атрибут, указывающий язык таблицы стилей, содержащейся в данном элементе. Язык таблицы стилей указывается как тип содержимого. Обычно это text/css либо text/javascript.

- **media** – необязательный атрибут, задает целевое устройство для информации о стиле. Имеет следующий вид:

  media=дескриптор_устройства (например,projection - проектор).
  По умолчанию установлен в значение screen (экран компьютера).

Таблицы стилей значительно повышают эффективность разработки визуального представления документа (его внешнего вида), предлагая для этого богатый набор возможностей.

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Команда "Победители во всём!" </title>
      <meta charset="UTF-8">
      <style type="text/css">
      h1{color:red;
border-width:3;
border-style:solid;
text-align:center}
      </style>
   </head>
   <body>
      <h1> Информация о команде "Победители во всём!" </h1>
      <h2> Иван Смирнов <br> Капитан команды </h2>
      <p> Выпускник по специальности "Инноватика".
         Обладает 5-летним практическим опытом руководства
         техническими проектами. </p>
      <h2> Пётр Демидов </h2>
     <p> Окончил МГТУ им. Баумана по специальности
      "Робототехника". Имеет огромный опыт конструирования
      роботов.</p>
   <h2> Елизавета Казакова </h2>
   <p> Окончила Санкт-Петербургский политехнический университет
         по направлению "Графический дизайн". После знакомства с
         Петром стала активно интересоваться промышленным
         дизайном. </p>
   <h2> Алексей Сидоров </h2>
   <p> Отвечает за программирование роботов. </p>
   </body>
</html>
```

Благодаря простейшей таблице стилей заголовок, задаваемый использованием тега **h1**, отображен с выравниваем по центру и заключен в рамку. И если в дальнейшем в тексте документа будут присутствовать заголовки первого уровня (тег **h1**), то они также будут расположены по центру и обведены рамкой.

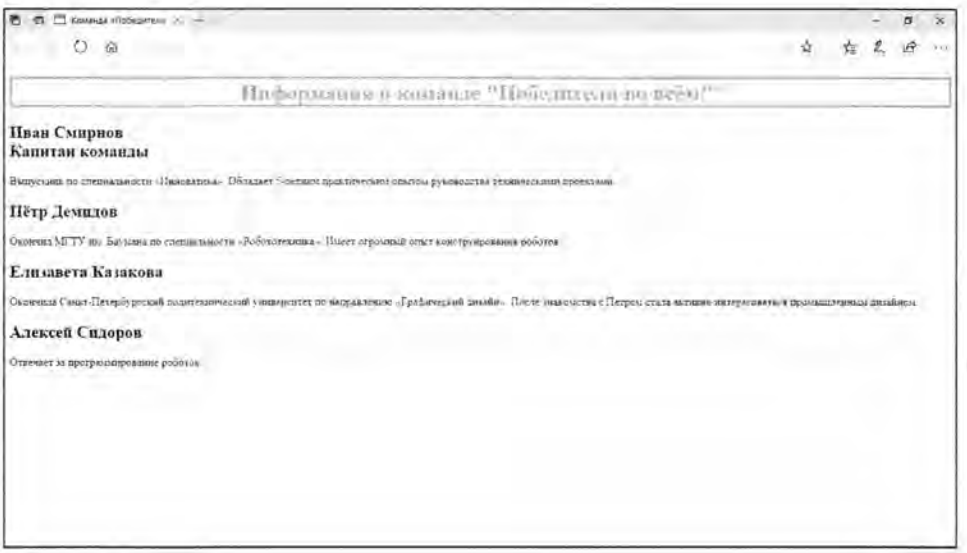


Рис. 2.4. Пример использования тега STYLE

2.7. Как добавить скрипт? Тег SCRIPT

Тег SCRIPT служит для размещения кода сценария, написанного на языке скриптов (например, JavaScript, VBScript, JScript). Этот тег может быть задан и в заголовке, и в теле HTML-документа, причем несколько раз.

Начальный тег обязателен, конечный тег обязателен.

Атрибуты:

- **type** - указывает язык, на котором написан скрипт, как тип содержимого (например, type="text/javascript"). Является обязательным атрибутом (значение по умолчанию не задано).
- **src** – указывает месторасположение (URL) внешнего скрипта. Общий вид: src=URL.
- **language** – указывает язык скрипта, содержащегося в элементе. Его значением является идентификатор языка. Является нежелатель-

ным элементом, так как идентификаторы языка не стандартизированы. Вместо него используется атрибут **type**. Пример задания: language="javascript".

- **defer** – необязательный логический атрибут. Его указание сообщает браузеру, что скрипт не будет генерировать содержимое текста.
- **charset** – кодировка символов. Необязательный атрибут, определяемый в любом месте документа. Указывает кодировку внешнего скрипта (заданного атрибутом **scr**), и не относится к содержимому тега SCRIPT.

Если не задан атрибут **scr**, содержимое тега SCRIPT интерпретируется браузером как скрипт. В том случае, если этот атрибут задан, скрипт загружается с указанного URL, а все содержание тега SCRIPT браузером игнорируется.

Более подробная информация об использовании скриптов в HTML-документах будет описана в следующих главах.

HTML5

Глава 3.

Тело HTML-документа. Тег BODY

В этой главе вы узнаете:

- *Как использовать атрибуты тега BODY?*
- *Как присваивать тегам уникальные имена? Атрибуты id и class*

3.1. Как использовать атрибуты тега BODY?

В теле документа располагается содержательная его часть. В качестве тела документа определен тег-контейнер BODY.

Начальный и конечный теги являются необязательными (необходимость их использования определяется контекстом)

Атрибуты:

- **alink=color** – определяет цвет активной (выбранной пользователем) ссылки;
- **link=color** – указывает, каким должен быть цвет еще не просмотренной ссылки;
- **vlink=color** – задает цвет уже просмотренной ссылки;
- **bgcolor=color** – устанавливает какого цвета должен быть фон документа;
- **background=URL** – указывает месторасположение (URL) изображения, которое используется в качестве фонового;
- **bgproperties** – если установлен в положение *fixed*, то фоновое изображение не прокручивается. По умолчанию установлено положение *nonfixed*;
- **text=color** – определяет цвет текста документа;

- **leftmargin, rightmargin, topmargin=pixels** – устанавливают размер левого, правого и верхнего полей в документе соответственно. Задаются в пикселах;
- **scroll** – логический атрибут: если установлен в положение no, то полоса прокрутки документа не отображается. По умолчанию используется в значении *yes*.

Пример документа с использованием возможностей задания тега BODY:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Команда "Победители во всём!" </title>
    <base href= "http://techrussia.org">
    <meta charset="UTF-8">
    <style type="text/css">
    h1{color:#fe4918; border-width:3; border-style:solid;
text-align:center; border-color:white}
    </style>
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no link=#fe4918 vlink=#ffffff
alink=#ffffff>
    <h1> Информация о команде "Победители во всём!" </h1>
    <h2> Иван Смирнов <br> Капитан команды </h2>
  <p> Выпускник по специальности "Инноватика". Обладает
5-летним практическим опытом руководства техническими
проектами.</p>
    <h2> Пётр Демидов </h2>
  <p> Окончил МГТУ им. Баумана по специальности
"Робототехника". Имеет огромный опыт конструирования
роботов. </p>
    <h2> Елизавета Казакова </h2>
  <p> Окончила Санкт-Петербургский политехнический
университет по направлению "Графический дизайн".
После знакомства с Петром стала активно интересоваться
промышленным дизайном. </p>
    <h2> Алексей Сидоров </h2>
  <p> Отвечает за программирование роботов. </p>
  <p><a href="/#directions"> Ссылка на направления
TechRussia </a><p>
  </body>
```

```
</html>
```

На экране монитора это выглядит следующим образом:

Рис. 3.1. Вид HTML-страницы в браузере до нажатия ссылки

Рис. 3.2. Вид HTML-страницы в браузере после нажатия ссылки

Все атрибуты тега BODY являются необязательными и нежелательными, ввиду приоритетного использования каскадных таблиц стилей (CSS), предлагающих для этих целей более эффективные и развитые возможности.

3.2. Как присваивать тегам уникальные имена? Атрибуты id и class

Атрибуты **id** и **class** служат для идентификации содержащих их тегов. Через атрибут **id** тегу можно присвоить уникальное имя. Атрибут **class** причисляет тег к классу тегов, созданному разработчиком. Причем сам класс тегов образуется с первым включенным в него тегом. Если тег причисляется к нескольким классам сразу, то тогда имена классов должны быть разделены пробелами.

Атрибут **id** служит в HTML для выполнения следующих функций:

- для осуществления выборочного обращения таблицы стилей к определенным элементам (задания их стиля);
- для указания цели (якоря) гипертекстовых ссылок, что позволяет им ссылаться не только на документ в целом, но и на его отдельные части;
- для реализации ссылки на определенный элемент сценария;
- для задания имени объекта, вставляемого в документ тегом OBJECT.

Атрибут **class** служит в HTML для выполнения следующих функций:

- для осуществления выборочного обращения таблицы стилей к определенной группе элементов (задания их стиля);
- для реализации особой обработки браузером определенной (заданной разработчиком) группы элементов.

Атрибуты **id** и **class** могут быть установлены почти для всех тегов языка HTML. Именование HTML-тегов и объединение их в определенные группы позволяет осуществлять обращение к ним, что особенно полезно при применении к ним таблиц стилей.

Пример использования:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Команда "Победители во всём!" </title>
    <base href= "http://techrussia.org">
    <meta charset="UTF-8">
    <style type="text/css">
    h1{color:#fe4918; border-width:3; border-style:solid;
text-align:center; border-color:white}
    #cite{text-align:right; color:#45525b; font-style:italic}
    p.ssyl{text-align:center}
    </style>
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no link=#fe4918 vlink=#ffffff alink=#ffffff>
    <h1> Информация о команде "Победители во всём!" </h1>
    <h2> Иван Смирнов <br> Капитан команды </h2>
  <p> Выпускник по специальности "Инноватика". Обладает
5-летним практическим опытом руководства техническими
проектами. </p>
    <h2> Пётр Демидов </h2>
  <p> Окончил МГТУ им. Баумана по специальности
"Робототехника". Имеет огромный опыт конструирования роботов.
</p>
    <h2> Елизавета Казакова </h2>
  <p> Окончила Санкт-Петербургский политехнический университет
по направлению "Графический дизайн". После знакомства с Петром
стала активно интересоваться промышленным дизайном.</p>
    <h2> Алексей Сидоров </h2>
  <p> Отвечает за программирование роботов. </p>
  <p class="ssyl"><a href="/#directions"> Ссылка на
направления TechRussia </a><p>
  <divid="cite"> "Мы живем в эпоху, когда расстояние от самых
безумных фантазий до совершенно реальной действительности
сокращается с невероятной быстротой." Максим Горький </div>
  </body>
</html>
```

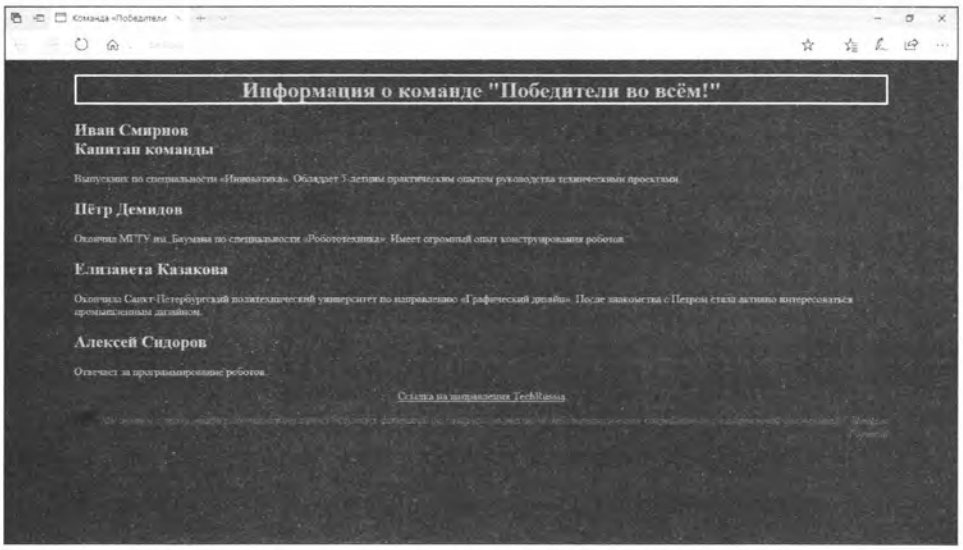


Рис. 3.3. Вид HTML-страницы в браузере после применения атрибутов id и class

HTML5

Глава 4.

Работа с текстом

4.1. Как указать язык документа?

Сведения о языке, на котором представлена информация в документе, может быть задана с помощью атрибута **lang**. Этот атрибут присутствует почти у всех тегов языка HTML и может быть задан в любом месте документа.

Сведения о языке могут быть полезны в следующих случаях:

- Для более эффективной работы поисковых машин;
- Для синтезаторов речи;
- Для выбора глифов при высококачественной печати документа;
- Для проверки грамматики и орфографии.

Атрибут **lang** указывает код содержимого тега. Коды языков состоят (в общем случае) из первичного кода и ряда подкодов, который может быть пустым.

```
Код языка = первичный код - подкод
```

Например,

- en — английский;
- en-us — американская версия английского;
- en-au — австралийская версия английского.

Каждый HTML-документ наследует информацию о языке в следующем порядке (от высшего к низшему)

- Атрибут **lang** заданный непосредственно для самого тега;

- Ближайший родительский тег, у которого определен атрибут **lang**;
- Значение, принимаемое браузером по умолчанию.

Пример:

```
<!DOCTYPE html >
<html lang="de" >
   <head>
      <title>Das ist multilanguage document</title>
   </head>
   <body>
      .....Текст на немецком языке.....
      .....Текст на немецком языке.....
      .....Текст на немецком языке.....
      <Plang="es"> .....Текст на испанском.....</P>
      <P>.....Текст на немецком языке.....</P>
      <P>.....Текст на немецком языке.....</P>
      <citelang="en">...цитата на английском...</cite>
      .....Текст на немецком языке.....
      .....Текст на немецком языке.....
      .....Текст на немецком языке.....
   </body>
</html>
```

4.2. Как указать направление текста?

В связи с интернационализацией сети Internet, для корректного отображения текста документа, информации о языке может быть недостаточно. Атрибут **dir** предоставляет разработчикам возможность указывать направленность текста. Этот атрибут так же, как и атрибут **lang**, может быть задан в любом месте документа почти для любого тега.

Принимаемые значения:

- LTR – слева направо (dir=LTR);
- RTL – справа налево (dir=RTL).

По умолчанию используется направленность текста слева направо.

Атрибут **dir** действует на протяжении всего тега, включая все вложенные в него теги.

Для того, чтобы установить основное направление текста для всего документа, задание атрибута **dir** производится в теге HTML.

Пример:

```
<!DOCTYPE html>
<html dir="rtl" >
<head>
<title> заголовок также будет выводиться справа налево </
title>
</head>
   <body>
      ……текст справа налево……
      ……текст справа налево……
      <pdir="ltr">.……текст слева направо…….</p>
      <p>…...текст справа налево……</p>
      ……текст справа налево……
      ……текст справа налево……
   </body>
</html>
```

4.3. Что такое структурное и физическое форматирование текста документа?

Для форматирования текста в HTML могут использоваться два подхода: задание структуры документа и использование элементов непосредственного (физического) форматирования.

При структурном представлении текста, он разбивается на фрагменты, которым ставится в соответствие определенный структурный тип, а вместе с ним и его свойства. Описание же этих свойств происходит применительно к структурному типу (в начале HTML-документа), а не к тексту. Благодаря этому достигается разделение содержательной и описательной частей HTML-документа. Фрагменты текста, которые могут быть выделены – самые разные: цитата (тег <cite>), фрагмент компьютерного кода (тег <code>), выделенный текст (тег), тег (элемент <samp>) и другие.

При непосредственном (физическом) форматировании визуальное свойство ставится в соответствие и характеризует непосредственно фрагмент

текста. Например, тег <i> выводит текст, заключенный между его начальным и конечным тегами, курсивом; тег <b> – жирным шрифтом, и т.д.

Момент разделения HTML-элементов форматирования текста на структурные и физические может оказаться достаточно сложным для неподготовленного читателя. Поэтому при первом ознакомлении с HTML им рекомендуется ориентироваться на видимый эффект, производимый тем или иным элементом разметки. Для этих целей более удобным и понятным является использование тегов физического форматирования, так как некоторые теги структурного форматирования приводят к одинаковому визуальному эффекту. Например, для выделения текста курсивом предусмотрен один тег физического форматирования <i>. Этого же видимого эффекта можно достичь при помощи четырех тегов структурного форматирования <var>, , <dfn>, <cite>, просто они еще характеризуют смысловое содержание выделенного текста. Понимание структуры документа, а также необходимость ее четкого представления произойдет при разработке и организации сложных HTML-документов.

Первое время (в ранних версиях HTML) использовался в основном метод физического форматирования, но с развитием HTML-технологий и каскадных таблиц стилей приоритетным стал структурный подход.

4.4. Какие существуют теги структурного форматирования текста?

Тег ABBR - отмечает заключенный в себе текст как аббревиатуру. А атрибут <title> добавляет всплывающую подсказку к тексту, в которой можно дать расшифровку аббревиатуры.

Пример:

```
<!DOCTYPE HTML>
<html>
<head>
<title> Тег ABBR </title>
<meta charset="utf-8">
<style>
   abbr {
    border-bottom: 1px dashed red; /* Пунктирное подчеркивание
текста */
color: #000080; /* Темно-синий цвет текста */}
</style>
```

```
</head>
<body>
   <h1> Направления Всероссийского технологического марафона
TechRussia </h1>
   <p><abbrtitle="Виртуальная и дополненная реальность">VR&AR</
abbr> </p>
   <p> Интернет-вещей </p>
   <p> <abbrtitle="Беспилотные летательные аппараты"> БПЛА</
abbr> </p>
   <p> Космос </p>
   <p> Робототехника </p>
   <p> <abbrtitle="Автоматизированные системы управления">
АСУ</abbr> </p>
   <p> Промышленный дизайн </p>
   <p> Финансы </p>
   <p> Нейротехнологии </p>
</body>
</html>
```

Рис. 4.1. Вид HTML-страницы с использованием элемента ABBR

Тег ACRONYM - отмечает свой текст как акроним, т.е. произносимое слово, состоящее из аббревиатур. Тег ACRONYM обычно находит свое применение с использованием атрибута **title**, значением которого выставляется текст расшифровки аббревиатуры. Тогда, при наведении курсора на аббревиатуру, расшифровывающий текст будет отображаться браузером как примечание.

Тег CITE - отмечает заключенный в себе текст как цитату. Браузерами такой текст отображается курсивом.

Пример написания:

```
<p><cite> "Мы живем в эпоху, когда расстояние от самых
безумных фантазий до совершенно реальной действительности
сокращается с невероятной быстротой." </cite> Максим Горький
</p>
```

Тег CODE - отмечает заключенный в себе текст, как фрагмент компьютерного кода. Браузерами такой текст по умолчанию отображается моноширинным шрифтом (ширина всех символов одинакова).

Тег DFN - отмечает заключенный в себе текст, как определение (англ. definition). Браузеры отображают содержимое тега <dfn> с помощью курсивного начертания.

Тег EM - отмечает заключенный в себе текст, как выделенный. Браузерами по умолчанию отображается курсивом. Является предпочтительной альтернативой тега физического форматирования **I** (выделяет текст курсивом).

Тег KBD - отмечает заключенный в себе текст, как введенный пользователем с клавиатуры. Браузерами по умолчанию отображается моноширинным шрифтом.

Тег MARK – отмечает заключенный в себе текст, как выделенный. Браузерами по умолчанию выделяется желтым цветом.

Тег SAMP - отмечает заключенный в себе текст, как пример (пример отчета, выдаваемого программой, сценарием и т.п.). Браузерами по умолчанию отображается моноширинным шрифтом.

Тег STRONG - отмечает заключенный в себе текст, как сильно выделенный. Браузерами по умолчанию отображается жирным шрифтом. Является предпочтительной альтернативой тега физического форматирования **B** (выделяет текст жирным шрифтом).

Тег VAR - отмечает заключенный в себе текст, как экземпляр переменной или аргумента какой-либо программы. Браузерами по умолчанию отображается курсивом.

Из описания этих тегов видно, что некоторые из них приводят к одинаковому визуальному результату: выделение текста курсивом, например, можно осуществить тегами VAR, EM, DFN, CITE. Однако не только это является результатом их применения. Использование тегов структурного формати-

рования позволяет разделить и выделить в тексте логически обособленные фрагменты и кратко описать их суть. Это в свою очередь помогает понять структуру документа и более широко использовать возможности каскадных таблиц стилей.

Для каждого тега структурной разметки обязательно задание начального и конечного тегов.

Пример использования:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
       Использование тегов структурного форматирования текста
    </title>
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
      bgcolor=#112232>
    <h1>
      Использование тегов структурного форматирования текста
    </h1>
    Обычная строка
    <p> <cite> Строка, отформатированная тегом CITE</cite></p>
    <p> <code> Строка, отформатированная тегом CODE</code></p>
    <p> <dfn> Строка, отформатированная тегом DFN</dfn></p>
    <p> <em> Строка, отформатированная тегом EM </em></p>
    <p> <kbd> Строка, отформатированная тегом KBD </kbd></p>
    <p> <mark> Строка, отформатированная тегом MARK</mark></p>
    <p> <samp> Строка, отформатированная тегом SAMP </samp></p>
    <p>
      <strong> Строка, отформатированная тегом STRONG
      </strong></p>
    <p> <var> Строка, отформатированная тегом VAR </var></p>
  </body>
</HTML>
```

Все вышеуказанные элементы имеют следующие необязательные атрибуты, которые могут быть заданы, а могут и нет:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);

- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Теги **INS** и **DEL** используются для разметки вставленных и удаленных фрагментов текста документа, по отношению к другой версии документа. Эти два тега обособлены от других ввиду того, что они могут выступать и качестве элементов уровня блока, и в качестве встроенных элементов. Но не теми и другими сразу. То есть теги INS и DEL не должны содержать тегов уровня блока, если они используются как встроенные.

Пример правильного использования:

```
<p> По последним сведениям в TechRussia примут участие
<del> 300 </del><ins> 500 </ins> лучших студентов и молодых
специалистов со всей России</P>
```

Текст, отмеченный как "удаленный", отображается браузером перечеркнутым. Тег DEL является предпочтительной альтернативой тега физического форматирования **S** (обозначает текст как перечеркнутый)

Текст, отмеченный как "вставленный", отображается браузерами подчеркнутым.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Использование тегов INS и DEL </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
    bgcolor=#112232>
      <h1>Использование теговINS и DEL</h1>
      Обычная строка
      <p><del> Строка, отформатированная тегом DEL </del></p>
      <p><ins> Строка, отформатированная тегом INS </ins></p>
      <p> По последним сведениям в TechRussia примут участие
<DEL> 300 </DEL><INS> 500 </INS> лучших студентов и молодых
специалистов со всей России </P>
   </body>
</HTML>
```

Рис. 4.3. Демонстрация использования тегов INS и DEL

Эти теги имеют следующие необязательные атрибуты:

- **cite=URL** – указывает месторасположение документа с информацией, объясняющей причину изменения документа;
- **date time** – указывает дату изменения документа.

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента)
- **style** – встроенная информация о стиле
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события

Тег PRE

Наличие этого тега говорит браузеру о том, что заключенный в нем текст является отформатированным текстом. Такой текст отображается браузерами в том же виде, в каком он присутствует в HTML-тексте документа.

Например:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Использование тега PRE </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <h1> Использование тега PRE </h1>
   <pre>==========##############################==########=========###
##################################==###########
####==##############################==########==######===########
#####################################
####==#####=======###=======###=======###==########==##==###
##==###=====###=======##==###======#
####==####=#####%==##==######==##==####*==###==######==###==###
##==##==#######==#########==#########==
######==###==========#==#########==######==##========#####==###
##==###=====###=====####==###=======
####==###==#########==#########==######==##==######==###==###
##==########==#######==###==##==######==
####==####====%===##====##===##==######==##==#######==###===##
====##===##===#===##===##==##===##====
##############==########%=%###################################==
#########==#######=%############==####
   </pre>
   </body>
</HTML>
```

На экране это будет выглядеть так:

При использовании тега PRE задание начального и конечного тегов является обязательным.

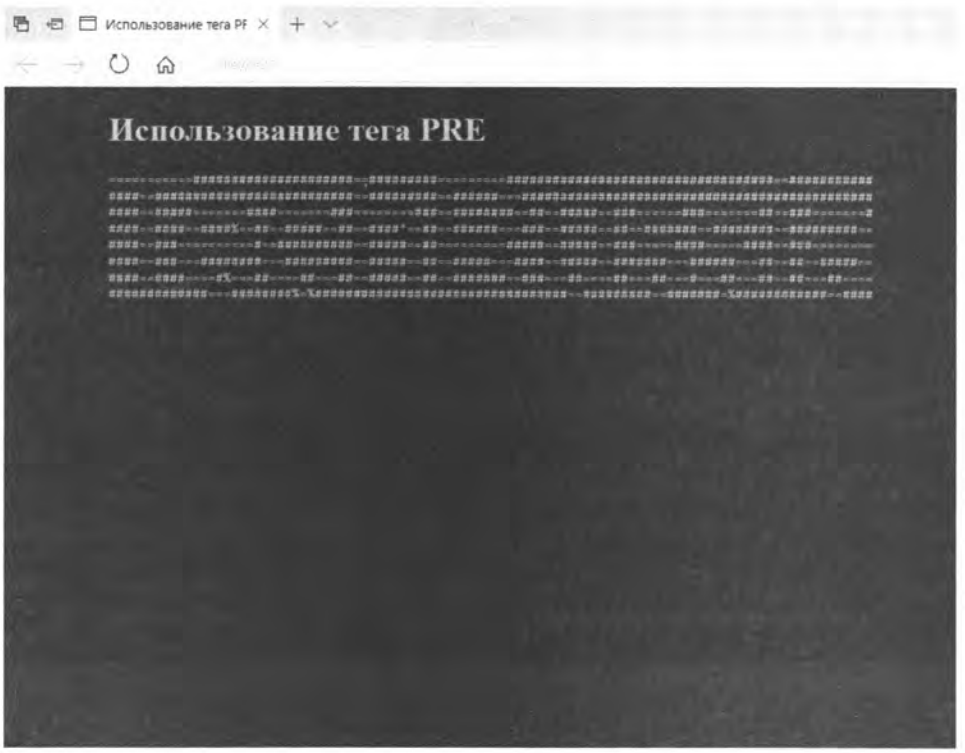


Рис. 4.4. Применение тега PRE

4.5. Какие существуют теги физического форматирования документа?

Теги физического форматирования непосредственно определяют вид текста в окне браузера. По указанным выше причинам многие из них считаются нежелательными, но все они по-прежнему поддерживаются браузерами.

1. **Тег B** – отображает заключенный в себе текст жирным шрифтом. Вместо него рекомендуется использовать тег структурного форматирования STRONG.
2. **Тег I** – отображает заключенный в себе текст курсивом. Вместо него рекомендуется использовать теги структурного форматирования VAR, EM, DFN, CITE, которые помимо выделения текста курсивом указывают его смысловое значение.

3. Тег **RUBY** – отображает аннотацию сверху или снизу от заданного текста. В браузерах, которые не поддерживают тег RUBY, возможно использование тега RP с аналогичным функционалом. Для добавления аннотации используется тег RT.
4. **Тег S** - отображает заключенный в себе текст перечеркнутым.
5. **Тег SMALL** - отображает заключенный в себе текст шрифтом, меньшим, чем шрифт окружающего текста.
6. **Тег SUB** – выводит заключенный в себе текст в нижнем индексе.
7. **Тег SUP** – выводит заключенный в себе текст в верхнем индексе.

Теги SUR и SUB обычно используются для вывода математических и химических формул.

Пример использования:

```
<!DOCTYPE html>
<html>
  <HEAD>
    <title>
Использование тегов физического форматирования текста
</title>
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
    <h1> Использование тегов физического форматирования
текста </h1>
    Обычная строка
    <p><b> Строка, отформатированная тегом B </b></p>
    <p><i> Строка, отформатированная тегом I </i></p>
    <p><ruby> Строка, отформатированная тегом RUBY с <rt>
аннотацией (тег RT)</rt></ruby></p>
    <p><s> Строка, отформатированная тегом S </s></p>
    <p><small> Строка, отформатированная тегом SMALL </
small></p>
    Обычный текст <sup> Текст в верхнем индексе </sup></p>
    Обычный текст <sub> Текст в нижнем индексе </sub></p>
  </body>
</html>
```

На экране монитора это будет выглядеть так:

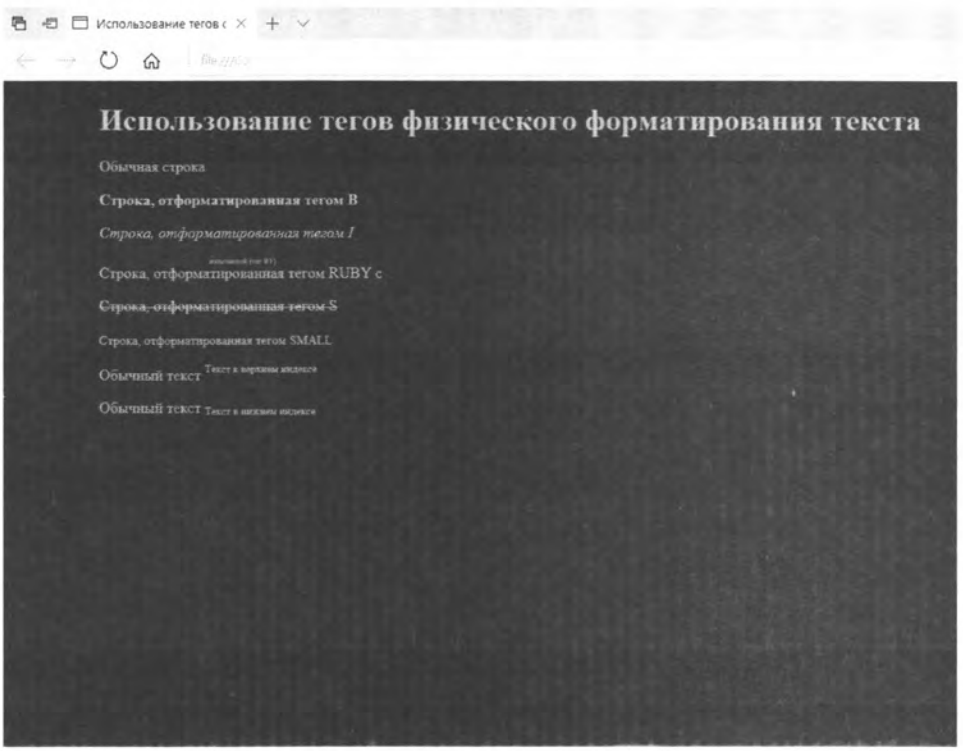


Рис. 4.5. Использование тегов физического форматирования текста

HTML5

Глава 5.

Работа с текстом (продолжение)

В этой главе вы узнаете:

- *Как обозначить цитату?*
- *Как выделять строки и абзацы?*
- *А что можно делать с заголовками?*
- *Как вставить горизонтальные линии?*
- *Как скрывать текст?*

5.1. Как обозначить цитату?

Теги Q и BLOCKQUOTE задуманы как теги, выделяющие в цитаты, приводимые в тексте документа:

Q – короткие цитаты (нет разбиения на абзацы);

BLOCKQUOTE – длинные цитаты.

Тег BLOCKQUOTE, в отличие от тега **Q**, является элементом уровня блока. На практике его использование приводит к тому, что заключенный в нем блок текста выделяется из контекста отступом слева (всего текста) и пустыми строками сверху и снизу.

Тег **Q** выводит заключенный в себе текст в кавычках.

Атрибуты:

- **cite** – необязательный атрибут, указывающий месторасположение источника, из которого взята цитата. В качестве значения этого атрибута указывается URL-адрес источника.

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);

- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Текст, заключенный в содержимом тега BLOCKQUOTE, сверху и снизу отделяется пустыми строками, а также он выводится с небольшим отступом влево (при направленности текста слева направо). Допускается вложение тегов BLOCKQUOTE друг в друга, чем достигаются дополнительные возможности форматирования документов (управление отступами).

Пример использования:

```
<!DOCTYPE html>
<html>
      <head>
            <title> Использование тега BLOCKQUOTE</title>
      </head>
      <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
            <h1>Использование тега BLOCKQUOTE </h1>
            <p> Что такое "TechRussia"?</p>
            <blockquote> Технологический марафон
TechRussia – это событие, направленное на популяризацию
мира технологий. <p> Будущие и начинающие инженеры,
программисты, руководители проектов и отраслевые
специалисты соберутся на одной площадке, чтобы раскрыть
свой потенциал, познакомиться с новейшими научными
разработками и встретиться с профессионалами мира
технологий. </p>
            <cite>http://techrussia.org/</cite>
</blockquote>
</body>
</html>
```

Рис. 5.1. Внешний вид документа с использованием тега BLOCKQUOTE

5.2. Как выделять строки и абзацы?

Текст документа обычно разбивается авторами на абзацы, текст в которых имеет законченное смысловое содержание. Это делает прочтение и понимание документа более удобным. Обычно в текстовых редакторах разделение абзацев производится переходом на следующую строку с помощью клавиши "Enter". На работу браузеров этот метод не оказывает никакого влияния. Это значит, что при отображении текста, при отсутствии в документе каких-либо указаний со стороны HTML-разработчика, браузер сам будет осуществлять переходы на следующую строку из соображений экономного использования пространства своего окна. При изменении размеров окна соответственно будет меняться длина строк и, следовательно, визуальное представление. Все это может привести к большим неудобствам. Чтобы их избежать в языке HTML предусмотрено разбиение текста на отдельные абзацы путем заключения их в содержимое тегов **P**.

Тег P

HTML располагает своими средствами для разбиения на абзацы. Для этой функции используется тег уровня блока **P**. Текст, находящийся между начальным и конечным тегами, воспринимается браузерами как абзац. При отображении абзацы сверху и снизу выделяются пустой строкой.

Указание начального тега обязательно, конечный тег может не задаваться: при этом абзацем будет считаться все, что расположено после начального тега.

Атрибуты, все необязательные:

- **align** – выравнивание, может принимать следующие значения:
  - *left*: выравнивание по левому краю (используется по умолчанию);
  - *right*: выравнивание по правому краю
  - *center*: выравнивание по центру;
  - *justify*: выравнивание по ширине.
- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

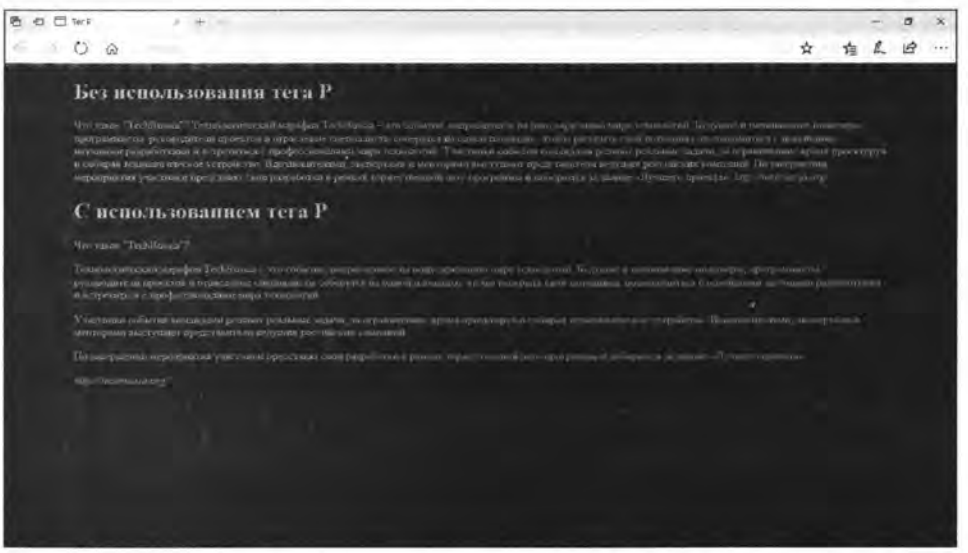


Рис. 5.2. Пример использования тега P для разбиения на абзацы. Сверху текст представлен без разбиения на абзацы, а снизу – с разбиением

В теге **P** могут содержаться теги уровня блока, включая и сам тег **P**.

Тег BR

Место переноса строки в пределах абзаца, тем не менее, определяется браузером по-прежнему автоматически исходя из размера окна и размера шрифта. HTML предоставляет возможность принудительного переноса строки, независимого от настроек браузера. Элементом, осуществляющим принудительный перенос строки, является тег **BR**. Перенос строки, при этом, происходит сразу после места его задания.

Начальный тег обязателен, конечный тег запрещен.

Необязательные атрибуты:

- **clear** – указывает, где в окне браузера должна появиться перенесенная после использования элемента **BR** строка. Этот атрибут учитывает прикрепляемые объекты (таблицы, изображения и т.п.). Обсуждается в спецификации HTML. Принимаемые значения:
  - *none* – следующая строка текста отображается обычным образом, на ближайшем свободном снизу пространстве. Это значение используется по умолчанию;
  - *left* - следующая строка текста отображается на ближайшем свободном пространстве под прикрепленным у левого поля объектом;
  - *right* - следующая строка текста отображается на ближайшем свободном пространстве под прикрепленным у правого поля объектом;
  - *all* - следующая строка текста отображается на ближайшем свободном пространстве под прикрепленным у любого поля объектом.

Прикрепление объектов к полям документа осуществляется соответствующим заданием у них атрибута **align**. Тогда если объект прикреплен, например, к левому полю, то он обтекается текстом по правой своей стороне.

Например:

Пусть имеется исходный документ, в котором объект-изображение не прикреплен ни к какому краю документа (не определен атрибут **align**) и, соответственно, текстом не обтекается:

Рис. 5.3. Внешний вид документа с неприкрепленным изображением

```
<!DOCTYPE html>
<html>
   <head>
      <title> Исходный вариант </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <h2 align=center> Иван Смирнов <br> Капитан команды </h2></
h2>
   <IMG src="face1.jpg">
   Выпускник по специальности "Инноватика". Обладает 5-летним
практическим опытом руководства техническими проектами.
   </body>
</html>
```

Теперь организуем обтекание изображение текстом, прикрепив его к левому краю. Для этого изображению укажем атрибут **align=left**, то есть в HTML-коде документа строку:

```
<IMG src="face1.jpg">
```

(вставка картинки face1.jpg в документ) заменим на строку:

```
<IMG src="face1.jpg" align=left>
```

В результате чего получим следующее:

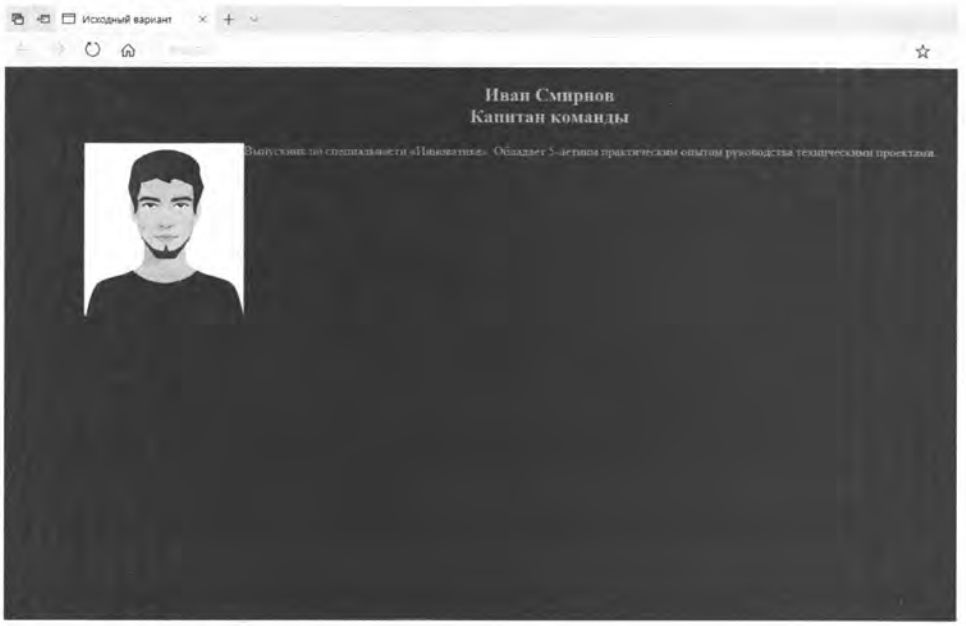


Рис. 5.4. Внешний вид документа с прикрепленным изображением

Применение тега BR с заданным атрибутом **clear**, определяет, будет ли расположенный за элементом текст и дальше обтекать объект, или будет выводиться под ним. В приведенном примере, использование тега BR без указания атрибута **clear** (что равносильно указанию **clear=none**) не прерывает обтекания текстом, а только осуществляет перенос на следующую строчку. Вставим тег
 после предложения “Выпускник по специальности "Инноватика"“.

Использование тега BR с атрибутом **clear=left** прервет обтекание текстом и продолжит вывод текста сразу под прикрепленным объектом.

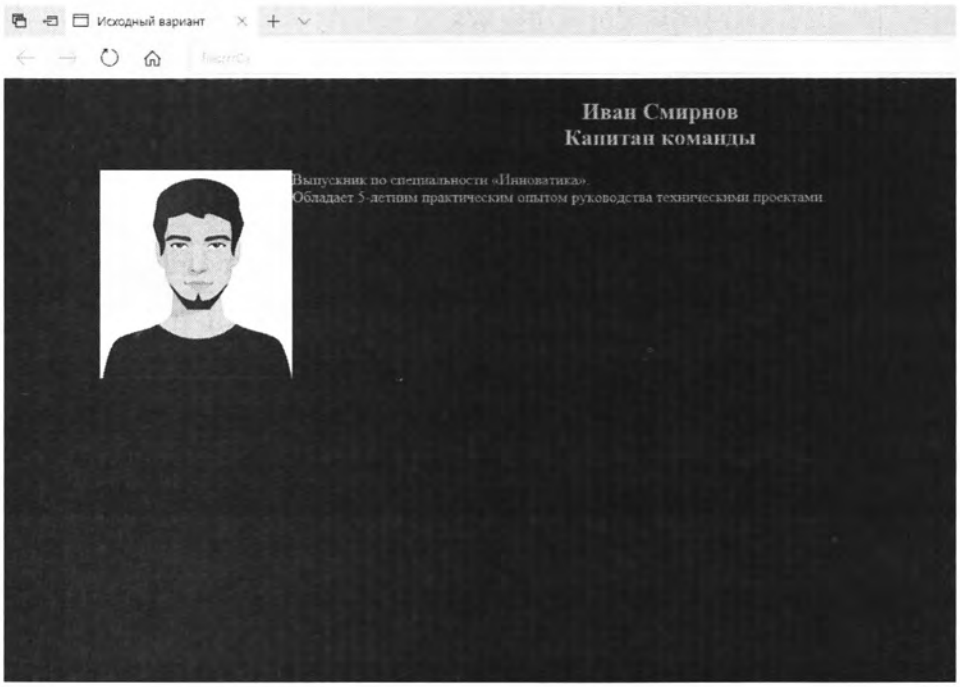


Рис. 5.5. Демонстрация применения тега BR

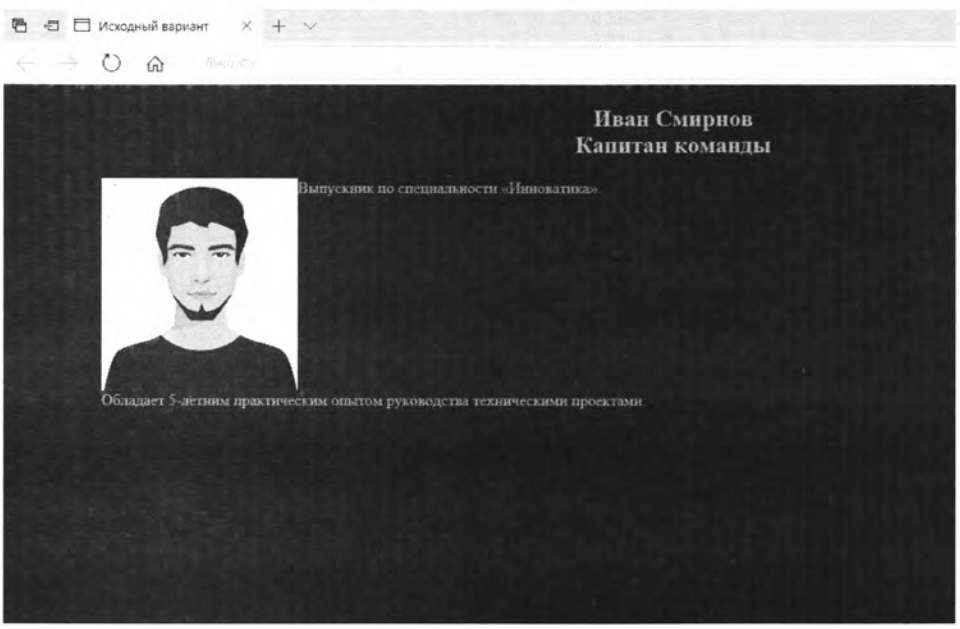


Рис. 5.6. Демонстрация применения тега BR с атрибутом clear

Тег WBR

При изменении ширины окна браузера, текст автоматически переносится на следующие строки, не выходя за границы окна. Для того, чтобы указать браузеру место, где допускается делать перенос строки, используется тег WBR. Следует помнить, что при переносе слов на следующую строку, браузер не добавляет символ дефиса.

Открывающий тег WBR является обязательным, закрывающий – нет.

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Тег WBR </title>
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
    <h1> Использование тега WBR </h1>
    Что такое "TechRussia"? <wbr>
Технологический марафон TechRussia - это событие, направленное
на популяризацию мира технологий.
Будущие и начи<wbr>нающие инженеры, программисты,
руково<wbr>дители проектов и отрас<wbr>левые специалисты
соберутся на одной площадке, чтобы раскрыть свой потенциал,
познако<wbr>миться с новейшими научными разработками и
встре<wbr>титься с профессионалами мира технологий.
Участники со<wbr>бытия командами ре<wbr>шают реальные
задачи, за огра<wbr>ниченное время проектируя и собирая
технологическое устройство.
Вдохновителями, экспертами и менто<wbr>рами выступают
предста<wbr>вители ведущих российских компаний.
По завершении мero<wbr>приятия участники представят свои
разработки в рамках торжественной шоу-<wbr>программы и
поборются за звание "Лучшего проекта".
  </body>
</html>
```

Рис. 5.7. Демонстрация применения тега WBR

5.3. А что можно делать с заголовками?

В языке HTML предусмотрены теги, определяющие содержащийся между их начальным и конечным тегами строку текста как заголовок. Всего определено шесть заголовков различного уровня. Каждому из них соответствует определенный размер: самым маленьким является заголовок шестого уровня, самым большим – заголовок первого уровня. HTML-элементами, соответствующими заголовкам с первого по шестой уровень, являются теги H1, H2, H3, H4, H5, H6. Все заголовки отображаются жирным шрифтом.

Теги H1, H2, H3, H4, H5, H6 являются блокообразующими элементами. Это, в частности, означает, что они не могут использоваться внутри текста для выделения отдельных его фрагментов, так как строка, содержащая заголовок, может содержать только заголовок. Весь остальной текст располагается выше и ниже его.

Заголовки сверху и снизу выделяются пустыми строками.

При использовании тегов H1, H2, H3, H4, H5, H6 задание начального тега является обязательным. Конечный тег может не указываться, тогда заголовком будет считаться все, что расположено после начального тега.

Необязательные атрибуты:

- **align** – выравнивание, может принимать следующие значения:
  - *left*: выравнивание по левому краю (используется по умолчанию);
  - *right*: выравнивание по правому краю
  - *center*: выравнивание по центру;
  - *justify*: выравнивание по ширине.
- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Пример:

```
<!DOCTYPE html>
<html>
      <head>
            <title> Заголовки в HTML-документах </title>
      </head>
      <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
            Обычный текст
            Обычный текст
            <h6> Заголовок 6-го уровня </h6>
            <h5> Заголовок 5-го уровня </h5>
            <h4> Заголовок 4-го уровня </h4>
            <h3> Заголовок 3-го уровня </h3>
            <h2> Заголовок 2-го уровня </h2>
            <h1> Заголовок 1-го уровня </h1>
      </body>
</html>
```

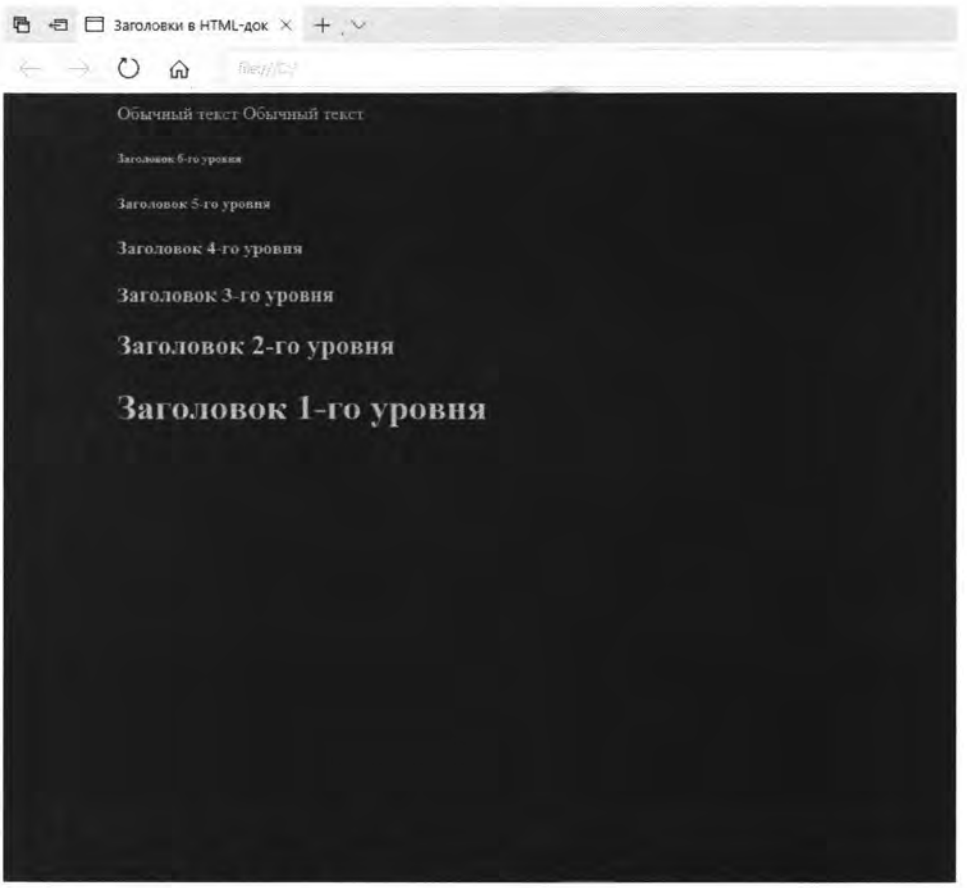


Рис. 5.8. Примеры заголовков различного размера

5.4. Как вставить горизонтальные линии?

В HTML реализована возможность включения простейших горизонтальных линий средствами самого HTML. Можно еще вставлять линии в виде изображения, но это требует дополнительной памяти и дополнительного файла с самим изображением. Это является оправданным при вставке каких-то особых, разноцветных линий.

Простейшую линию можно нарисовать с помощью тега HR.

Визуальные параметры этой линии задаются с помощью атрибутов тега HR:

- **align** – выравнивание, может принимать следующие значения:
  - *left*: выравнивание по левому краю;

  - *right*: выравнивание по правому краю;
  - *center*: выравнивание по центру (используется по умолчанию);
- **size** – этим атрибутом задается толщина линии в пикселах;
- **width** – длина линии в пикселах или процентах от ширины окна браузера
- **noshade** – логический атрибут, задание которого отменяет рельефность линии. По умолчанию линия прорисовывается рельефной.
- **color** – через этот атрибут указывается цвет линии в виде ключевого слова, обозначающего цвет, или в виде RGB-кода.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Горизонтальные линии </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h2 align=center> Примеры горизонтальных линий </h2>
      HR
      <hr>
      HR noshade
      <hr noshade> HR size=20
      <hr size=20> HR size=20 noshade
      <hr size=20 noshade> HR size=45
      <hr size=45>   HR width=50%
      <hr width=50%> HR width=50% color=black
      <hr width=50% color=black>
      HR width=50% color=black size=10
      <hr width=50% color=black size=10>
      HR width=50% color=black size=10 align=left
      <hr width=50% color=black size=10 align=left>
   </body>
</html>
```

5.5. Как скрывать текст?

С появлением HTML5, в спецификацию был введен тег DETAILS, который позволяет скрывать и отображать информацию по требованию пользователя. По умолчанию текст, расположенный под тегом DETAILS, скрыт, одна-

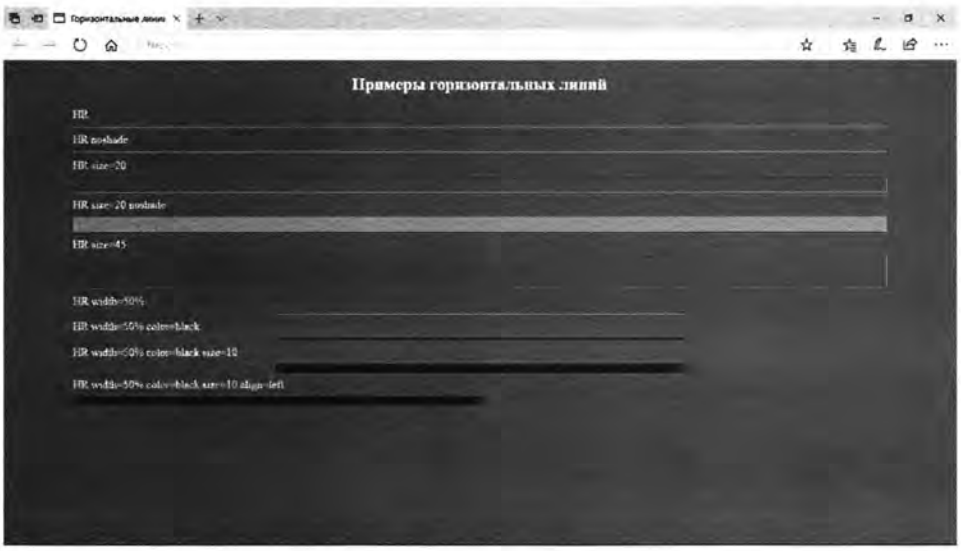


Рис. 5.9. Примеры горизонтальных линий, прорисовываемых тегом HR

ко использование атрибута open позволит изменить настройки "по умолчанию" и отображать текст.

Браузер Internet Explorer (Microsoft Edge) не поддерживает данный тег.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Тег DETAILS </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <p>Что такое "TechRussia"?</p>
      <details> Технологический марафон TechRussia - это
событие, направленное на популяризацию мира технологий.
<p>Будущие и начинающие инженеры, программисты, руководители
проектов и отраслевые специалисты соберутся на одной площадке,
чтобы раскрыть свой потенциал, познакомиться с новейшими
научными разработками и встретиться с профессионалами мира
технологий. </p>
      </details>
   </body>
</html>
```

Рис. 5.10. Скрытый текст в окне браузере Mozilla Firefox

Чтобы указать заголовок для текста, скрытого в теге DETAILS, можно использовать тег SUMMARY. Тогда вместо слова "Подробнее" на рис. 5.10., будет отображен текст, заданный в теге SUMMARY.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Тег DETAILS </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <p>Что такое "TechRussia"?</p>
      <details><summary> Технологический марафон TechRussia
- это</summary> событие, направленное на популяризацию мира
технологий. <p>Будущие и начинающие инженеры, программисты,
руководители проектов и отраслевые специалисты соберутся на
одной площадке, чтобы раскрыть свой потенциал, познакомиться
с новейшими научными разработками и встретиться с
профессионалами мира технологий. </p>
      </details>
   </body>
</html>
```

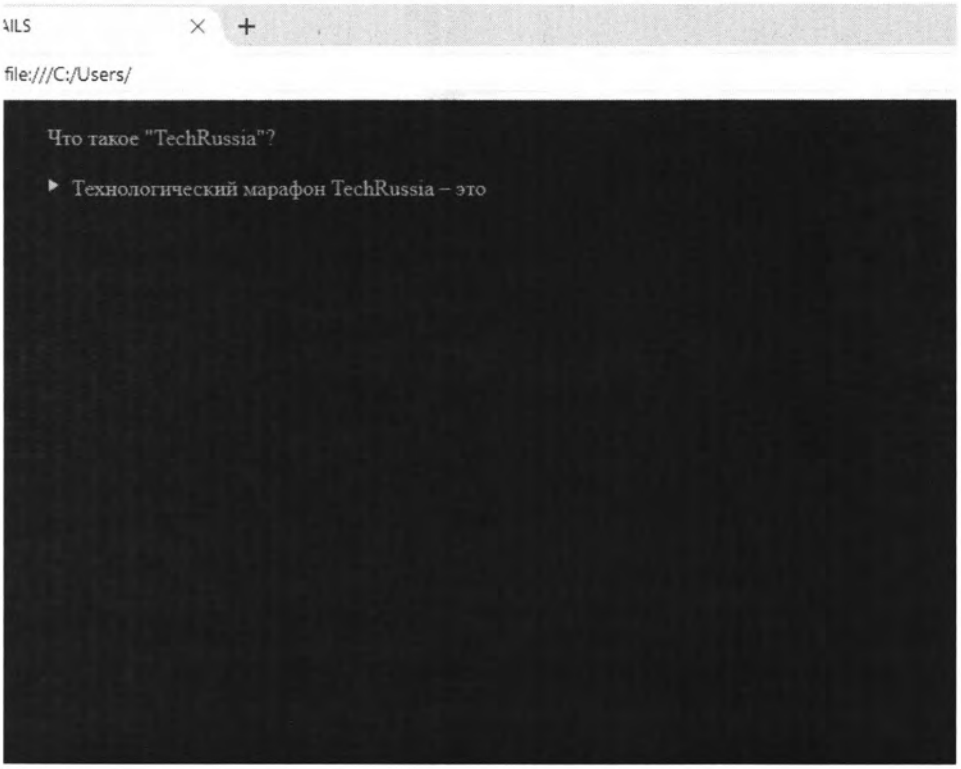


Рис. 5.11. Скрытый текст с заголовком в окне браузера Mozilla Firefox

HTML5

Глава 6.

Списки

В этой главе вы узнаете:

- *Какие бывают списки?*
- *Как создать неупорядоченный (маркированный) список?*
- *Как создать упорядоченный (нумерованный) список?*
- *Как создать список определений?*
- *Как создать список меню?*
- *Как комбинировать различные списки?*

6.1. Какие бывают списки?

Язык HTML обладает возможностями предоставления информации в виде списков. Список служит для добавления структуры в документ. Причем эта структура отображается визуально, например: список покупок, пошаговое описание каких-либо действий, толковый словарь и т.п.

В HTML различают:

- Неупорядоченные списки (список покупок);
- Упорядоченные списки (пошаговое описание, в котором каждый шаг пронумерован);
- Список определений (толковый словарь);
- Список меню.

В любом списке должен присутствовать хотя бы один элемент списка, иначе он будет проигнорирован.

Неупорядоченные списки создаются тегом UL, упорядоченные списки – тегом OL. Списки обоих типов состоят из последовательности элементов списка, которые задаются тегом LI. Тег LI, а точнее его содержание (например, название покупки), является обособленной частью списка. Неупорядоченные списки отображаются браузерами как маркированные, а упорядоченные – как пронумерованные.

6.2. Как создать неупорядоченный (маркированный) список?

Начальный тег обязателен, конечный тег обязателен.

Атрибут (необязательный):

- **type** – задает информацию о виде используемых маркеров. Может принимать следующие значения:
  - *circle* – маркеры отображаются в виде не закрашенных кружков;
  - *disc* - маркеры отображаются в виде закрашенных кружков;
  - *square* - маркеры отображаются в виде закрашенных квадратов.

  Примечание: если не задать значение атрибуту type, браузер будет использовать маркеры по своему усмотрению (обычно закрашенные кружки).

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Список, заключенный в теге UL, отделяется от контекста пустыми строками сверху (перед началом списка) и снизу (после последнего элемента списка. Вся информация, заключенная в теге UL, отображается со сдвигом вправо.

Каждый элемент списка (это относится и к упорядоченному списку) представляет собой HTML-тег LI. Открывающий тег LI обязателен, закрывающий не обязателен и обычно не указывается. В этом случае, элементом списка считается весь текст, расположенный до следующего открывающего

тега LI (или другого какого-либо элемента, например, заголовка) или до закрывающего тега UL (OL).

Пример:

```
<!DOCTYPE HTML>
<html>
   <head>
      <title> Неупорядоченный (маркированный) список </title>
      <meta charset="utf-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h1> Пример неупорядоченного (маркированного) списка </
h1>
      <ulid="napravleniya_list">
      <h3> Направления Всероссийского технологического марафона
TechRussia: </h3>
      <li> Виртуальная и дополненная реальность
      <li> Интернет-вещей
      <li> Беспилотные летательные аппараты
      <li> Космические технологии
      <li> Робототехника
```

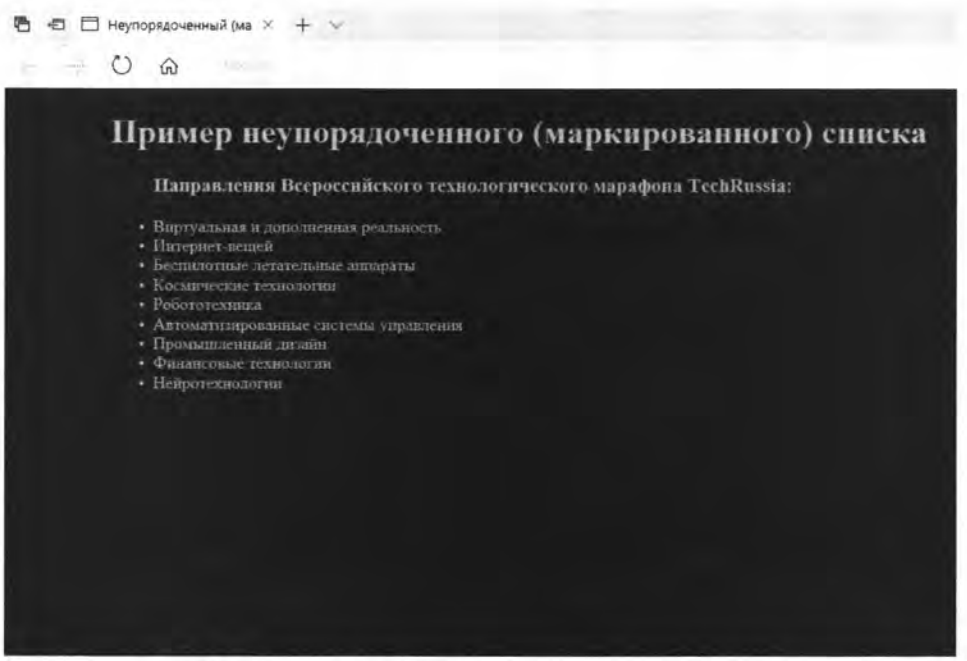


Рис. 6.1. Пример документа с неупорядоченным списком

```
      <li> Автоматизированные системы управления
      <li> Промышленный дизайн
      <li> Финансовые технологии
      <li> Нейротехнологии
      </ul>
   </body>
</html>
```

Для тега LI определены такие же атрибуты, задаваемые в любом месте, что и для тега UL, и еще два необязательных атрибута **value** и **type**, используемые при работе с упорядоченными списками. Визуально функции тега LI сводятся к отображению маркера в неупорядоченных списках или нумерации в упорядоченных списках.

Допускается использование вложенных списков. При отображении вложенных неупорядоченных списков браузеры по умолчанию используют различную маркировку списков разного уровня.

Например:

```
<!DOCTYPE HTML>
<html>
   <head>
      <title> Вложенный список </title>
      <meta charset="utf-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <h1> Пример вложенного списка </h1>
   <h2> Лучшие технологические события и их направления: </h2>
      <ul>
         <li> Всероссийский технологический марафон TechRussia:
         <ul>
            <li> Виртуальная и дополненная реальность
            <li> Интернет-вещей
            <li> Беспилотные летательные аппараты
            <li> Космические технологии
            <li> Робототехника
            <li> Автоматизированные системы управления
            <li> Промышленный дизайн
            <li> Финансовые технологии
            <li> Нейротехнологии
         </ul>
         <li> Всероссийский хакатон hackRussia:</li>
         <ul>
```

```
            <li> Социальные сервисы
            <li> Экология
            <li> Геймификация
            <li> Медицина и спорт
            <li> BigData
            <li> Образование и культура
            <li> Туризм
            <li> Sharing economy
        </ul>
    </ul>
  </body>
</html>
```

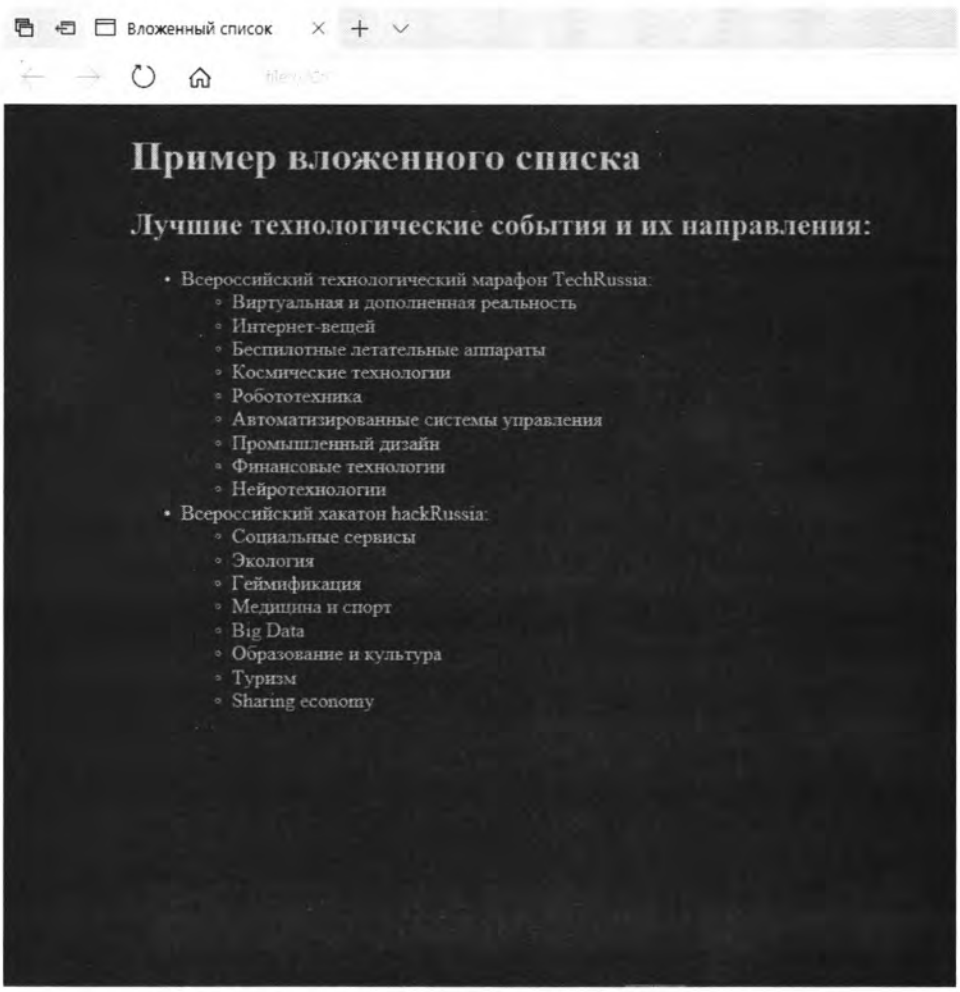


Рис. 6.2. Демонстрация вложенных неупорядоченных списков

С помощью атрибута TYPE можно непосредственно указывать вид пульки (маркера). Например:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример использования атрибута TYPE </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h2> Пример использования атрибута TYPE </h2>
      <ultype=circle>
         <li> Маркер - незакрашенный кружок
         <li> Маркер - незакрашенный кружок
         <li> Маркер - незакрашенный кружок
         <li> Маркер - незакрашенный кружок
      </ul>
      <ultype=square>
         <li> Маркер - закрашенный квадратик
         <li> Маркер - закрашенный квадратик
         <li> Маркер - закрашенный квадратик
         <li> Маркер - закрашенный квадратик
      </ul>
      <ul>
         <li> Маркер - закрашенный кружок
         <li> Маркер - закрашенный кружок
         <li> Маркер - закрашенный кружок
         <li> Маркер - закрашенный кружок
      </ul>
   </body>
</html>
```

Для повышения привлекательности документа можно создавать списки с нестандартными маркерами, например: звездочками, шариками и т.п. Для этого, вместо тега LI для каждого тега списка используются тег IMG - вставки изображения маркера и тег BR – перехода на новую строку.

Пример:

```
<!DOCTYPE HTML>
<html>
   <head>
      <title> Неупорядоченный (маркированный) список </title>
      <meta charset="utf-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
```

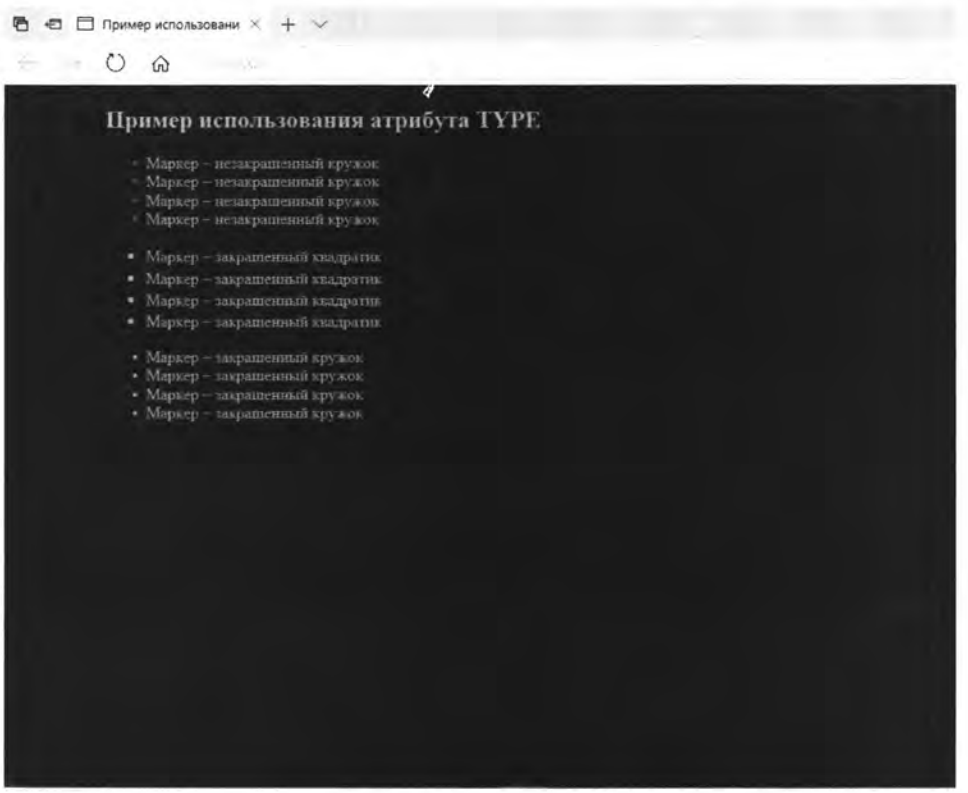


Рис. 6.3. Пример использования атрибута type для задания внешнего вида маркеров неупорядоченного списка

```
    <h1> Пример нестандартных маркеров </h1>
    <ul>
    <h3 align=center>Направления Всероссийского
технологического марафона TechRussia: </h3>
    <img src="marker.png"> Виртуальная и дополненная
реальность <br>
    <img src="marker.png"> Интернет-вещей <br>
    <img src="marker.png"> Беспилотные летательные аппараты
<br>
    <img src="marker.png"> Космические технологии <br>
    <img src="marker.png"> Робототехника <br>
    <img src="marker.png"> Автоматизированные системы
управления <br>
    <img src="marker.png"> Промышленный дизайн <br>
    <img src="marker.png"> Финансовые технологии <br>
    <img src="marker.png"> Нейротехнологии <br>
```

```
    </ul>
  </body>
</html>
```

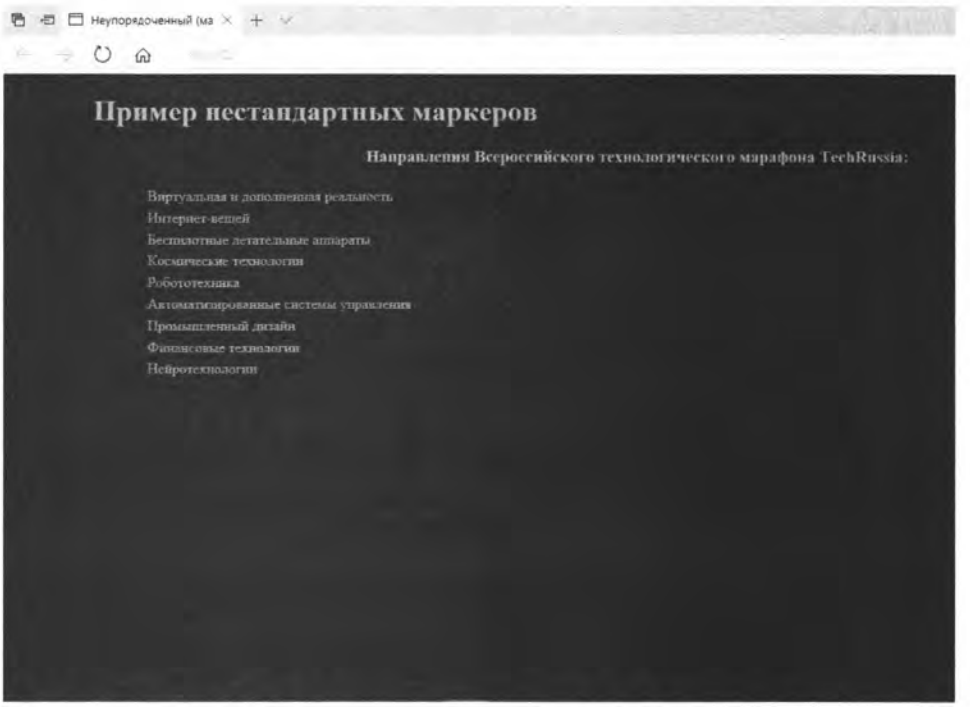


Рис. 6.4. Пример списка с маркерами в виде шестеренок

6.3. Как создать упорядоченный (нумерованный) список?

С помощью тега OL в HTML-документах реализуются упорядоченные (пронумерованные) списки. Так же, как и в случае неупорядоченных списков, элементом упорядоченного списка является тег LI со своим содержимым.

Указание начального и конечного тегов является обязательным.

Атрибуты (все необязательные):

- **type** – указывает вид нумерации элементов упорядоченного списка. Осуждается в спецификации HTML. Может принимать следующие значения:

- *type =1* – задает нумерацию арабскими цифрами(используется браузерами по умолчанию);
- *type=A* – задает нумерацию прописными латинскими буквами;
- *type=a* – задает нумерацию строчными латинскими буквами;
- *type=I* – задает нумерацию большими римскими цифрами;
- *type=i* – задает нумерацию маленькими латинскими цифрами.

- **start** – задает начальный номер первого элемента в упорядоченном списке. В качестве значения может принимать только натуральное число, независимо от вида нумерации, т.к. задает начальный номер, а не начальное значение в нумерации. Например, номер значения "C" в нумерации прописными латинскими буквами равен 3. По умолчанию значение атрибута **start** равно 1.

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Тип нумерации и значение номера тега HTML позволяет менять непосредственно при задании элемента списка LI. В этом случае используется вышеописанный атрибут **type** и атрибут **value**, задаваемые для тега LI. Задание значения атрибута **value** для элемента списка LI позволяет изменить номер элемента списка. При этом изменится нумерация всех последующих элементов списка. Принимает в качестве значений натуральные числа, как и в случае с атрибутом **start** тега OL.

Пример:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title> Упорядоченный список </title>
```

```
    <meta charset="utf-8">
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
    <h1> Пример упорядоченного списка </h1>
    <h3> Направления всероссийского технологического марафона
TechRussia:</h3>
    <ol>
      <li> Виртуальная и дополненная реальность
      <li> Интернет-вещей
      <li> Беспилотные летательные аппараты
      <li> Космические технологии
      <li> Робототехника
      <li> Автоматизированные системы управления
      <li> Промышленный дизайн
      <li> Финансовые технологии
      <li> Нейротехнологии
    </ol>
  </body>
</html>
```

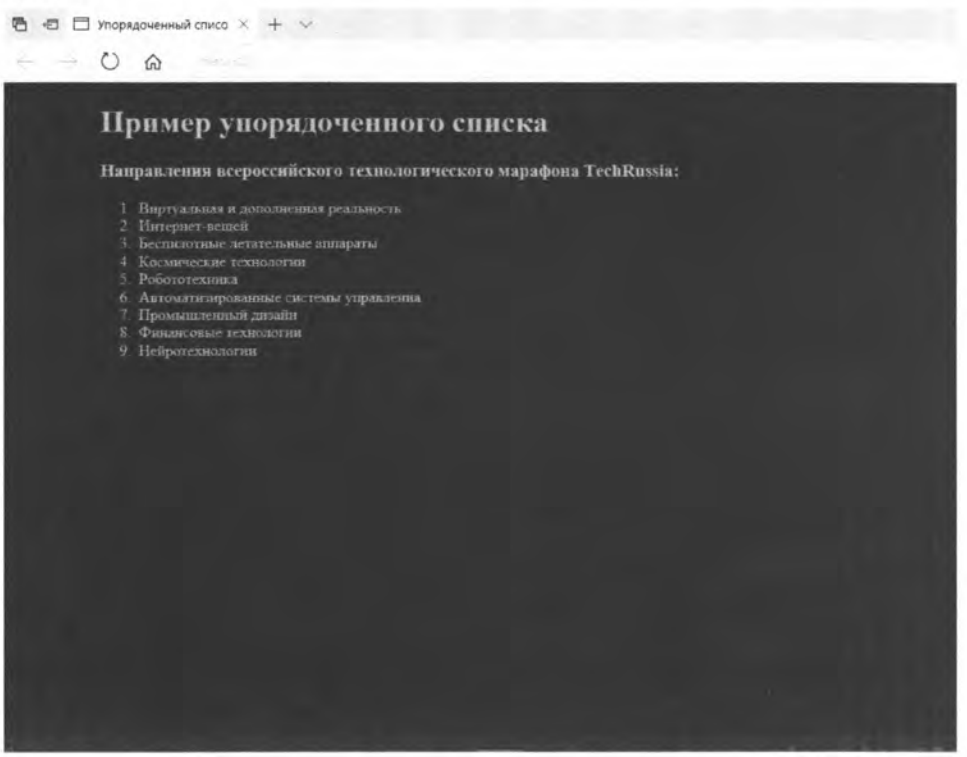


Рис. 6.5. Пример документа с упорядоченным списком

Аналогично неупорядоченным спискам для вложенных упорядоченных списков автоматически происходит смена вида нумерации. Например:

```
<!DOCTYPE HTML>
<html>
   <head>
      <title> Упорядоченный список </title>
      <metacharset="utf-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h2> Лучшие технологические события и их направления:</h2>
      <oltype=I>
      <li><b> Всероссийский технологический марафон TechRussia:</b>
         <ol>
            <li> Виртуальная и дополненная реальность
            <li> Интернет-вещей
            <br> ........................
            <livalue=8> Финансовые технологии
```

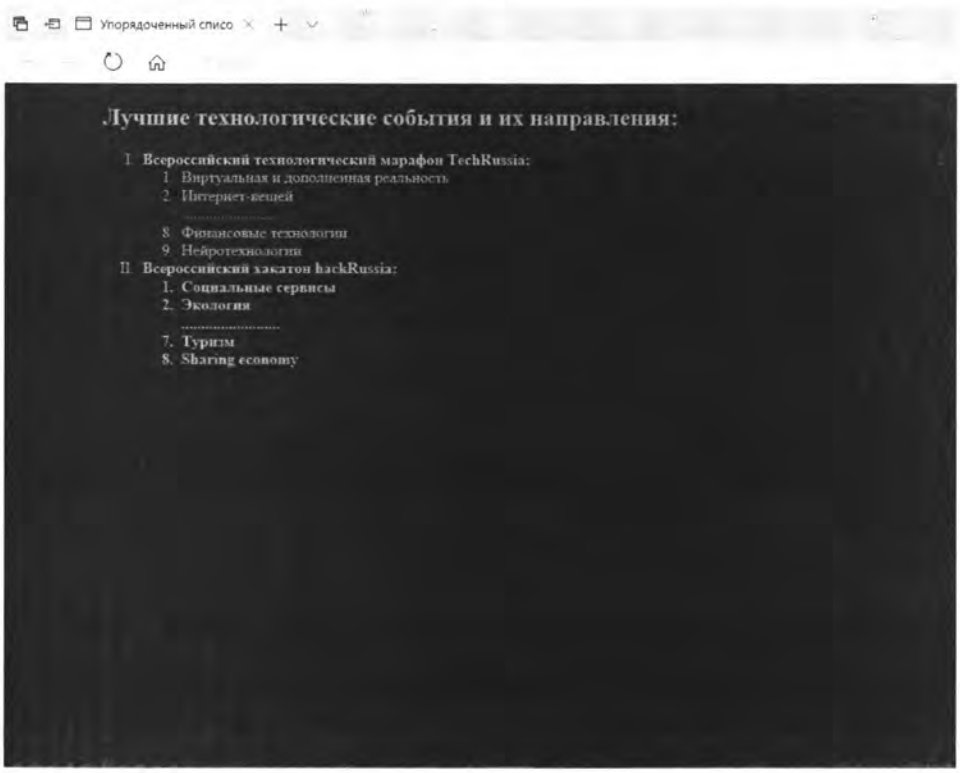


Рис. 6.6. Документ с вложенными упорядоченными списками

```
            <li> Нейротехнологии
          </ol>
          <li><b> Всероссийский хакатон hackRussia:</li><b>
          <ol>
            <li> Социальные сервисы
            <li> Экология
            <br>.........................
            <li value=7>Туризм
            <li> Sharing economy
          </ul>
        </ul>
    </body>
</html>
```

В этом примере также проиллюстрировано использование атрибута **value** тега LI. С ним связана маленькая тонкость: если используется нумерация не арабскими цифрами, а, например, прописными латинскими буквами, то в качестве значения атрибута **value** по прежнему указывается арабское число, соответствующее порядковому номеру буквы в алфавите.

6.4. Как создать список определений?

Списки определений являются особым видом списков, которые представляют содержащуюся в себе информацию в виде словарных статей: определяемый термин и его определение, отображенное с отступом. В отличие от других списков, каждый элемент такого списка состоит из двух частей:

- определяемого термина, задаваемого с помощью тега DT (Definition Term);
- текста с его определением, задаваемого с помощью тега DD (Definition Description).

Сам список определений заключается в содержимом тега-контейнера DL (Definition List).

Теги DL, DT и DD имеют только стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);

- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Задание начального и конечного тега DL является обязательным. Для тегов DT и DD начальный тег обязателен, а конечный тег можно опускать (что обычно и делается).

Недопустимо использование тегов уровня блока (например, элементы заголовков H1...H6 или элемент абзаца **P**) в содержимом тега DT. В содержимом тега DD их использование разрешается. Это, кроме всего прочего, говорит о возможности создания вложенных списков определений.

Пример использования списка определений:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример списка определений </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h2 align=center> Математический справочник </h2>
      <dl>
         <dt><b> Делитель нуля </b>
         <dd> Это ненулевой элемент кольца или полугруппы с
нулем, произведение которого на некоторый ненулевой элемент
равно нулю. В некоммутативном случае различают левые и правые
делители нуля.
         <dt><b> Делийская задача </b>
         <dd> Задача об удвоении куба. Заключается в построении
циркулем и линейкой стороны куба, объем которого вдвое больше
объема данного куба. Название "Делийская задача" связано с
преданием, по которому жители острова Делос, чтобы избежать
чумы, должны были исполнить повеление дельфийского оракула:
удвоить объем жертвенника, не изменяя при этом его кубической
формы.
         <dt><b> Дискретная математика </b>
         <dd> Область математики, занимающаяся изучением свойств
дискретных структур, которые возникают как внутри математики,
так и в ее приложениях. К числу таких структур могут быть
отнесены, например, конечные группы, конечные графы, а также
некоторые математические модели преобразователей информации,
конечные автоматы, машины Тьюринга и т.п.
```

```
        <dt><b> Дискретное программирование </b>
        <dd> Раздел математического программирования,
посвященный нахождению экстремумов функций, заданных на
конечных множествах. Задачи дискретного программирования
нетривиальны в том смысле, что число допустимых решений в
реальных задачах настолько велико, что их полный перебор
практически нереализуем.
     <dl>
  </body>
</html>
```

Рис. 6.7. Пример списка определений

6.5. Как создать список меню?

С помощью тега MENU в HTML-документах реализуются списки меню. Так же, как и в случае упорядоченных и неупорядоченных списков, элементом списка меню является тег LI со своим содержимым.

Указание начального и конечного тегов является обязательным.

С помощью атрибута **type** можно задать тип меню. Атрибут **type** может принимать следующие значения:

- **context** – контекстное меню;

- **toolbar** – меню в виде панели инструментов;
- **list** – меню-список.

Атрибут **type** пока не поддерживается браузерами.

Пример:

```
<!DOCTYPE HTML>
<html>
   <head>
      <title> Список меню </title>
      <meta charset="utf-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h1> Список меню </h1>
      <menu>
      <h3 align=center> Направления Всероссийского
технологического марафона TechRussia: </h3>
      <li> Виртуальная и дополненная реальность </li>
      <li> Интернет-вещей </li>
      <li> Беспилотные летательные аппараты </li>
      <li> Космические технологии </li>
      <li> Робототехника </li>
```

Рис. 6.8. Демонстрация списков меню

```
      <li> Автоматизированные системы управления </li>
      <li> Промышленный дизайн </li>
      <li> Финансовые технологии </li>
      <li> Нейротехнологии </li>
      </menu>
   </body>
</html>
```

6.6. Как комбинировать различные списки?

Возможности языка HTML позволяют комбинировать списки различного типа друг с другом. Вкладывать их друг в друга. Рассмотрим эту возможность на примере:

```
<!DOCTYPE HTML>
<html>
   <head>
      <title> Комбинированный список </title>
      <meta charset="utf-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <ul>
      <h2> Лучшие технологические события и их направления:</
h2>
         <li><b> Всероссийский технологический марафон
TechRussia:</b>
         <ol>
            <li>VR & AR
            <dl>
            <dt> Виртуальная и дополненная реальность
            <dd> Примеры проектов: проекция модели оперируемой
области для хирурга, система для совместной разработки
архитектурного проекта в смешанной реальности, реабилитация
заключенных с помощью приложения для VR
            </dl>
            <li> Интернет-вещей
            <br> .....................
            <li value=8> Финансовые технологии
            <li> Нейротехнологии
         </ol>
```

```
        <li><b> Всероссийский хакатон hackRussia:</li><b>
        <ol>
           <li> Социальные сервисы
           <li> Экология
           <br>.....................
           <li value=7> Туризм
           <li> Sharing economy
        </ol>
     </ul>
  </body>
</html>
```

Рис. 6.9. Пример вложения друг в друга списков различного типа

HTML5

Глава 7.

Таблицы

В этой главе вы узнаете:

- *Как создать таблицу в HTML 5?*
- *Как добавить название таблицы?*
- *Как редактировать строки и ячейки таблицы?*
- *Что такое структурное форматирование таблиц?*
- *Как подсчитать количество столбцов?*
- *Как определить ширину таблицы?*
- *Как выровнять текст внутри ячеек?*
- *Как изменять границы таблицы?*

7.1. Как создать таблицу в HTML 5?

Одним из самых распространенных и эффективных способов представления информации в документах является использование таблиц. В HTML применение таблиц носит более общий характер. Они позволяют разработчикам упорядочивать информацию в документе: текст, изображения, объекты, формы, поля форм и т.п.

Это значит, что помимо построения табличек как таковых, теги таблиц используются как средство форматирования документа. Таблицы с неотображаемыми (невидимыми) границами долгое время использовались для верстки веб-страниц, позволяя разбивать документ на прямоугольные области, в каждую из которых может быть помещена своя информация. Подобный способ применения таблиц нашел воплощение на многих сайтах, пока ему на смену не пришел более современный способ верстки с помощью слоев.

В HTML-документе может содержаться любое количество таблиц, причем допускается вложение таблиц друг в друга. Каждая таблица является содержимым тега-контейнера **<TABLE>** и состоит, по крайней мере, из одной строки. Каждая строка задается тегом <TR> (Table Row), внутри которой осуществляется деление на ячейки. Каждая ячейка строки представляет собой тег <TH> (Table Header – ячейка заголовка) или тег <TD> (Table Data – ячейка с данными). Разница в использовании этих тегов заключается в различном визуальном представлении в заключенной в них информации:

- **Тег <TH>** – содержащаяся в нем информация, по умолчанию выводится полужирным шрифтом и выравнивается по центру ячейки (align=center; valign=middle);

- **Тег <TD>** – содержащаяся в нем информация по умолчанию выводится обычным шрифтом, выровненная по горизонтали влево (align=left) и по центру по вертикали (valign=middle).

Количество строк в таблице определяется количеством начальных тегов TR. Количество столбцов – максимальным количеством ячеек в одной строке среди всех строк таблицы. Остальным строкам браузер добавляет необходимое количество пустых ячеек. Пустые ячейки добавляются в строки справа для таблиц, имеющих направление слева направо, и слева для таблиц, направленных справа налево. Порядок столбцов (направление таблицы) задается атрибутом **dir** тега <TABLE>.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример простейшей таблицы </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no>
   <h2> Пример простейшей таблицы </h2>
   <table border WIDTH=700>
      <tr>
         <th> 1-ая ячейка 1-ой строки
         <th> 2-ая ячейка 1-ой строки
         <th> 3-ая ячейка 1-ой строки
      <tr>
         <td> 1-ая ячейка 2-ой строки
         <td> 2-ая ячейка 2-ой строки
      <tr>
         <td> 1-ая ячейка 3-ей строки
         <td> 2-ая ячейка 3-ей строки
         <td> 3-я ячейка 3-ей строки
      <tr>
         <td> 1-ая ячейка 4-ой строки
   </table>
   </body>
</html>
```

Таблица может иметь название, которое задается с помощью тега CAPTION. Об этом читайте далее.

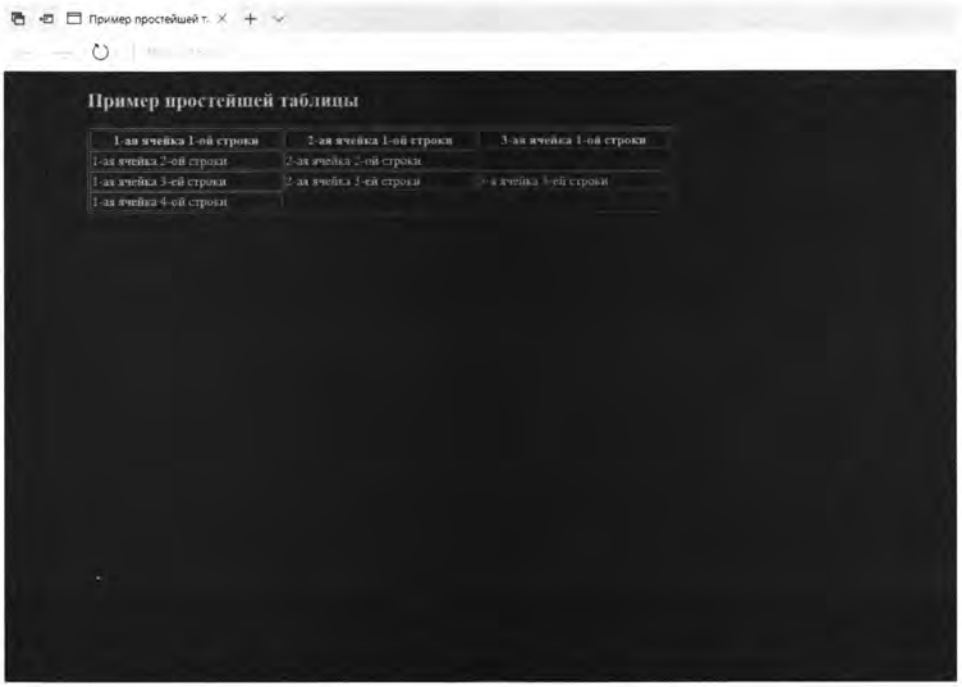


Рис. 7.1. Пример документа с простой таблицей

Тег TABLE

Тег TABLE является тегом-контейнером таблицы. Он задает таблицу. Все теги построения и форматирования таблицы должны заключаться в его содержимом.

Указание начального и конечного тегов является обязательным.

Все атрибуты тега TABLE являются необязательными. В спецификации HTML 5 они объявлены нежелательными, вместо атрибутов рекомендуется использовать стили каскадных таблиц CSS.

- **align** – задает параметры выравнивания самой таблицы в документе.

  Возможные значения:

  - *left*– таблица прикрепляется к левому краю документа;
  - *right* - таблица прикрепляется к правому краю документа;
  - *center* – таблица располагается в центре документа.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
   <title> Пример простейшей неприкрепленной таблицы </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no>
   <h2> Неприкрепленная таблица </h2>
   <table border>
      <tr>
         <th> 1-ая ячейка 1-ой строки
         <th> 2-ая ячейка 1-ой строки
      <tr>
         <td> 1-ая ячейка 2-ой строки
         <td> 2-ая ячейка 2-ой строки
      <tr>
         <td> 1-ая ячейка 3-ой строки
         <td> 2-ая ячейка 3-ой строки
   </table>
   </body>
</html>
```

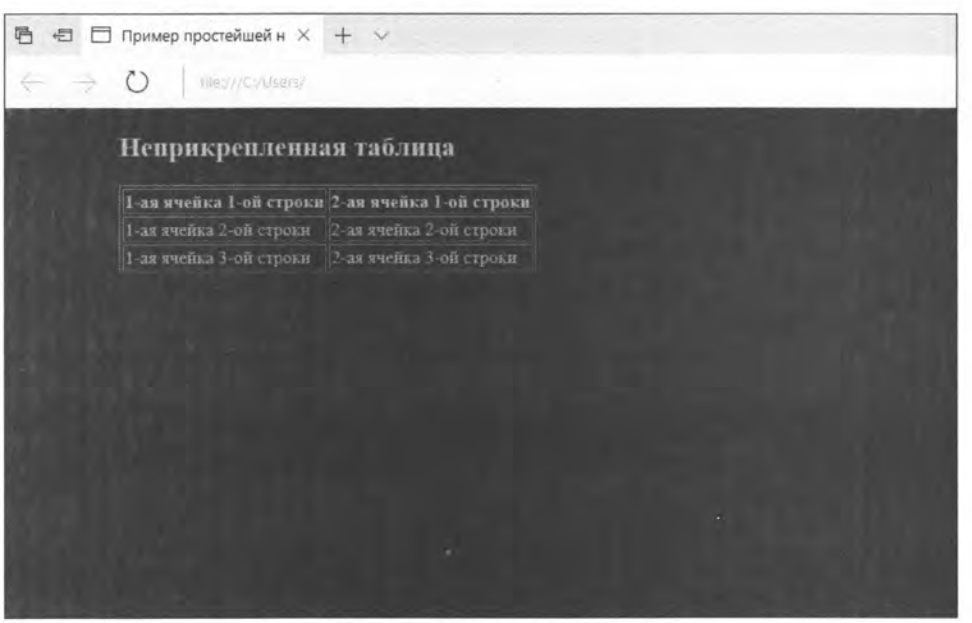


Рис. 7.2. Пример неприкрепленной таблицы

Теперь используем атрибут **align**, чтобы прикрепить таблицу к правому краю. Для этого в HTML-коде документа заменим строку

```
"<TABLE border>"
```

на

```
"<TABLE border align=right>"
```

Рис. 7.3. Пример таблицы, прикрепленной к правому краю

В результате этого документ в окне браузера примет следующий внешний вид:

- **width** – задает рекомендуемую ширину таблицы. Значение атрибута задается в пикселах или в процентах. При процентном задании допустимо указывать значения больше 100%, например, width=200%. Задание этого атрибута не гарантирует, что таблица будет иметь ширину указанный в значении атрибута, т.к. этот размер является не строго требуемым, а рекомендуемым.
- **border** – управляет видимостью и толщиной рамки вокруг таблицы и видимостью рамки вокруг каждой ячейки. В контексте использования атрибута border задание тега <TABLE> может быть в трех вариантах:
  - <TABLE> - атрибут **Border** не задан. В этом случае никакие рамки не прорисовываются. На экране отображается только содержащаяся в ячейках информация в отформатированном виде;

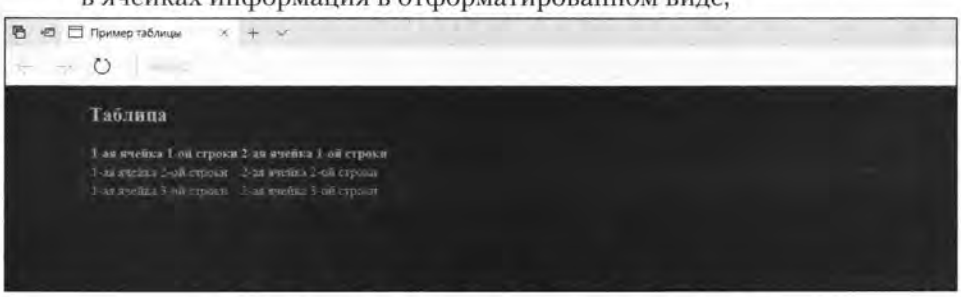


Рис. 7.4. Для тега <TABLE> не указан атрибут border

- <TABLE border> - атрибут **Border** задан, но не задано его значение;

Рис. 7.5. Для тега <TABLE> указан атрибут border, но не указано его значение

- <TABLE border=pixels> - атрибут **Border** задан и задано его значение в пикселах. В этом случае рамка вокруг таблицы отображается толщиной, указанной в значении, а рамки вокруг ячеек по-прежнему отображаются толщиной в 1 пиксел.

Рис. 7.6. Для тега <TABLE> указан атрибут border в значении 10 пикселов

Примечание: *Записи <TABLE> и <TABLE border=0> идентичны по своему действию.*

- **cellspacing** – указывает расстояние в пикселах между смежными ячейками (а точнее между их рамками) как по горизонтали, так и по вертикали. Значение данного атрибута по умолчанию принято равным 2 пикселам. При этом соседние ячейки будут разделены двумя границами (у каждой своя), между которыми и будут эти 2 пиксела. Отсюда следует, чтобы соседние ячейки имели одну общую границу, надо указать cellspasing=0. Рассмотрим его действие на примерах:

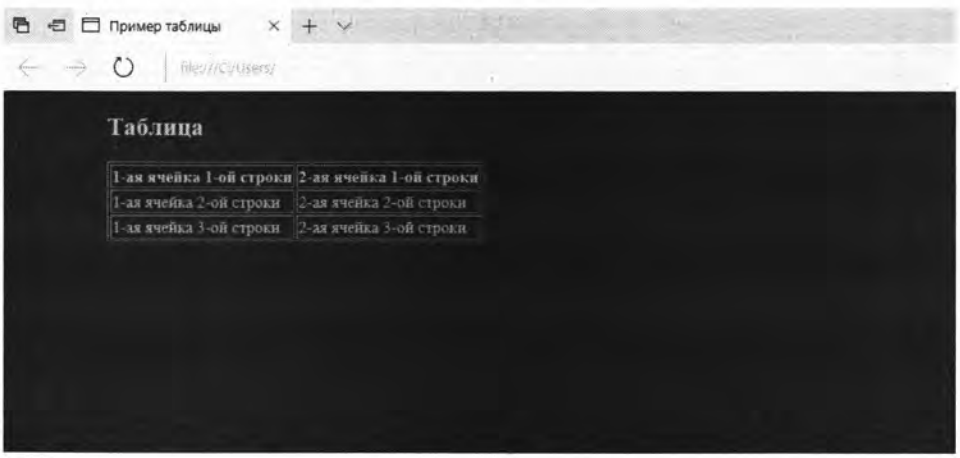


Рис. 7.7. Вид таблицы по умолчанию

Рис. 7.8. Вид таблицы, для которой указано cellspacing=0

Рис. 7.9. Вид таблицы, для которой указано cellspacing=20

- **cellpadding** – указывает размер отступа (по горизонтали и по вертикали) в пикселах между рамкой и содержимым ячейки. По умолчанию принят отступ в 1 пиксел. Если задать cellpadding =0, то содержимое некоторых ячеек будет соприкасаться с рамкой. Рассмотрим его действие на примерах:

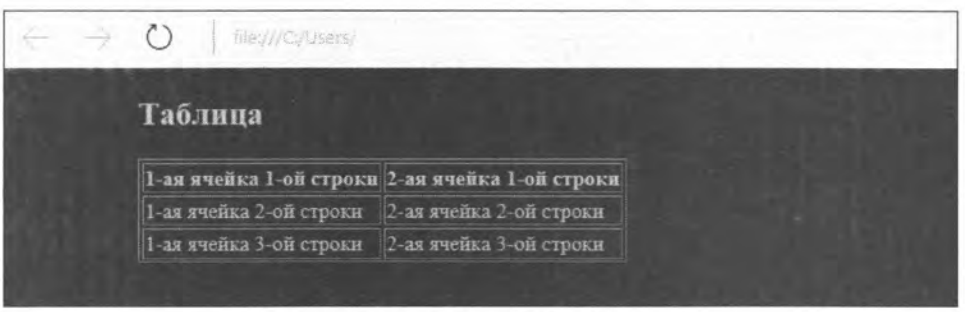


Рис. 7.10. Вид таблицы по умолчанию

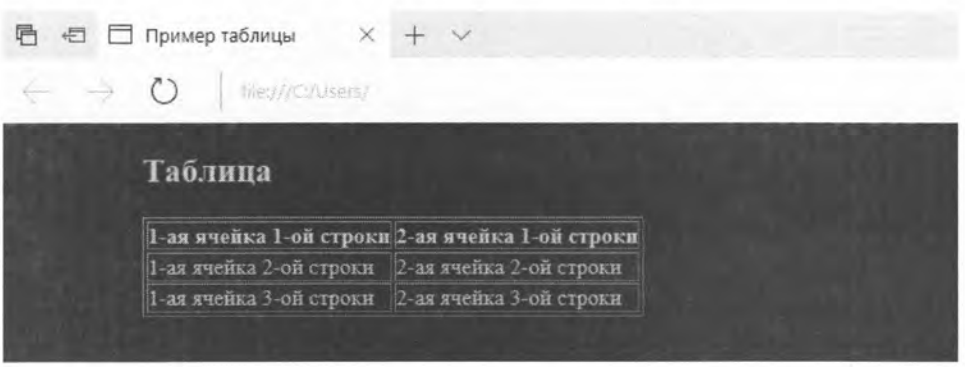


Рис. 7.11. Вид таблицы, для которой указано cellpadding=0

Примечание:

Наименьшая по размерам таблица имеет следующие атрибуты:

```
<TABLE border=0 cellspcing=0 cellpadding=0>.
```

- **bgcolor** – задает цвет фона на всем пространстве, занимаемом таблицей: на содержимом ячеек и на свободном пространстве между ними.

Стандартные необязательные атрибуты:

- id, class – идентификаторы в пределах документа;
- lang, dir – информация о языке и направленности текста;
- title – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- style – встроенная информация о стиле;
- onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup – внутренние события.

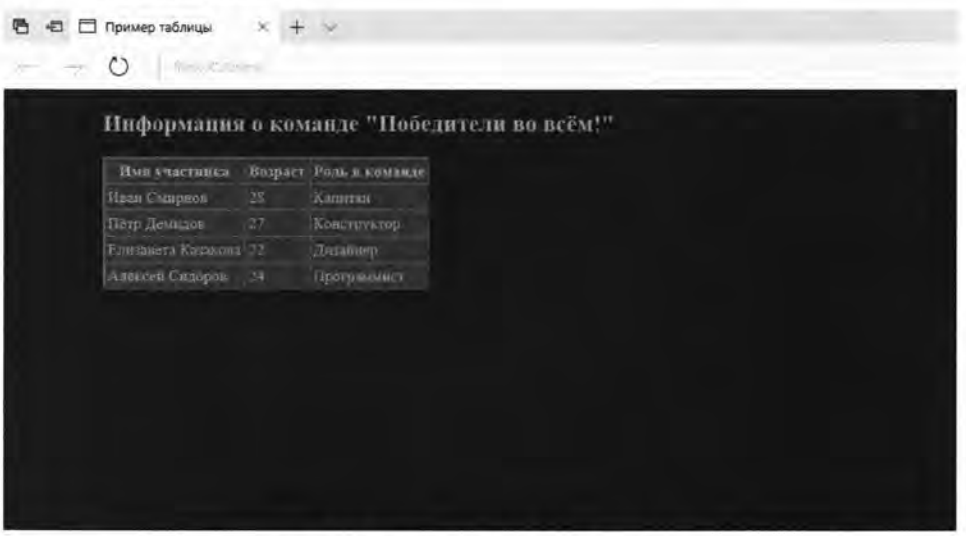


Рис. 7.13. Тег <TABLE> и его атрибуты

7.2. Как добавить название таблицы?

С помощью тега <CAPTION> можно задать название таблицы. При этом строка названия принадлежит таблице и при обтекании таблицы текстом,

обтекается вместе с ней. Одна таблица может иметь только одно название (если указано несколько, то они игнорируются)

Начальный и конечный теги <CAPTION> являются обязательными. Все атрибуты являются необязательными:

- **Align** – необязательный и нежелательный атрибут, указывающий месторасположение названия таблицы. Может принимать следующие значения:
  - *top* – название выводится сверху таблицы, выровненное по центру таблицы. Это значение принято по умолчанию.
  - *bottom* - название выводится снизу таблицы, выровненное по центру таблицы.
  - *left* - название выводится сверху таблицы, выровненное по левому краю таблицы.
  - *right* - название выводится сверху таблицы, выровненное по правому краю таблицы.

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

ВНИМАНИЕ! *Задание тега <CAPTION> может производиться только непосредственно после открывающего тега <TABLE>.*

Например:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример задания таблицы с названием </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0 bgcolor=#112232
scroll=no>
```

```
      <h2 align=left> Пример использования таблицы с названием </h2>
      <table align=center border=1 cellspacing=0 cellpadding=3
bgcolor=#3f544a>
      <caption align=center>
      <em><b> Информация о команде "Победители во всём!" </b></em>
      </caption>
      <tr>
         <th> Имя участника
         <th> Возраст
         <th> Роль в команде
      <tr>
         <td> Иван Смирнов
         <td> 28
         <td> Капитан
      <tr>
         <td> Пётр Демидов
         <td> 27
         <td> Конструктор
      <tr>
         <td> Елизавета Казакова
         <td> 22
         <td> Дизайнер
      <tr>
         <td> Алексей Сидоров
         <td> 24
         <td> Программист
   </table>
   </body>
</html>
```

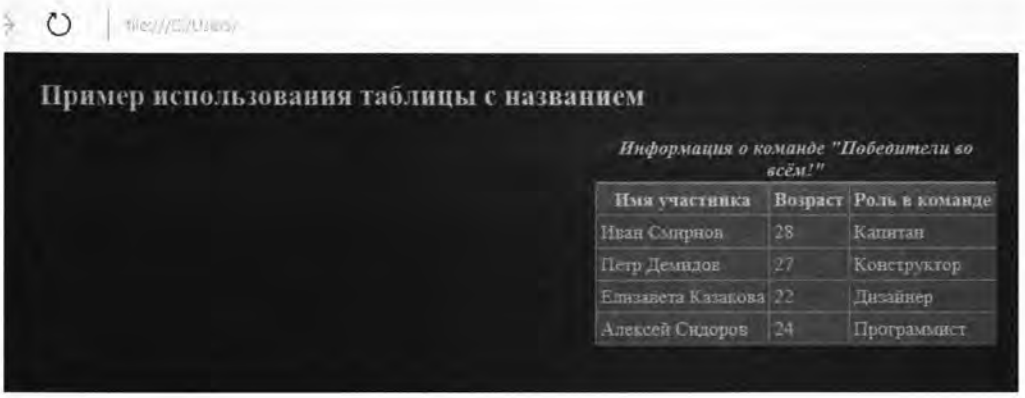


Рис. 7.14. Пример задания таблицы с названием

7.3. Как редактировать строки и ячейки таблицы?

Содержимым **тега <TR>** является строка таблицы, а точнее ячейки, расположенные в одной строке. Количество строк в таблице определяется количеством тегов <TR>. Начальный тег обязателен, конечный тег не обязателен и обычно не указывается.

Атрибуты (все необязательные):

- **align**, **char**, **charoff**, **valign** – атрибуты выравнивания. Их описание и применение приведены в разделе "7.7. Как выровнять текст внутри ячеек?"
- **bgcolor** – атрибут, задающий цвет фона ячеек, содержащей его строки.

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Теги <TH> и <TD> определяют ячейки таблицы. Содержимым этих тегов является информация, содержащаяся в отдельной ячейке.

У обоих начальный тег обязателен, конечный тег необязателен и обычно не указывается.

Теги <TH> и <TD> имеют следующие необязательные атрибуты:

- **headers** – атрибут, значением которого является список имен (id) ячеек, выступающих в роли заголовка (описания) для содержащей его ячейки. Этот атрибут обычно находит свое применение при невизуальном представлении таблицы. Например, при использовании синтезаторов речи, перед произнесением информации из ячейки будет произнесен поясняющий заголовок.

- **scope** – задает область таблицы, для которой информация из данной ячейки является заголовочной (описанием). Принимает одно из следующих значений:
  - *row* – говорит о том, что информация из данной ячейки является заголовком (описанием) для всех последующих ячеек строки;
  - *col* - говорит о том, что информация из данной ячейки является заголовком (описанием) для всех последующих ячеек столбца, содержащего данную ячейку;
  - *rowgroup* - говорит о том, что информация из данной ячейки является заголовком (описанием) для всех последующих строк таблицы;
  - *colgroup* - говорит о том, что информация из данной ячейки является заголовком (описанием) для всех последующих столбцов таблицы.

- **abbr** – атрибут, предоставляющий браузеру сокращенную форму содержимого ячейки. В качестве своего значения принимает текст, заключенный в кавычки.
- **rowspan** – атрибут, отвечающий за объединение соседних строк в рамках одного столбца (т.е. соседних ячеек в столбце) В качестве своего значения принимает натуральные числа. По умолчанию установлен в значении, равном единице.
- **colspan** - атрибут, отвечающий за объединение соседних столбцов в рамках одной строки (т.е. соседних ячеек в строке). В качестве своего значения принимает натуральные числа. По умолчанию установлен в значении, равном единице.
- **nowrap** – атрибут, задание которого запрещает разбивать содержимое ячейки на несколько строк. Является нежелательным атрибутом ввиду предпочтительного применения каскадных таблиц стиля.
- **width** – нежелательный атрибут, указывающий браузеру рекомендуемую ширину ячейки в пикселах.
- **height** - нежелательный атрибут, указывающий браузеру рекомендуемую высоту ячейки в пикселах.
- **align**, **char**, **charoff**, **valign** – атрибуты выравнивания. Их описание и применение приведены в разделе "7.7. Как выровнять текст внутри ячеек?"
- **bgcolor** – атрибут, задающий цвет фона внутри ячейки. Указывается название цвета, или соответствующее ему значение в RGB-формате.

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Пример задания тегов TH и TD:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример таблицы </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no>
      <h2 align=center> Пример таблицы </h2>
      <table align=left border cellspacing=0 cellpadding=5
bgcolor=#3f544a>
      <caption align=right>
      <em><b> Информация о команде "Победители во всём!" </b></
em>
      </caption>
      <tr bgcolor=blue>
         <th scope=col nowrap> Имя участника
         <th scope=col nowrap> Возраст
         <th id=role nowrap> Роль в команде
      <tr>
         <td> Иван Смирнов
         <td> 28
         <td headers=role> Капитан
      <tr>
         <td> Пётр Демидов
         <td> 27
         <td headers=role> Конструктор
      <tr>
         <td> Елизавета Казакова
         <td> 22
```

```
        <td headers=role> Дизайнер
      <tr>
        <td> Алексей Сидоров
        <td> 24
        <td headers=role> Программист
   </table>
   </body>
</html>
```

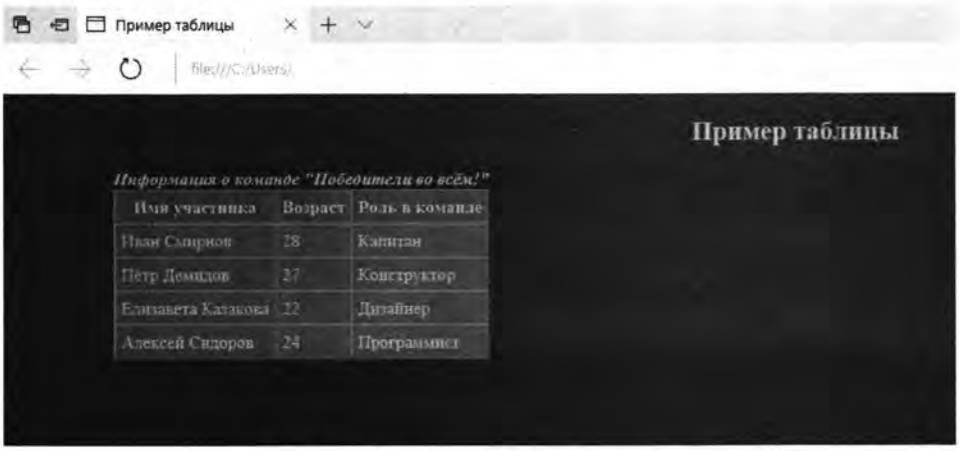


Рис. 7.15. Пример использования тегов таблицы

Объединение ячеек в примерах:

Исходная таблица:

```
   <table align=left border cellspacing=0 cellpadding=5
bgcolor=#3f544a>
      <caption align=right>
        <em><b> Название таблицы </b></em>
      </caption>
      <tr>
        <th> Ячейка 1
        <th> Ячейка 2
        <th> Ячейка 3
      <tr>
        <td> Ячейка 4
        <td> Ячейка 5
        <td> Ячейка 6
      <tr>
        <td> Ячейка 7
        <td> Ячейка 8
        <td> Ячейка 9
   </table>
```

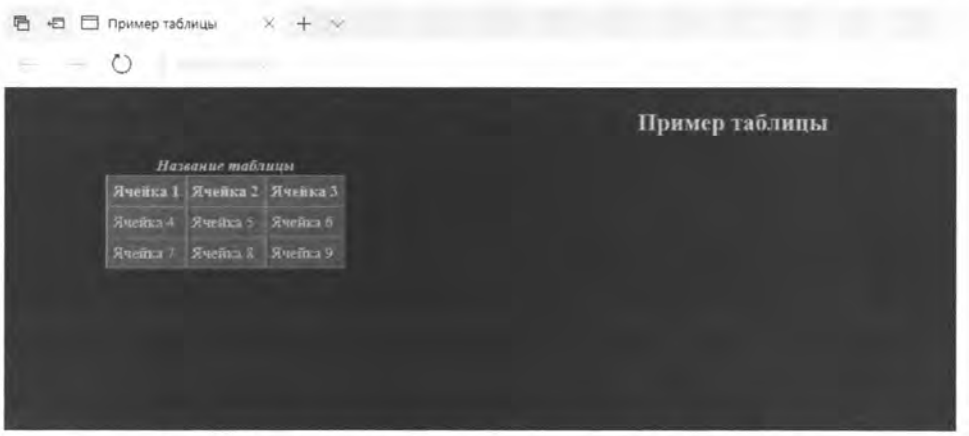


Рис. 7.16. Исходная таблица без каких-либо объединений

Объединим две первые ячейки в одну. Для этого надо указать атрибут **colspan=2** для первой ячейки и стереть вторую или третью ячейку:

```
  <table align=left border cellspacing=0 cellpadding=5
bgcolor=#3f544a>
     <caption align=right>
        <em><b> Название таблицы </b></em>
     </caption>
     <tr>
        <th colspan=2> Ячейка 1
        <th> Ячейка 3
     <tr>
        <td> Ячейка 4
        <td> Ячейка 5
        <td> Ячейка 6
     <tr>
        <td> Ячейка 7
        <td> Ячейка 8
        <td> Ячейка 9
  </table>
```

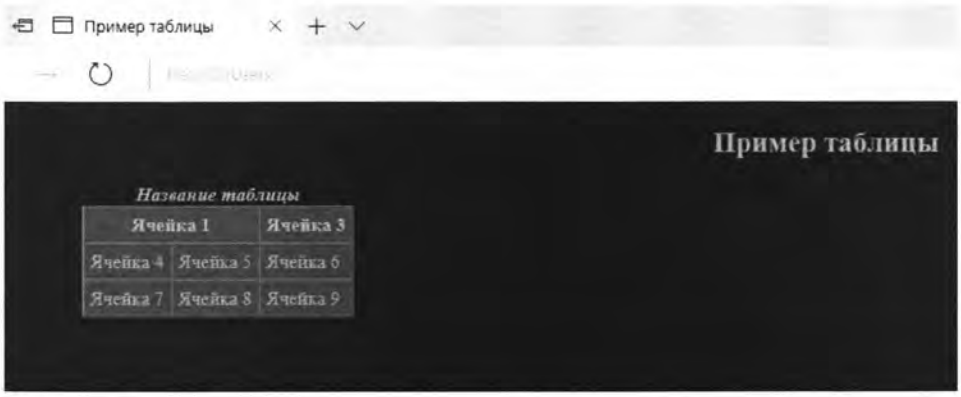


Рис. 7.17. Таблица с объединенными ячейками в одной строке

При объединении ячеек в сроке все последующие ячейки сдвигаются вправо. Поэтому если в рассматриваемой таблице не убрать вторую ячейку (или третью), то таблица примет следующий вид:

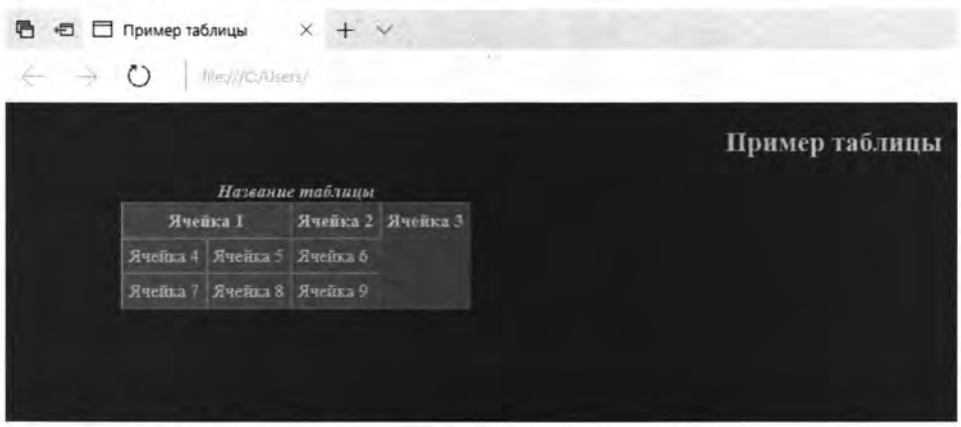


Рис. 7.18. Демонстрация сдвига следующих после объединения ячеек

Аналогичная ситуация с объединением соседних ячеек одного столбца:

```
<table align=left border cellspacing=0 cellpadding=5
bgcolor=#3f544a>
     <caption align=right>
        <em><b> Название таблицы </b></em>
     </caption>
     <tr>
        <th> Ячейка 1
        <th> Ячейка 2
```

```
        <th> Ячейка 3
      <tr>
        <td> Ячейка 4
        <td rowspan=2> Ячейка 5
        <td> Ячейка 6
      <tr>
        <td> Ячейка 7
        <td> Ячейка 9
  </table>
```

Рис. 7.19. Таблица с объединенными ячейками в одном столбце

Допускается одновременное объединение в одну ячейку соседних ячеек и столбца и строки. Например:

```
  <table align=left border cellspacing=0 cellpadding=5
bgcolor=#3f544a>
     <caption align=right>
       <em><b> Название таблицы </b></em>
     </caption>
     <tr>
       <th> Ячейка 1
       <th> Ячейка 2
       <th> Ячейка 3
     <tr>
       <td> Ячейка 4
       <td> Ячейка 6
     <tr>
       <td> Ячейка 9
  </table>
```

Рис. 7.20. Пример объединения в одну ячейку соседних ячеек и столбца и сроки

7.4. Что такое структурное форматирование таблиц?

Таблица в HTML может быть отструктурирована. Это значит, что ее строки и столбцы могут быть объединены в логические группы определенного структурного типа. Так же, как и в случае со структурным форматированием текста, этот раздел при первом ознакомлении с HTML может быть пропущен. Привнесение структуры практически не сказывается на внешнем виде таблицы. Оно бывает целесообразно для осуществление более быстрой загрузки таблицы. Также структурное разделение таблицы позволяет более эффективно обращаться к отдельным группам ячеек, что может быть особенно полезно при использовании каскадных таблиц стилей.

Группы строк

Для каждой таблицы можно указать, какие строки являются заголовком ("шапкой") таблицы, а какие телом таблицы. Причем в HTML различают верхние заголовки (верхние строки таблицы) и нижние заголовки (нижние строки таблицы). На данный момент практическая польза от такого разбиения заключается в более удобном задании свойств внешнего вида таблицы (свойств визуализации) и в предоставлении дополнительных возможностей использованию каскадных таблиц стиля.

Выделение строк в структурные группы осуществляется посредством тегов <THEAD>, <TFOOT> и <TBODY>:

- строки верхнего заголовка ограничиваются тегами <THEAD>
- строки нижнего заголовка ограничиваются тегами <TFOOT>
- строки тела таблицы ограничиваются тегами <TBODY>

В одной таблице может присутствовать несколько тегов-контейнеров <TBODY>, которые объединяют строки в несколько групп. Теги <THEAD> и <TBODY> в рамках одной таблицы должны встречаться единожды.

Задание тега <TFOOT> должно проводиться раньше по тексту HTML-документа, чем задание тега <TBODY>, хотя при отображении нижний заголовок располагается под телом таблицы.

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример таблицы </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no>
      <h2 align=center> Пример таблицы объединением строк в
структур ы</h2>
   <table align=left border cellspacing=0 cellpadding=5
bgcolor=#3f544a>
      <caption align=right>
         <em><b> Название таблицы </b></em>
      </caption>
      <thead>
         <tr>
            <th> Ячейка верхнего заголовка 1
            <th> Ячейка верхнего заголовка 2
            <th> Ячейка верхнего заголовка 3
         <tr>
            <th> Ячейка верхнего заголовка 4
            <th> Ячейка верхнего заголовка 5
            <th> Ячейка верхнего заголовка 6
      </thead>
      <tfoot>
         <tr>
            <th> Ячейка нижнего заголовка 1
            <th> Ячейка нижнего заголовка 2
            <th> Ячейка нижнего заголовка 3
      </tfoot>
         <tr>
```

```
            <td> Ячейка тела таблицы 1
            <td> Ячейка тела таблицы 2
            <td> Ячейка тела таблицы 3
         <tr>
            <td> Ячейка тела таблицы 4
            <td> Ячейка тела таблицы 5
            <td> Ячейка тела таблицы 6
   </table>
   </body>
</html>
```

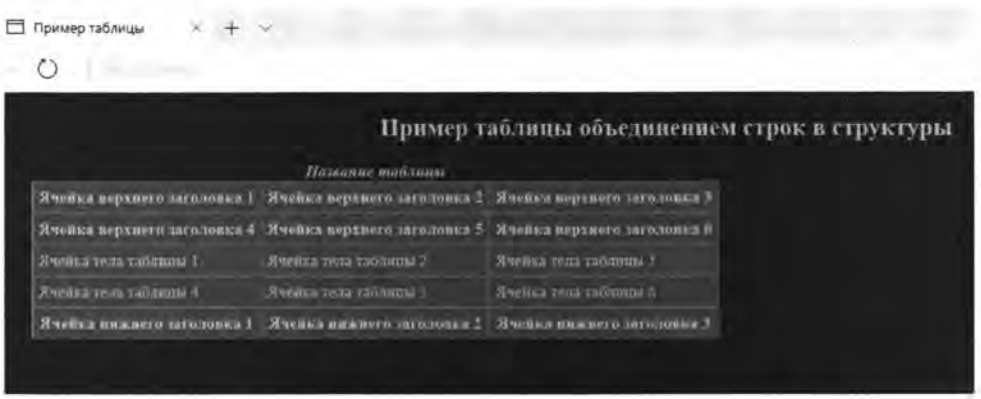


Рис. 7.21. Пример таблицы с объединением строк в структуры

Атрибуты тегов <THEAD>, <TFOOT> и <TBODY> (все необязательные):

- **align**, **char**, **charoff**, **valign** – атрибуты выравнивания. Их описание и применение приведены в разделе "7.7. Как выровнять текст внутри ячеек?"

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, onkeyup – внутренние события.

Начальный тег обязателен, конечный тег не обязателен: необходимость его задания определяется контекстом. Например, если в таблице предусматривается нижний заголовок и указан конечный тег <TFOOT>, то начальный тег <TBODY> можно не указывать.

Группы столбцов COLGROUP и COL

Если теги <THEAD>, <TFOOT> и <TBODY> структурируют таблицу на уровне строк, то тег <COLGROUP> – на уровне столбцов. Этот тег объединяет рядом расположенные столбцы в группы, к которым могут быть применены какие-либо действия (например, обращение к группе столбцов таблицы стилей, то есть задание им индивидуального внешнего вида). Тег <COL> напротив, позволяет задавать одни и те же характеристики для разных столбцов, не используя их структурной группировки.

Таблица может отображаться вся сразу, а может последовательно. В первом случае браузер ждет загрузки содержимого всех ячеек таблицы и только после этого отображает таблицу. Для убыстрения загрузки рекомендуется организовывать последовательное отображение таблицы. В этом случае сначала загружаются и отображаются границы таблицы (если они есть), затем по очереди загружается и отображается содержимое ячеек. При этом загрузка всей таблицы по времени не меняется, но зато значительно уменьшается время ожидания первого представления таблицы (первой отображенной информации), что делает ее загрузку гораздо привлекательней.

Для обеспечения последовательного отображения таблицы браузеру необходимо указать ее предварительные размеры. Для этого используются теги <COLGROUP> и <COL>.

Тег <COLGROUP>

Начальный тег <COLGROUP> является обязательным, конечный тег необязательным.

Атрибуты:

- **span** – необязательный атрибут, задает количество столбцов в группе. В качестве значений принимает только натуральные числа. Общий вид задания: span=N. В этом случае тег <COLGROUP> определяет группу, содержащую N столбцов. Если атрибут span не задан, то тег <COLGROUP> определяет группу только из одного столбца.

Пример задания:

```
<!DOCTYPE html>
```

```
<html>
   <head>
      <title> Пример таблицы </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no>
      <h2 align=center> Пример таблицы </h2>
   <table align=left border cellspacing=0 cellpadding=5
bgcolor=#3f544a>
      <caption align=right>
         <em><b> Название таблицы </b></em>
      </caption>
      <colgroup span=2>
         <tr>
            <th> Ячейка 1
            <th> Ячейка 2
         <tr>
            <th> Ячейка 3
            <th> Ячейка 4
   </table>
   </body>
</html>
```

Примечание: *Первый тег <COLGROUP> (если перед ним отсутствует тег <COL>) выделяет группу столбцов, начиная с нулевого. Каждый последующий тег COLGROUP группирует столбцы, начиная с первого столбца, не находящегося под влиянием предыдущего тег <COLGROUP> и тега <COL>.*

Поэтому, чтобы выделить группу, например, из пяти столбцов, начиная с десятого, надо сначала выделить первые десять (включая нулевой столбец). Здесь возможны два варианта:

Пример с использованием еще одного тега <COLGROUP>:

```
<table>
<caption> Название таблицы </caption>
<colgroup span=9>
<colgroup span=5>
<thead>
<tr>
<td> Ячейка 1 <td> Ячейка 2 <td> Ячейка 3
.....
</table>
```

Пример с использованием тега <COL>:

```
<table>
<caption> Название таблицы </caption>
<colspan=9>
<colgroup span=5>
<thead>
<tr>
<td> Ячейка 1<td> Ячейка 2<td> Ячейка 3
.....
</table>
```

- **width** – задает ширину каждого столбца в выделенной группе, используемую по умолчанию. Это значит, что при последовательном отображении таблица прорисовывается сначала исходя из такой ширины столбцов. Затем, при загрузке и отображении содержимого ячеек ее размеры могут меняться. Может принимать значения в пикселах и в процентах горизонтального размера окна браузера. Допускается задание атрибута **width** в пропорциональном значении, имеющем вид: число*. Пропорциональное задание указывает число частей, необходимого для таблицы горизонтального пространства, приходящееся на один столбец в группе. Также в качестве значения может быть использована запись 0* (ноль со звездочкой), которая указывает браузеру минимизировать размеры столбцов. При этом столбцы будут иметь наименьшую необходимую ширину, для размещения своего содержимого.

  Примечание: *использование записи 0* (ноль со звездочкой), не позволяет браузеру осуществлять отображение таблицы последовательно, т.к. чтобы задать размеры таблицы ему нужно будет загрузить сначала все содержимое ячеек.*

Атрибут **width** может быть переопределен для любого столбца выделенной группы тегом COL с аналогичным атрибутом.

- **align**, **char**, **charoff**, **valign** – атрибуты выравнивания. Их описание и применение приведены в разделе "7.7. Как выровнять текст внутри ячеек?"

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);

- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** - внутренние события.

Тег <COL>

Тег <COL> служит для задания или изменения определенных характеристик некоторого количества столбцов, без объединения их в группу. Тег <COL> не осуществляет структурной разметки, которую производит тег <COLGROUP>. Тег <COL> может задавать характеристики как столбцам, объединенным в группы, так и столбцам в группах не состоящим (но не тем и другим одновременно). В первом случае он задается в содержимом тега <COLGROUP>, а во втором – вне содержимого элемента тега <COLGROUP>.

> **Примечание:** *наглядно разницу между тегами <COL> и <COLGROUP> можно увидеть при использовании атрибута rules тега <TABLE>, который может осуществлять отображение границ между группами столбцов (элементами структуры), но не реагирует на столбцы, описанные тегом <COL>.*

Например:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Олимпиада 2014 </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <h2 align=center> Таблица с объединенными в структуры
столбцами </h2>
   <center>
   <table cellspacing=0 cellpadding=10 bgcolor=#3f544a
frame=box rules=groups>
      <caption align=right><em> Итоги Олимпиады 2014 в Сочи </
em></caption>
      <colgroup span=1>
      </colgroup>
      <colgroup span=1 >
      </colgroup>
      <colgroup span=1 width=100>
      </colgroup>
      <colspan=3>
```

```
      <thead>
        <tr>
        <th>N<th> Страна <th> Всего медалей<th> Золото <th>
Серебро <th> Бронза
      </thead>
      <tr>
      <td>1<td> Россия <td>33<td>13<td>11<td>9
      <tr>
      <td>2<td> Норвеги я<td>26<td>11<td>5<td>10
      <tr>
      <td>3<td> Канада <td>25<td>10<td>10<td>5
      <tr>
      <td>4<td> США <td>28<td>9<td>7<td>12
      <tr>
      <td>5<td> Нидерланды <td>24<td>8<td>7<td>9
   </table>
   </center>
   </body>
</html>
```

Рис. 7.22. Таблица с использованием элементов структурной разметки COL и COLGROUP

В этом примере первые три столбца выделены как самостоятельные структурные единицы. Кстати, задание атрибута **width** для третьего из них позволило расположить фразу "Всего медалей" в две строки. Атрибут **frame** в значении **border** прорисовывает рамку вокруг таблицы, а атрибут **rules=group** прорисовывает границы внутри таблицы между структурными единицами. Первая строка отчерчена потому, что она также является структурной еди-

ницей, так как заключена в тег <THEAD>. Подробнее об атрибутах **frame** и **rules** читайте немного ниже, в разделе "7.8. Как изменять границы таблицы?"

Начальный тег <COL> является обязательным, конечный тег запрещен.

Атрибуты:

- **span** – необходимый атрибут, задает количество последовательных столбцов, подпадающих под влияние тега <COL>. Отсчет ведется, как и в случае с тегом <COLGROUP>, от первого столбца, не находящегося под влиянием предыдущих тегов <COL> или <COLGROUP>. В том случае, если тег <COL> задан внутри тега <COLGROUP> (встроен в него), то значение атрибута **span** тега <COL> перекрывает значение **span** родительского <COLGROUP>.
- **width** – указывает браузеру ширину каждого столбца, описываемого данным тегом <COL>. Принимает те же значения что и атрибут **width** тега <COLGROUP> и имеет перед ним приоритет.
- **align**, **char**, **charoff**, **valign** – атрибуты выравнивания. Их описание и применение приведены в разделе "7.7. Как выровнять текст внутри ячеек?"

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

7.5. Как подсчитать количество столбцов?

Чтобы браузер мог отобразить таблицу, он должен знать количество столбцов в таблице и их ширину. Определение числа столбцов в таблицу производится по следующим двум схемам:

- Если в содержимом тега <TABLE> присутствуют теги <COLGROUP> и/или <COL> (корректно заданные), то количество столбцов определяется как сумма следующих величин:

- Сумма значений всех атрибутов **span** содержащихся тегов <COL>
- Сумма значений атрибута span содержащихся тегов <COLGROUP>, не включающих в себя тегов <COL>. Атрибут **span** тега <COLGROUP>, содержащего в себе тег <COL>, не учитывается.

- Если тег <TABLE> не включает в свое содержимое теги <COLGROUP> и <COL>, то за количество столбцов принимается число ячеек строки, с максимальным их количеством. Строкам, у которых ячеек меньше, чем столбцов в таблице браузер, присоединяет пустые строки.

Примеры:

Все таблицы имеют по пять столбцов.

```
<table align=left>
   <caption><em> первый пример таблицы </em></caption>
   <colgroup span=5>
   </colgroup>
   <tr><td> Ячейка 1. <td>.....
   ..........
</table>

<table align=left>
   <caption><em> второй пример таблицы </em></caption>
   <colgroup span=2>
   <colspan=3>
   </colgroup>
   <colgroup span=2>
   <tr><td> Ячейка 1. <td>.....
   ..........
</table>

<table align=left>
   <caption><em> третий пример таблицы </em></caption>
   <colgroup>
   <colspan=2>
   <colspan=3>
   </colgroup>
   <tr><td> Ячейка 1. <td>.....
   ..........
</table>

<table align=left>
   <caption><em> четвертый пример таблицы </em></caption>
   <colspan=2>
```

```
   <colgroup>
   <colspan=2>
   <tr><td> Ячейка 1. <td>.....
   ..........
</table>

<table align=left>
   <caption><em> пятый пример таблицы </em></caption>
   <tr>
   <td><td><td><td>
   </tr>
   <tr>
   <td><td><td><td><td>
   </tr>
</table>
```

7.6. Как определить ширину таблицы?

Ширина таблицы определяется как сумма значений ширин всех столбцов таблицы, которые, в свою очередь, могут:

- быть заданы разработчиком:
  - в пикселах, в виде фиксированного значения. Таблица может быть отображена последовательно;
  - в процентах горизонтального размера окна браузера. Поскольку такое задание не зависит от самой таблицы, то и в этом случае поддерживается ее последовательное отображение;
  - в пропорциональном значении от всей необходимой ширины таблицы. Последовательное отображение таблицы может быть реализовано, если предварительно задана ее рекомендуемая ширина через атрибут **width** тега <TABLE>. В противном случае, браузеру придется сначала загрузить все данные таблицы, затем определить ее необходимую ширину, и только после этого распределить ее между столбцами.
- Быть не заданы разработчиком. В этом случае браузер не может форматировать таблицу последовательно, т.к. он вынужден будет подождать загрузки содержимого всех ячеек столбца, для определения его ширины. А поскольку загрузка осуществляется построчно, то придется ожидать загрузки содержимого всей таблицы.

Если в первом представлении таблицы ширина столбца оказалась недостаточной для отображения его содержимого, то она увеличивается, и таблица переформатируется.

7.7. Как выровнять текст внутри ячеек?

Язык HTML позволяет использовать дополнительные критерии горизонтального и вертикального выравнивания содержащейся в ячейке информации. Для этого используются дополнительные атрибуты **char** и **charoff**, значения атрибутов **align** и **valign**.

Атрибут горизонтального выравнивания **align** может принимать следующие значения:

- *left* – задает выравнивание по левому краю (используется по умолчанию);
- *right* – задает выравнивание по правому краю;
- *center* - задает выравнивание по центру;
- *justify* - задает выравнивание по ширине.

Возможные значения атрибута **valign**, ответственного за вертикальное выравнивание:

- *top* – содержимое ячейки выравнивается по верхнему краю;
- *bottom* - содержимое ячейки выравнивается по нижнему краю;
- *middle* – содержимое ячейки центрируется по вертикали (используется по умолчанию);
- *baseline* – информация всех ячеек одной строки располагается так, чтобы первые строки содержимого во всех ячейках располагались на одной линии. Если этот атрибут задан для одной ячейки строки, то он все равно распространяется на всю строку.

Атрибут **char** – указывает символ в текстовом фрагменте, служащий центром выравнивания. В качестве значения принимает символ, заключенный в кавычки, например: char="e". По умолчанию используется символ десятичной точки для текущего языка.

Пример задания:

```
<TR align="char" char="e">
```

Атрибут **charoff** – задает величину отступа содержимого ячейки от края, по которому производится выравнивание. Значения принимает в пикселах.

7.8. Как изменять границы таблицы?

За прорисовку рамки вокруг таблицы ответственен атрибут FRAME тега <TABLE>, а за внешний вид сетки внутри таблицы – атрибут RULES того же тега.

Атрибут **frame** может принимать следующие значения:

- *border* – рамка прорисовывается вокруг всей таблицы;
- *void* – рамка вокруг таблицы не прорисовывается;
- *above* – рамка прорисовывается только по верхней границе таблицы;
- *below* - рамка прорисовывается только по нижней границе таблицы;
- *hsides* - рамка прорисовывается и по верхней и по нижней границам таблицы;
- *lhs* - рамка прорисовывается только по левой границе таблицы;
- *rhs* - рамка прорисовывается только по правой границе таблицы;
- *vsides* - рамка прорисовывается и по правой и по левой границам таблицы.

Для атрибута **Rules** допустимы следующие значения:

- *all* – прорисовываются все внутренние линии таблицы;
- *none* – сетка внутри таблицы не отображается полностью;
- *rows* – прорисовываются линии сетки, разделяющие строки таблицы;
- *cols* - прорисовываются линии сетки, разделяющие столбцы таблицы;
- *groups* - прорисовываются линии сетки, разделяющие группы.

Использование атрибутов **frames** и **rules** можно наблюдать на примере:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример таблицы </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no>
      <h2 align=center> Пример таблицы </h2>
   <table align=left border=3 cellspacing=0 cellpadding=10
bgcolor=#3f544a>
      <tr><td>
         <table frame=hsides rules=groups cellspacing=0
cellpadding=5 bgcolor=#fce8e8>
            <caption align=bottom>
               <em><b> Название таблицы </b></em>
            </caption>
      <thead>
      <tr>
         <th> Заголовок 1
         <th> Заголовок 2
         <th> Заголовок 3
      </thead>
      <tr>
         <td> Ячейка 1-1 <td> Ячейка 2-1 <td> Ячейка 3-1
      <tr>
         <td> Ячейка 1-2 <td> Ячейка 2-2 <td> Ячейка 3-2
      <tr>
         <td> Ячейка 1-3 <td> Ячейка 2-3 <td> Ячейка 3-3
      <tr>
         <td> Ячейка 1-4 <td> Ячейка 2-4 <td> Ячейка 3-4
      <tr>
         <td> Ячейка 1-5 <td> Ячейка 2-5 <td> Ячейка 3-5
      <tr>
         <td> Ячейка 1-6 <td> Ячейка 2-6 <td> Ячейка 3-6
      <tr>
         <td> Ячейка 1-7 <td> Ячейка 2-7 <td> Ячейка 3-7
      <tr>
         <td> Ячейка 1-8 <td> Ячейка 2-8 <td> Ячейка 3-8
      <tr>
         <td> Ячейка 1-9 <td> Ячейка 2-9 <td> Ячейка 3-9
   </table>
   </body>
</html>
```

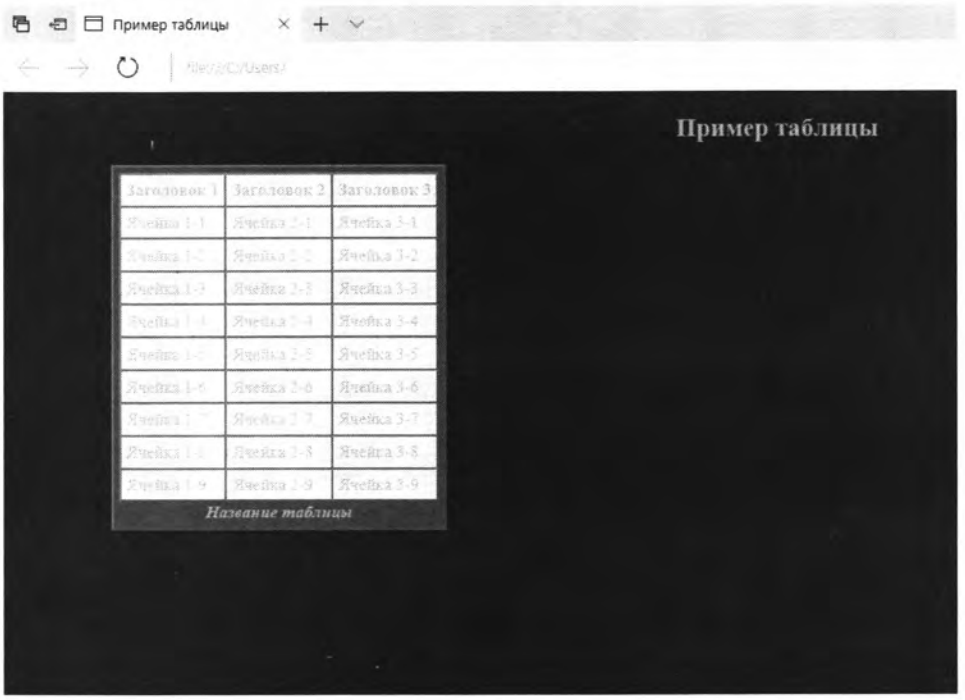
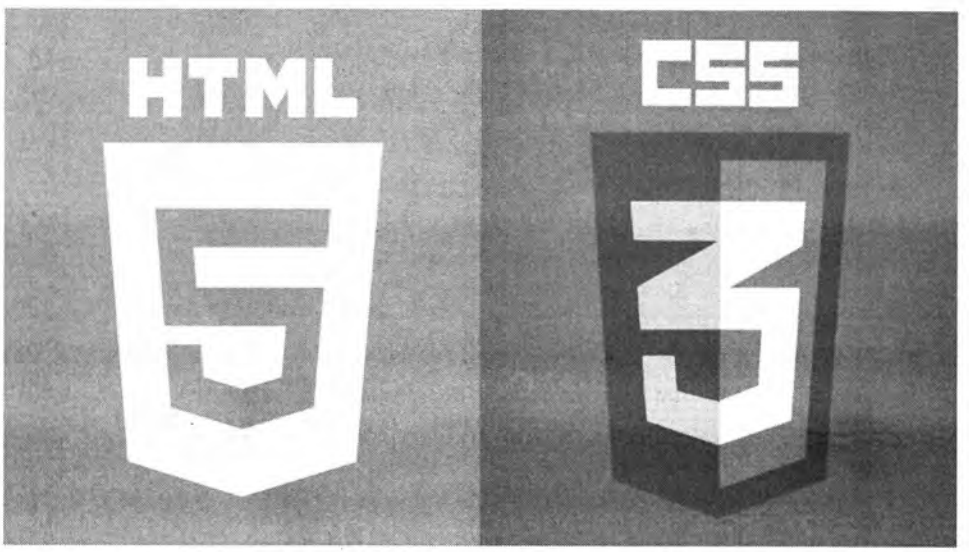


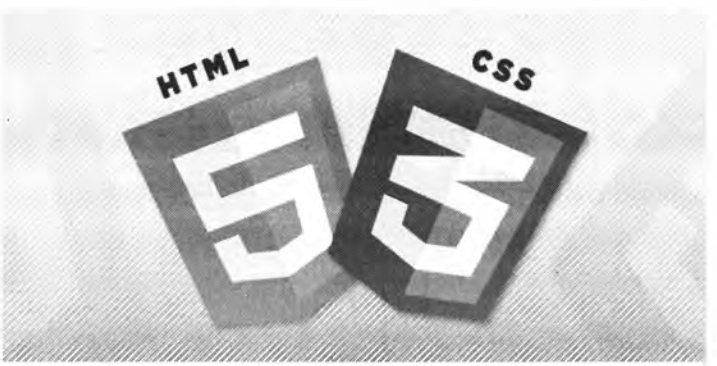
Рис. 7.23. Пример использования атрибутов frames и rules для прорисовки границ таблицы


HTML
5
CSS
3

HTML5

Глава 8.

Скрипты



8.1. Что такое скрипт?

Скрипты (или сценарии) представляют собой программы, написанные на специальном языке скриптов: JavaScript, VBScript, Perl и др. Различают клиентские скрипты и скрипты, выполняемые на стороне сервера. Последние использовать нежелательно, т.к. это приводит к более частому обмену данными между сервером и удаленным пользователем. Однако есть случаи, когда применение скриптов, выполняемых на стороне сервера, является необходимым:

- для динамического поддержания Web-страницы с переменным содержанием, учитывающем информацию об удаленном пользователе;
- для поддержания взаимодействия Web-страницы с базами данных, хранящихся на сервере;
- для организации взаимодействия Web-страницы скаким-либо внешними прикладными программами и отображения результатов их выполнения в содержимом страницы;
- для обработки форм пользователя на сервере.

Клиентские скрипты сопровождают HTML-документ или встраиваются в его содержимое. Использование скриптов позволяет сделать документы более интерактивными и визуально привлекательными. Скрипты позволяют:

- изменять содержимое документа и его представление во время загрузки;
- производить обработку таких событий как: загрузка, удаление определенной информации HTML-документа, перемещение мыши, нажатие кнопок на мыши и т.п.;
- осуществлять предварительную обработку вводимых в формы данных, например, заполнять определенные поля формы по данным, введенным в другие поля;
- управлять фокусом в рамках документа;
- осуществлять возможности пользовательского интерфейса на Web-странице.

Скрипты разделяются на два типа:

- скрипты, выполняющиеся один раз при загрузке документа;

- скрипты, выполняемые всякий раз, когда происходит определенное событие.

Вот далеко не полный список возможностей, реализуемых с помощью клиентских скриптов:

- Создание бегущих строк и сообщений в статусной строке браузера.
- Выпадающие горизонтальные и вертикальные меню.
- Перемещаемые меню.
- Создание системных кнопок различного назначения.
- Создание анимационного следа вслед за движущимся курсором.
- Календарь.
- Создание окон с сообщениями.
- Звуковое сопровождение ссылок.
- Часы.
- Создание переливающихся изображений (эффект волн).
- Отключение правой или левой кнопки мыши.
- Создание уникального меню, возникающего при нажатии на правую или левую кнопки мыши.

Включение скрипта в HTML-документ осуществляет тег <SCRIPT>. Тег <SCRIPT> может быть задан как в заголовке, так и в теле документа.

Браузеры при обработке документа не могут определить язык скрипта самостоятельно и, следовательно, корректно его интерпретировать. При использовании скриптов необходимо указывать язык, на котором они написаны. В HTML возможны два способа задания языка скриптов:

- установить определенный язык в качестве используемого по умолчанию;
- указывать язык персонально для каждого скрипта при его задании.

Чтобы задать определенный язык сценариев используемым по умолчанию, надо либо указать его название в заголовке HTML-документа через элемент META:

```
<META http-equiv="Content-Script-Type" content="text/
javascript">
```

либо в поле Content-Script-Type заголовка HTTP:

```
Content-Script-Type: text/javascript.
```

Языком скриптов, установленным по умолчанию, считается язык, указанный в последнем объявлении Content-Script-Type тега <META>. Аналогичное поле HTTP-заголовка при этом не учитывается. Если отсутствует объявление языка скриптов в тега <META>, то по умолчанию используется язык, указанный в HTTP-заголовке.

Чтобы указать язык для конкретного скрипта, применяется атрибут **Type** тега <SCRIPT>. Задание языка в самом теге <SCRIPT> имеет приоритет над заданным по умолчанию, но только для одного конкретного скрипта.

Выполнение скриптов можно привязать с определенными событиями. Эти события представляют собой результат работы пользователя с HTML-документом в окне браузера. Привязка скриптов к событиям осуществляется по схеме

```
Имя_события=Имя_скрипта
```

В HTML 5 различают следующие события, указываемые в качестве атрибутов элементов, для которых они определены:

- Onclick – событие, происходящее при однократном нажатии кнопки мыши, когда курсор наведен на содержимое элемента, для которого определено это событие.
- Ondblclick–событие, происходящее при двойном нажатии кнопки мыши, когда курсор наведен на содержимое элемента, для которого определено это событие.
- Onload – это событие происходит, когда браузером закончилась загрузка документа в окно или всех фреймов. Задается у элементов BODY и FRAMESET.
- Onunload – это событие происходит, когда браузер закончил удаление документа из своего окна или из фрейма. Задается у элементов BODY и FRAMESET.
- Onmousedown – это событие происходит при нажатии кнопки мыши, когда курсор наведен на содержимое элемента, для которого определено это событие.

- Onmouseup - это событие происходит при отпускании кнопки мыши, когда курсор наведен на содержимое элемента, для которого определено это событие.
- Onmouseover – это событие происходит, когда курсор находится над содержимым элемента, для которого определено это событие.
- Onmousemove – это событие происходит при перемещении курсора над содержимым элемента, для которого определено это событие.
- Onmouseout - это событие происходит при перемещении курсора за пределы содержимого элемента, для которого определено это событие.
- Onfocus – это событие происходит, когда элемент, для которого определено это событие, получает фокус. Фокус на элемент может быть установлен либо с помощью "мыши", либо с помощью перебора элементов последовательности перебора (клавишей "Tab", например). Событие (атрибут) Onfocus может быть определено у следующих тегов: LABEL, INPUT, SELECT, TEXTAREA и BUTTON.
- Onblur – это событие происходит переходе фокуса с элемента, для которого определено это событие.
- Onkeypress – это событие происходит при нажатии и отпускании любой клавиши на содержимом элемента. Событие Onkeypress определено почти для всех HTML-тегов.
- Onkeydown – это событие происходит при нажатии любой клавиши на содержимом элемента. Событие Onkeydown определено почти для всех HTML-тегов.
- Onkeyup - это событие происходит при отпускании нажатой на содержимом элемента клавиши. Событие Onkeyup определено почти для всех HTML-элементов.
- Onsubmit – это событие происходит при отправке формы. Соответственно, оно может быть определено только для тега формы FORM.
- Onereset - это событие происходит при сбросе содержимого формы. Так же, как и событие Onsubmit определено только для тега формы FORM.
- Onselect – это событие происходит при выделении пользователем некоторого текстового фрагмента в текстовом поле. В качестве последнего может выступать содержимое тегов INPUTи TEXTAREA. Именно для них и может быть определено событие Onselect.
- Onchange – это событие происходит при потере управляющим элементом фокуса, если с момента получения фокуса его содержимое было из-

менено. Событие Onchange может быть определено для тегов заполнения форм SELECT, INPUT и TEXTAREA.

Использование тега PROGRESS совместно с JavaScript позволяет графически отображать процент завершенности задачи.

8.2. Тег NOSCRIPT

На всякий случай, разработчиком должна быть предусмотрена, насколько это возможно, корректная обработка документа пользователями, не поддерживающими скрипты. Для этого предполагается использование элемента NOSCRIPT и метода сокрытия скрипта.

С помощью элемента NOSCRIPT реализуется возможность задания альтернативного содержимого скрипта, если он не может быть выполнен. Содержимое этого элемента отображается в окне браузера в трех случаях:

- когда браузер по своей природе не поддерживает скрипты;
- когда браузер так настроен, чтобы не поддерживать скрипты;
- когда браузер вообще поддерживает скрипты, но не поддерживает данный язык скриптов.

Например:

```
<SCRIPT type="text/porh">
.....текст скрипта написанного.....
.....на языке porh (вымышленном).....
.....который вставляет в страницу.....
.....переливающуюся картинку.....
</SCRIPT>
<NOSCRIPT>
<!—вместо переливающейся будет вставлена обычная картинка->
```

```
<IMG src="kartinka.gif"
</NOSCRIPT>
```

Задание начального и конечного тегов элемента NOSCRIPT является обязательным.

8.3. Как рисовать разные объекты? Тег CANVAS

Тег CANVAS позволяет задать область на веб-страницы, на которой с помощью JavaScript можно отобразить любые рисованные объекты, анимацию, игры и т.д.

Пример:

```
<!DOCTYPE html>
<html>
<head>
<title> Использование тега CANVAS </title>
<meta charset="utf-8">
<script>
   window.onload = function() {
    var drawingCanvas = document.getElementById('smile');
    if(drawingCanvas && drawingCanvas.getContext) {
     var context = drawingCanvas.getContext('2d');
     // верхняя окружность
     context.strokeStyle = "#f00";
     context.fillStyle = "#f00";
     context.beginPath();
context.arc(300,100,100,0,Math.PI*2,true);
     context.closePath();
     context.stroke();
     context.fill();
    // верхний левый глаз
     context.fillStyle = "#fff";
     context.beginPath();
     context.arc(284,90,8,0,Math.PI*2,true);
     context.closePath();
     context.stroke();
     context.fill();
     // верхний правый глаз
     context.beginPath();
```

```
      context.arc(316,90,8,0,Math.PI*2,true);
      context.closePath();
      context.stroke();
      context.fill();
      // верхний рот
      context.beginPath();
      context.moveTo(270,115);
      context.quadraticCurveTo(300,130,330,115);
      context.quadraticCurveTo(300,150,270,115);
      context.closePath();
      context.stroke();
      context.fill();
      // средняя окружность
      context.strokeStyle = "#ff0";
      context.fillStyle = "#ff0";
      context.beginPath();
      context.arc(300,300,100,0,Math.PI*2,true);
      context.closePath();
      context.stroke();
      context.fill();
   // средний левый глаз
context.fillStyle = "#000";
context.beginPath();
      context.arc(284,290,8,0,Math.PI*2,true);
      context.closePath();
      context.stroke();
      context.fill();
// средний правый глаз
context.beginPath();
context.arc(316,290,8,0,Math.PI*2,true);
      context.closePath();
      context.stroke();
      context.fill();
      // средний рот
      context.beginPath();
      context.moveTo(270,315);
      context.quadraticCurveTo(300,330,330,315);
      context.quadraticCurveTo(300,350,270,315);
      context.closePath();
      context.stroke();
      context.fill();
      // нижняя окружность
      context.strokeStyle = "#0b3";
      context.fillStyle = "#0b3";
      context.beginPath();
      context.arc(300,500,100,0,Math.PI*2,true);
```

```
    context.closePath();
    context.stroke();
    context.fill();
  // нижний левый глаз
    context.fillStyle = "#fff";
    context.begin Path();
    context.arc(284,490,8,0,Math.PI*2,true);
    context.closePath();
    context.stroke();
    context.fill();
    // нижний правый глаз
    context.beginPath();
    context.arc(316,490,8,0,Math.PI*2,true);
    context.closePath();
    context.stroke();
    context.fill();
    // нижний рот
    context.beginPath();
    context.moveTo(270,515);
    context.quadraticCurveTo(300,530,330,515);
```

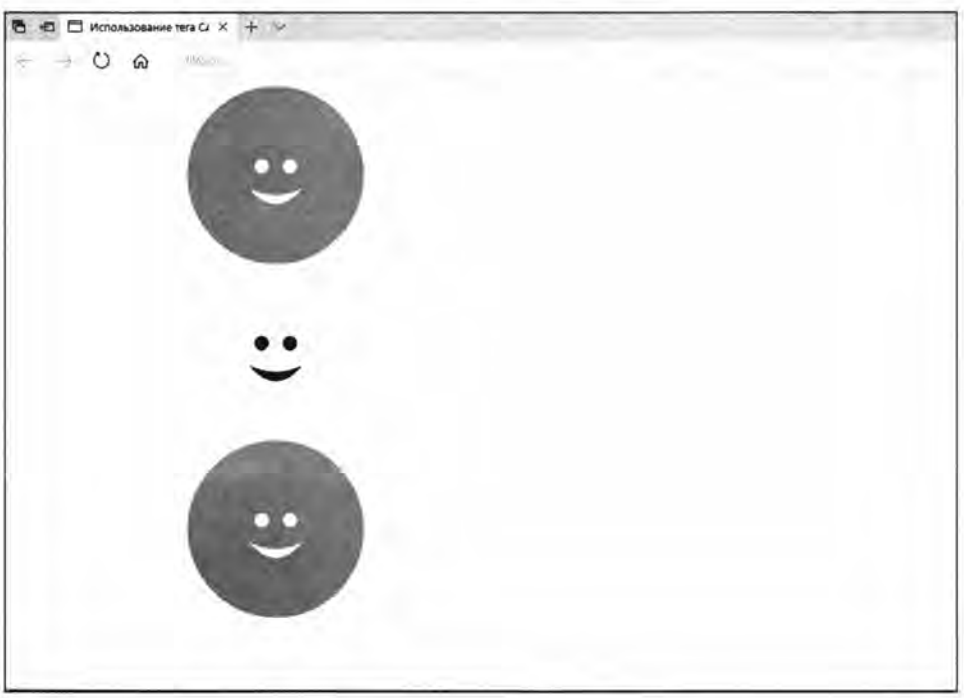


Рис. 8.1. Использование тега CANVAS

```
        context.quadraticCurveTo(300,550,270,515);
        context.closePath();
        context.stroke();
        context.fill();
      }
    }
</script>
</head>
<body>
<canvas id="smile" width="500" height="800">
</canvas>
</body>
</html>
```

HTML5

Глава 9.

Ссылки

В этой главе вы узнаете:

- *Что такое ссылки?*
- *Как использовать тег A?*
- *Как использовать тег LINK?*

9.1. Что такое ссылки?

Ссылки являются основой гипертекстовой организации документов.

Использование ссылок по праву считается основным достоинством World-WideWeb. Ссылки связывают один информационный ресурс с другим и каждая из них имеет два конца: исходящий и целевой.

Ссылки бывают двух видов: контекстные ссылки и заглавные ссылки.

Контекстные ссылки задаются тегом <A> в теле HTML-документа. В заголовке их использовать запрещено. Контекстные ссылки в качестве целевого ресурса могут использовать изображение, видеоклип, звуковой фрагмент, какой-либо элемент текущего или удаленного HTML-документа, сам удаленный HTML-документ и т.п. Контекстные ссылки могут ссылаться на определенные места документа и таким образом осуществлять навигацию в пределах одного документа. Активизация контекстной ссылки приводит к открытию нового целевого документа или к переходу фокуса на определенное место в текущем документе.

Заглавные ссылки задаются тегами <LINK> в пределах HTML-заголовка. В теле документа их использование не допустимо. Заглавные ссылки в качестве целевого ресурса могут использовать только другой HTML-документ. Эти ссылки устанавливают вид взаимных отношений между разными документами, то есть, например, заглавная ссылка указывает, что такая-то страница является персональной домашней страничкой автора данного сайта.

На данный момент, связи между документами, устанавливаемые заглавными ссылками почти не находят своего практического использования. Информация, содержащаяся в них, может использоваться поисковыми машинами в своей работе. Например, в качестве запроса поисковой системе

можно указать поиск документов, ссылающихся на определенный Web-ресурс через тег LINK.

Ссылки, определяемые тегами Link, могут описывать положение данного документа в последовательности нескольких документов, выступая в роли логической разметки этой последовательности. При этом, для каждого документа последовательности указывается (через элемент LINK) URL-адрес документа, который расположен перед ним в последовательности, и URL-адрес документа, следующего за ним.

9.2. Как использовать тег A?

Тег A в HTML реализует контекстные ссылки. Начальный и конечный теги этого элемента являются обязательными.

Атрибуты:

- **name** – необязательный атрибут, присваивает элементу **A** имя, в результате чего он сам может служить целью для другой ссылки. В качестве значения должно задаваться уникальное в пределах текущего документа имя. Данный атрибут использует общее пространство имен с атрибутом **id**.
- **Href** - обязательный атрибут, указывающий месторасположение целевого Web-ресурса. В качестве значения задается URL-адрес ресурса.
- **hreflang** – необязательный атрибут, указывающий базовый язык целевого Web-ресурса. Этот атрибут может использоваться только в сочетании с атрибутом **href**. В качестве значения пишется код языка.
- **rel** - атрибут, описывающий отношение текущего документа к целевому ресурсу при переходе к нему по данной ссылке. В качестве значения указываются имена видов связи. Практически не используется.
- **rev** - атрибут, описывающий отношение целевого ресурса к текущему документу, при переходе к нему по обратной ссылке (откате кнопкой браузера "Назад").
- **type= content-type** - указывает тип содержимого целевого ресурса.
- **shape** – атрибут, применяемый в случае использования клиентских навигационных карт. В качестве значения этого атрибута указывается форма активной области.

- **coords** - атрибут, применяемый в случае использования клиентских навигационных карт. В качестве значения этого атрибута указываются граничные координаты активной области.
- **accesskey** – в качестве значения этого атрибута указываются "горячие клавиши" для активизации данной ссылки.
- **tabindex** – атрибут, определяющий последовательность перехода между ссылки при нажатии на клавиатуре клавиши Tab.
- **download** – атрибут, предлагающий скачать документ, указанный в адресе ссылки.

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Пример задания:

```
<A href="http://techrussia.org"> переход на сайт TechRussia </A>
```

По умолчанию, все ссылки отображаются в окне браузера синим цветом и подчеркнутыми. Уже просмотренные ссылки меняют свой цвет на фиолетовый. Если в содержимом тега **A** указано изображение, то нажатие на него приведет к активизации ссылки. На этом принципе построены все фотогалереи в Интернете. Такие изображения-ссылки по умолчанию отображаются в синей рамке, меняющей затем свой цвет на фиолетовый. Стандартные цветовые установки ссылок могут быть изменены с помощью атрибутов элемента BODY или средствами каскадных таблиц стилей.

Пример задания изображения в качестве ссылки:

```
<A href="http://techrussia.org"><IMG src="robots.png"></A>
```

Элемент **A** и его целевой ресурс являются якорями контекстной ссылки. Наглядно это можно представить следующим образом.

При активизации ссылки, задаваемой тегом **A**, осуществляется переход на ресурс или на определенную его часть. В случае отката из целевого ресурса с помощью клавиши "Назад", переход осуществляется на тот документ и на то место документа, где находится тег **A**, задающий эту ссылку.

В качестве целевого якоря можно задать любой элемент тела, текущего или внешнего документов. Такое задание позволяет осуществлять переход на определенные части документа.

> **Примечание.** *Загружается по-прежнему весь документ, но если в окне браузера он целиком не помещается, то показ документа будет осуществляться с той его части, которая заключена внутри содержимого HTML-элемента, являющимся целевым ресурсом ссылки. Если такое задание отсутствует, то показ документа производится с его начала.*

Чтобы HTML-документ мог служить целевым якорем, необходимо присвоить ему уникальное имя, используя атрибут **id**.

Пример использования:

```
<!DOCTYPE html>
<HTML>
   <HEAD>
   <TITLE> Использование ссылок в HTML-документах </TITLE>
</HEAD>
<BODY leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no>
<H2 align=center>
Пример использования контекстных ссылок в HTML-документах
</H2>
<HR size=20 color=lightgrey>
<A href="http://techrussia.org"> переход на сайт TechRussia </
A>
<HR>
<A href="face1.jpg">
<IMG src="face1.jpg"></A>
<HR>
<A href="mailto:partners@a-techevent.ru">
послать письмо организаторам TechRussia</A>
</BODY>
</HTML>
```

Рис. 9.1. Пример использования тега A для ссылок на различные ресурсы

Каждый заданный тега A является якорем по крайней мере для одной ссылки(обратной ссылки по клавише "Назад"). Местонахождение якоря определяется расположением содержимого элемента A. Наличие атрибута href устанавливает тег A целевым якорем только одной, обратной ссылки. Задание атрибута name позволяет использовать содержащий его тег A в качестве целевого якоря для неограниченного числа ссылок.

Пример использования якорей:

В качестве примера рассмотрим документ, который содержит в себе описания всех направлений всероссийского технологического марафона TechRussia. Такой документ содержит довольно много текстовой информации, что затрудняет поиск нужного направления. В этом случае рекомендуется в начало документа вынести список ссылок, ссылающихся на все направления. Благодаря этому переход к нужной информации значительно упрощается.

```
<!DOCTYPE HTML>
<HTML>
   <head>
   <title>
   Пример организации навигации в переделах одного документа
   </title>
   </head>
```

```
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Направления всероссийского технологического
марафона TechRussia
      </H2>
      <ul>
      <li><A href=#VR> Виртуальная и дополненная реальность </A>
      <li><A href=#IoT> Интернет-вещей </A>
      <li><A href=#BPLA> Беспилотные летательные аппараты </A>
      <li><A href=#Space> Космические технологии </A>
      <li><A href=#Robots> Робототехника </A>
      <li><A href=#ASU> Автоматизированные системы управления </A>
      <li><A href=#Design> Промышленный дизайн </A>
      <li><A href=#Finance> Финансовые технологии </A>
      <li><A href=#Neuro> Нейротехнологии </A>
      </ul>
      <H3 id=VR> Виртуальная и дополненная реальность </H3>
      Трек, посвященный технологиям дополненной и виртуальной
реальности, их применению в различных областях. Примеры проектов:
      Проекция модели оперируемой области для хирурга
      Система для совместной разработки архитектурного проекта в
смешанной реальности
      Реабилитация заключенных с помощью приложения для VR
      .....
      <H3 id=IoT> Интернет-вещей </H3>
      Интернет вещей - технология взаимодействия физических объектов
друг с другом или со внешней средой. Примеры проектов:

      Умный умывальник со встроенным цифровым микроскопом,
отслеживающий бактерии на поверхности рук
      Браслет с датчиками, позволяющий туристам регистрироваться в
отеле, совершать покупки продуктов в супермаркете
      Программа "Яндекс.Пробки"

      .....
      <H3 id=BPLA> Беспилотные летательные аппаратов </H3>
      Создание и эксплуатация беспилотных летательных аппаратов
(БПЛА). Примеры проектов:

      Дроны для доставки интернет-заказов
      Технология полета БПЛА на расстояния более десяти
километров
      Дроны для контроля за удаленными пастбищами, водоемами и
лугами
      .....
      <H3 id=Space> Космические технологии </H3>
```

Космические технологии, GPS, спутниковое телевидение. Примеры проектов:

Аппараты для уборки космического мусора
"Купсат" - миниатюрный космический спутник, созданный благодаря знаниям в области микроминиатюризации и нанотехнологий
Космические тросовые системы

```
<H3 id=Robots> Робототехника </H3>
```

Роботы - это универсальные работники, которые применяются во всех сферах жизни человека, выполнять тяжелую и опасную работу. Примеры проектов:

Разработка роботов-официантов
Робот для прокладки труб и кабелей под землей
Робот для внутренней отделки помещений

```
<H3 id=ASU> Автоматизированные системы управления </H3>
```

Автоматизированные системы управления – это человеко-машинная совокупность, обеспечивающая автоматизированный сбор, обработку информации и оптимизацию управления в социальных, технических сферах человеческой деятельности.

Система контроля брака продукции при автоматической упаковке товарных партий
Причал с беспилотными катерами для водных прогулок
Система управления роботами-официантами

```
<H3 id=Design> Промышленный дизайн </H3>
```

Благодаря промышленному дизайну и эргономике создаются предметы, повышающие качество труда и жизни. Примеры проектов:

Миниатюрная солнечная панель Heli-on
Модульные ветрогенераторы Windflock
Коленный стул

```
<H3 id=Finance>Финансовые технологии </H3>
```

Блокчейн, схемы монетизации, финансовые технологии для потребителя. Примеры проектов:

Криптовалюта на основе блокчейна
Система проверки клиента на основе данных из других банков и организаций
Криптовалютный краудфандинг

```
    <H3 id=Neuro> Нейротехнологии </H3>
    Нейротехнологии или нейронауки - это совокупность
технологий, созданных на основе принципов работы нейросистемы.
Примеры проектов:

    Нейрокомпьютер, который перерабатывает информацию по
принципу работы мозга
    Система контроля бодрствования водителей
    Нейронные сети для распознавания образов, прогнозирования
рисков и поддержки принятия решений
    .....
</body>
</html>
```

На экране это будет выглядеть так:

Рис. 9.2. Документ, с использованием тега A для навигации в пределах одного документа

При нажатии на один из элементов списка происходит переход к соответствующей части документа.

В HTML допустимо задание тега **A**, не определяющего якоря и не имеющего атрибутов **href**, **name** и **id**. Тогда значения этих атрибутов могут быть заданы скриптами позднее.

В HTML ссылки и якоря, определяемые элементом **A** не могут быть вложенными, т.е. недопустимо использование одного тега **A** в содержимом

другого тега A. Имена якорей, задаваемые значениями атрибутов **name** и **id** должны быть уникальными в пределах одного документа. Атрибуты **name** и **id** используют одно пространство имен. Имена не чувствительны к регистру, и поэтому имена "TOM" и "tom" воспринимаются браузером как одно имя. Имена якорей должны содержать только символы ASCII.

Пример некорректного задания:

```
<P>
<A name = SHTIRLITZ> анекдоты про Штирлица </A> и хорошо
забытые старые <Aid = shtirlitz> анекдоты про Штирлица </A>
</P>
```

В этом случае никакой критической ситуации при этом не возникает, просто браузер будет осуществлять переход на первый от начала якорь с именем Shtirlitz.

9.3. Как использовать тег LINK?

Тег LINK реализует заглавные ссылки в HTML-документах. Он задает вид взаимоотношений между содержащим его документом и внешним документом, устанавливает между ними логическую связь. Начальный тег обязателен, конечный тег запрещен.

Атрибуты:

- **href** - указывает URL-адрес документа, взаимоотношение с которым описывается.
- **rel** - атрибут, определяющий отношение между текущим и внешним документами. С помощью этого атрибута W3C пытается запрограммировать клавишу "Вперед" браузера.
- **rev** - атрибут, определяющий отношение между текущим и внешним документами. С помощью этого атрибута W3C пытается запрограммировать клавишу "Назад" браузера.
- **hreflang** – необязательный атрибут, указывающий базовый язык целевого Web-ресурса. Этот атрибут может использоваться только в сочетании с атрибутом **href**. В качестве значения пишется код языка.
- **charset** - указывает кодировку символов целевого документа

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** - заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Один тег LINK устанавливает связь только с одним внешним документом. Однако в HTML-документе может присутствовать несколько тегов Link.

Тег Link обычно содержит URL-адрес документа, с которым задается взаимоотношение, и тип самого взаимоотношения показывает, чем документ, указанный в ссылке, является по отношению к текущему документу.

Вот несколько наиболее часто используемых типов взаимоотношений:

Для атрибута **rev**:

- *made* - адрес электронной почты автора текущего документа
- *author* - автор страниц автора данного документа
- *editor* - страница редактора данного документа
- *owner* - страница владельца данного документа

Для атрибута **rel**:

- *home* - начальная страница
- *previous* - предыдущая страница
- *next* - следующая страница
- *bookmark* - закладка
- *copyright* - авторское право
- *alternate* - альтернативная версия документа
- *help* - справочное пособие

Запись

```
<Linkhref= "http://techrussia.org" rel= next>
```

читается как:

```
документ по адресу http://techrussia.org является следующим в
логической последовательности после данного документа
```

Основное практическое применение тега Link, на данный момент времени, заключается в подключении внешних таблиц стилей (CSS).

Например:

```
<!DOCTYPE html >
<HTML>
<HEAD>
<TITLE> Динозавры - это круто </TITLE>
< LINK rel="stylesheet" href="dinoz.css" type="text/css">
</HEAD>
<BODY>
     <H1> Страница, посвященная динозаврам </H1>
<P>
Динозавры - это…….
…….
</P>
…….
</BODY>
</HTML>
```

HTML5

Глава 10.

Мультимедиа-объекты

В этой главе вы узнаете:

- *Что такое мультимедиа-объекты?*
- *Как вставить изображение?*
- *Как вставить аудио и видео?*
- *Как вставить другие мультимедиа-объекты?*
- *Как группировать объекты?*

10.1. Что такое мультимедиа-объекты?

Возможности HTML5 позволяют значительно повысить привлекательность Web-страницы благодаря включению в нее различных мультимедиа-объектов: изображений, видеоклипов, звуковых фрагментов, апплетов, другие HTML-документов. Универсальным HTML-элементом, осуществляющим вставку объектов мультимедиа, является тег OBJECT. Наряду с ним применяется тег IMG, вставляющий в HTML-документ изображения. Тег OBJECT имеет общий характер и может производить включение в Web-страницу любых существующих объектов (в том числе изображения) и объектов, которые будут появляться в будущем.

10.2. Как вставить изображение?

Не вызывает сомнений, что использование изображений является основным способом, благодаря которому любой документ делается привлекательным и доходчивым. Практически ни одна публикация, как в обычной печати, так и в Web, не обходится без иллюстраций. Они позволяют разработчику лучше и доходчивее передать содержимое документа, сделать его более наглядным. Декоративные изображения делают работу с документами приятной для глаз. Однако, используя изображения в своих публикациях, не следует забывать о чувстве меры. Большое количество графической информации может не только сделать вашу страницу безвкусной по дизайну, но и беспричинно увеличить длительность ее загрузки, что всегда приводит к нежелательным затратам времени на ожидание и отпугивает посетителей. Следует помнить, что излишняя пестрота даже неискушенного пользователя начинает очень скоро раздражать, что не способствует популярности Вашего Web-узла.

В HTML - документах различают изображения, встраиваемые в качестве иллюстраций, и фоновые изображения.

Работа с последними абсолютно идентична работе с фоновыми изображениями в Windows при помещении их на Desktop. Единственное отличие за-

ключается в том, что фоновые изображения в HTML-документах заполняют все окно браузера. В том случае, если геометрические размеры фонового изображения меньше окна браузера, все пространство окна заполняется по принципу мозаики. Отсюда следует, что при использовании маленьких картинок для задания фона, они должны быть выполнены таким образом, чтобы, при мозаичном расположении, границы между ними были незаметны.

В качестве фоновых изображений рекомендуется использовать небольшие декоративные картинки (текстуры) спокойного, мягкого цвета с ненавязчивым рисунком. Однако, могут использоваться и большие по размерам изображения. Только в этом случае, рекомендуется как можно лучше оптимизировать его, чтобы это изображение занимало как можно меньший объем памяти.

Задание фонового изображения в HTML, как уже говорилось, осуществляется атрибутом BACKGROUND тега BODY. При загрузке документа первоначально отображается его текстовая часть, и только потом загружаются изображения, в том числе и фоновые. При этом текст отображается на фоне, цвет которого установлен (или не установлен) атрибутом BGCOLOR. Поэтому настоятельно рекомендуется указывать в качестве фонового цвет, который наиболее близок к основному цвету фонового изображения.

```
Пример:
<!DOCTYPE html>
<html>
   <head>
      <title>
         Документ с заданным фоновым изображением
      </title>
   </head>
   <body background="fon.jpg">
      <h2 align=center> Пример документа с фоновым изображением
</h2>
      <hrcolor=darkgraysize=20>
      <p> "Мы живем в эпоху, когда расстояние от самых безумных
фантазий до совершенно реальной действительности сокращается с
невероятной быстротой."<p>
      <i> Максим Горький </i>
      <hr color=darkgray size=20>
   </body>
</html>
```

Внедрение иллюстративных изображений в HTML осуществляется посредством тега IMG, имеющего следующие атрибуты:

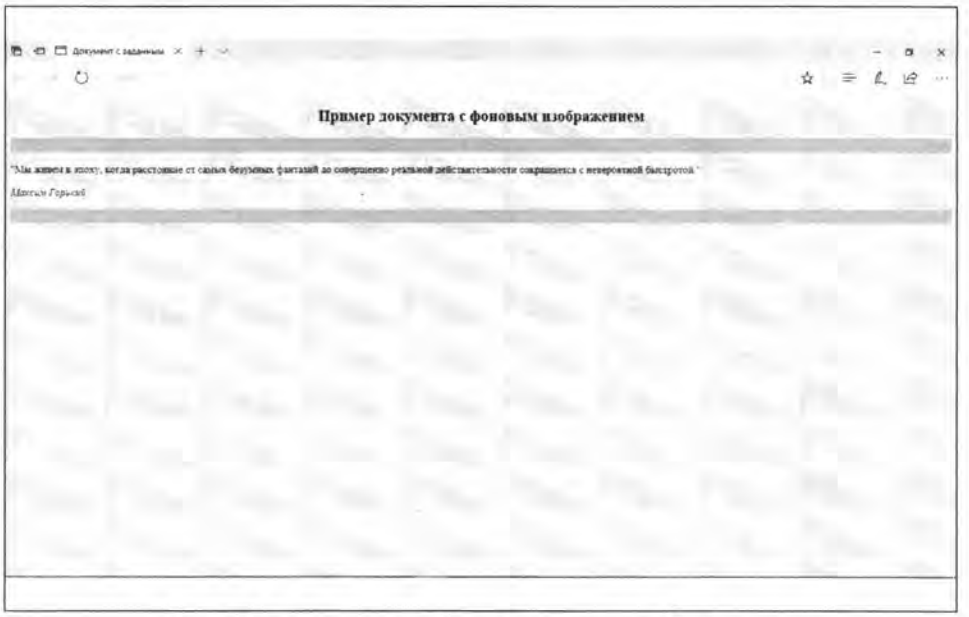


Рис. 10.1. Пример документа с фоновым изображением

- **src** - обязательный атрибут, задающий URL-адрес (полный или относительный) расположения изображения
- **align** – выравнивание
- **width**, **height** - геометрические размеры изображения, задаваемые либо в пикселах, либо в процентах видимого пространства окна браузера
- **hspace**, **vspace** - указывает горизонтальные и вертикальные отступы от изображения соответственно. Значение этих атрибутов задается в пикселях и позволяет вокруг изображения оставлять свободное пространство определенных размеров
- **border** - атрибут, управляющий наличием и шириной рамки вокруг изображения. Значение задается в пикселях и определяет ширину рамки. По умолчанию вокруг изображения не прорисовывается. Исключением являются случаи, когда изображение является ссылкой
- **alt** - атрибут, имеющий в качестве значения текстовую строку, которая является альтернативной текстовой информацией текущего изображения
- **longdesc** – атрибут, позволяющий указать адрес документа, где содержится аннотация к картинке

- **usemap** – использовать изображение совместно с клиентской навигационной картой, имя которой указывается в качестве значения этого атрибута
- **ismap** – использовать изображение совместно с серверной навигационной картой, имя которой указывается в качестве значения этого атрибута

Стандартные необязательные атрибуты:

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Умелое использование атрибутов позволяет осуществлять широкий спектр приемов представления изображений в HTML-документах. Остановимся на них поподробнее.

Выравнивание

Параметры выравнивания задают расположение изображения относительно текста и других элементов в документе. Также с помощью них можно осуществлять привязку изображения к краям документа, такие и только такие изображения могут обтекаться текстом.

Для тега IMG атрибут **align** может принимать следующие значения:

- *left* - изображение прикрепляется к левому краю документа. Обтекание изображения текстом производится по правой стороне;
- *right* - изображение прикрепляется к правому краю документа и обтекается текстом с левой стороны (рис. 10.2);

- *top* - выравнивание по верхней границе. Это значит, что изображение будет вставлено таким образом, чтобы его верхняя граница находилась на уровне верхней границы самого высокого элемента в текущей строке. В этом и последующих типах выравнивания изображение не обтекается,

Рис. 10.2. Демонстрация результата применения атрибута align в значении right

а как бы встраивается в текст. Для наглядности можно представить, что изображение является буквой, только большого размера и воспринимается браузером как обычный элемент строки;

- *middle* - указывает на то, что середина изображения должна находиться на уровне базовой линии текущей строки;

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Выравнивание изображения </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <h2 align=center> Иван Смирнов <br> Капитан команды </h2></
h2>
   <IMG src="face1.jpg" align=middle>
   Выпускник по специальности "Инноватика". Обладает 5-летним
практическим опытом руководства техническими проектами.

   </body>
</html>
```

Рис. 10.3. Результат применения атрибута align в значении middle

- *bottom* - при данном способе выравнивания нижняя граница изображения будет располагаться на базовой линии текущей строки.

Размеры изображения. Атрибуты width, height.

Благодаря использованию атрибутов **width**, **height** могут быть явно заданы размеры изображения, с которыми оно будет отображено в окне браузера. Следует помнить, что неправильное задание этих атрибутов может привести к искажению картинки. Можно задавать оба атрибута **width**, **height**, а можно указать только один из них. Недостающий размер изображения будет автоматически вычислен при загрузке из условия сохранения пропорций.

Задание размеров изображений позволяет достичь следующих результатов:

- последовательного представления документа, при котором сначала отображается текст, а потом изображения, под которые при первом проходе оставляется свободное место, с размерами, задаваемыми атрибутами **width**, **height**;
- возможность работы пользователя в режиме отключенной графики. При этом вместо изображений будет оставляться обведенное рамкой пустое пространство. В этом случае атрибуты **width**, **height** задают размеры это-

Рис. 10.4.

го пустого пространства, что позволяет сохранить авторское форматирование документа. В противном случае, вместо изображения будет выводиться маленькая иконка с крестиком, что приведет к неразберихе в документе, отформатированном с учетом размеров изображений.

Кроме этого, с помощью атрибутов **width** и **height** реализуются приемы, с помощью которых можно достичь экономии памяти, а значит и уменьшении времени загрузки документа. Например, очень часто на Web-страницах используются всякие горизонтальные разноцветные полоски. Так вот, благодаря наличию атрибутов **width** и **height** можно не включать в документ изображение всей полоски, а вставить маленькое изображение и растянуть его.

Например:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Примеры использования атрибутов WIDTH и HEIGHT для
создания полосок </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
```

```
   <h2 align=center> Примеры использования атрибутов WIDTH и
HEIGHT для создания полосок </h2>
   <center>
   <table>
      <tr>
         <th><img src="pol.jpg" border=1>
         <th> Исходное изображение
   </table>
   </center>
   <br><br>
   <table width=700 bgcolor=lightgrey cellpadding=5>
      <caption> Полоски, созданные на основе исходного изображения
</caption>
      <tr>
      <td><img src="pol.jpg" width=200 height=10 border=1>
      <tr>
      <td><img src="pol.jpg" width=400 height=10 border=5>

      <tr>
      <td><img src="pol.jpg" width=600 height=10 border=1>
      <tr>
      <td><img src="pol.jpg" width=200 height=20 border=1>
      <tr>
      <td><img src="pol.jpg" width=700 height=25 border=1>
   </table>
   </body>
</html>
```

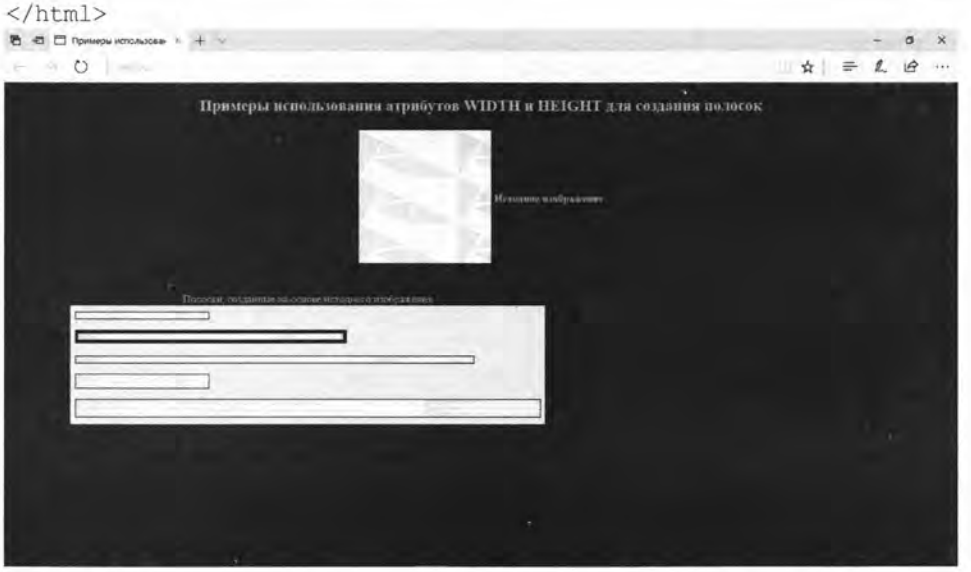


Рис. 10.5. Примеры использования атрибутов width и height для создания полосок

На основе такого применения атрибутов **width** и **height**, а также с использованием сценариев можно создавать динамически изменяющиеся гистограммы на Web-страницах.

Наличие у элемента IMG атрибутов **hspase** и **vspase** позволяет управлять размерами отступов между текстом и изображением. По умолчанию текст непосредственно примыкает к изображению, что вызывает некоторый дискомфорт восприятия. Наглядно положительный эффект применения атрибутов **hspace** и **vspace** можно увидеть на рис:

Рис. 10.6. Пример использования атрибута hspace

Изображение в HTML-документа может быть обведено рамкой. За наличие вокруг изображения рамки и ее ширину отвечает атрибут **border** тега IMG, значение которого задается в пикселях.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Примеры задания атрибута border </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <h2 align=center> Примеры задания атрибута border </h2>
   <center>
   <table>
```

```
      <tr>
         <th><img src="face1.jpg" width=200>
         <th><img src="face1.jpg" width=200 border=1>
      <tr>
         <th><img src="face1.jpg" width=200 border=5>
         <th><img src="face1.jpg" width=200 border=15>

   </table>
   </center>
   </body>
</html>
```

Рис. 10.7. Атрибут border

Альтернативный текст, заданный в качестве значения атрибута **alt** отображается при наведении курсора на изображение или вместо изображения, когда оно, ввиду каких-либо причин, не может быть показано.

10.3. Как вставить аудио и видео?

Для размещения аудио и видео объектов на HTML-странице, используются теги AUDIO и VIDEO соответственно. Путь к исходному файлу указывается через вложенный тег SOURCE или атрибут SRC.

Тег AUDIO

Используется для добавления аудиозаписи на веб-страницу. Настройки воспроизведения аудиофайла задаются с помощью следующих атрибутов:

- **autoplay** – автоматическое воспроизведение аудио;
- **controls** – позволяет добавить панель управления к аудиозаписи;
- **loop** – повторное воспроизведение аудиофайла после окончания его воспроизведения;
- **preload** – загружает аудиофайл вместе со страницей;
- **src** – указывает путь к источнику аудиофайла.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Добавление аудио </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
```

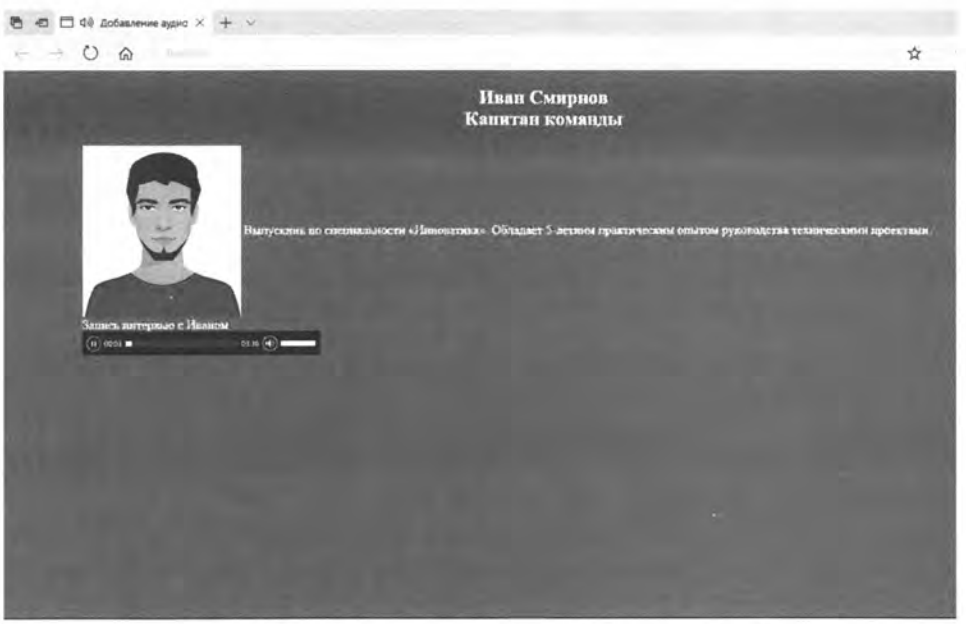


Рис. 10.8. Демонстрация страницы с аудиофайлом

```
   <h2 align=center> Иван Смирнов <br> Капитан команды </h2></
h2>
   <IMG src="face1.jpg" align=middle width=200>
   Выпускник по специальности "Инноватика". Обладает 5-летним
практическим опытом руководства техническими проектами.
   <br>
   Запись интервью с Иваном
   <br>
   <audio src="track01.mp3" controls="controls"
autoplay="autoplay">
   </audio>
   </body>
</html>
```

Тег VIDEO

Используется для добавления видеозаписи на веб-страницу. Настройки воспроизведения видеофайла задаются с помощью следующих атрибутов:

- **autoplay** – автоматическое воспроизведение видео;
- **controls** – позволяет добавить панель управления к видеозаписи.
- **loop** – повторное воспроизведение видеофайла после окончания его воспроизведения;
- **preload** – загружает видеофайл вместе со страницей
- **src** – указывает путь к источнику видеофайла.
- **height**, **width** – задают параметры области воспроизведения видеофайла

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Добавление видео </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <p>Что такое "TechRussia"?</p>
      <blockquote> Технологический марафон TechRussia – это
событие, направленное на популяризацию мира технологий.
<p>Будущие и начинающие инженеры, программисты, руководители
проектов и отраслевые специалисты соберутся на одной площадке,
чтобы раскрыть свой потенциал, познакомиться с новейшими
научными разработками и встретиться с профессионалами мира
технологий. </p>
```

```
    <br>
    Обзор мероприятия
    <br>
    <video width=400 height=300 src="hackrussia.mp4"
controls="controls" autoplay="autoplay">
    </video>
    </body>
</html>
```

Рис. 10.9. Демонстрация страницы с видеофайлом

Тег SOURCE

Использование тега SOURCE аналогично использованию атрибута SRC для тегов AUDIO и VIDEO. Данный тег позволяет указывать путь к исходному файлу, загружаемому на веб-страницу.

10.4. Как вставить другие мультимедиа-объекты?

Хоть в начале этого раздела и было сказано, что размещение объектов на Web-странице осуществляется тегом OBJECT, на самом деле сейчас для этих целей могут использоваться два HTML-тега: OBJECT и EMBED.

С элементом OBJECT связаны некоторые тонкости, на которых следует остановиться поподробнее. Этот элемент не просто вставляет в страницу файл мультимедиа. Элемент OBJECT создает внутри HTML- документа объект, содержимое которого находится во внешнем мультимедиа-файле. Таким образом, с точки зрения HTML сам мультимедиа-файл не является объектом. Он является содержимым объекта. Аналогично обстоит дело и с тегами: EMBED, IMG, да и вообще со всеми элементами, просто их содержимое задано в том же файле, что и они сами. Объект, будь то видеоклип или изображение, существует, вообще говоря, независимо от его содержимого.

Например, если для тега IMG указать несуществующий графический файл, то от этого сам он не перестанет существовать. Просто у него не будет содержимого. Аналогично и для тега OBJECT.

Это была теория. На практике, тому, кто ее не понял, сначала достаточно просто считать, что тег IMG вставляет изображения, тег IFRAME – фреймы, теги EMBED и OBJECT – объекты типа видеоклипов и звуковых фрагментов.

При задании объекта указывается тип его содержимого, в соответствии с которым браузер инициирует соответствующий подключаемый модуль, и месторасположение файла, который должен быть передан этому модулю. Дополнительные параметры файла задаются тегом PARAM внутри тега OBJECT. Все остальные теги, указанные между начальным и конечным тегами OBJECT, являются его альтернативным содержимым и используются только тогда, когда корректная обработка объекта является невозможной. Причиной последнего может быть невозможность доступа к файлу, либо отсутствие соответствующего ему подключаемого модуля. В качестве альтернативного содержимого тега OBJECT могут использоваться любые HTML-элементы, в том числе и сам тег OBJECT, заданный один внутри другого.

Задание начального и конечного тегов OBJECT является обязательным.

Список атрибутов тега OBJECT:

- **data** - указывает URL-адрес файла, в котором содержатся данные объекта;
- **classid** - используется в двух вариантах:
  - для указания URL-адреса подключаемого модуля ActiveX, и позволяет пользователю, если у него этот модуль отсутствует, скачать его из сети и проинсталлировать у себя на компьютере;
  - либо служит для подключения уже имеющегося управляющего элемента ActiveX, указывая его идентификационное имя, которое явля-

ется уникальным. В этом случае значением атрибута **classid** является шестнадцатеричный код, который использует Windows для идентификации управляющих элементов ActiveX. Информация обо всех установленных элементах ActiveX содержится в системном реестре Windows.

- **codebase** – указывает базовый URL-адрес для текущего объекта, относительно которого будут браться все относительные ссылки. Иными словами, задание атрибута **codebase** для объекта, аналогично заданию тега BASE для HTML-документа. Таким образом, если в значении атрибута data (или атрибута **classid**) указана относительная ссылка, то она будет браться относительно URL-адреса, указанного атрибутом **codebase**. Если этот атрибут отсутствует, то по умолчанию, в качестве базового адреса используется URL текущего документа;
- **codetype** – своим значением указывает тип содержимого данных, получения которых следует ожидать при загрузке объекта, задаваемого атрибутом **classid**. Этот атрибут не является обязательным, но рекомендуется, поскольку он позволяет браузеру не загружать информацию, тип содержимого которого он не поддерживает. Если этот атрибут отсутствует, то по умолчанию используется значение атрибута **type**;
- **type** – значением этого атрибута указывается MIME-тип содержимого мультимедиа-файла. Этот атрибут является необязательным, но его использование настоятельно рекомендуется. Задание атрибута **type** позволяет браузеру вообще загружать мультимедиа-файлы, обработать которые он не в состоянии. Иначе он сначала загрузит этот файл, а потом попытается определить его MIME-тип;
- **border** – атрибут, задающий толщину рамки в пикселах вокруг объекта. По умолчанию рамка вообще отсутствует;
- **height** – задает вертикальный размер прямоугольной области в пикселах, отводимой под данный объект в документе;
- **width** – ширина прямоугольной области в пикселах, отводимой под данный объект в документе;

Примечание: отображаемая информация в объекте при этом изменяется таким образом, чтобы полностью заполнить отведенную под него прямоугольную область. Если атрибуты height и width для объекта не заданы, браузер сам автоматически определяет их, исходя и размеров содержимого объекта.

- **hspace** – атрибут, указывающий размеры в пикселах горизонтальных отступов между объектом и остальным содержимым документа;
- **vspace** - атрибут, указывающий размер в пикселах вертикальных отступов между объектом и остальным содержимым документа;
- **align** –выравнивание объекта в документе. Как и для тега IMG может принимать следующие значения:
  - *left* – объект прикрепляется к левому краю документа
  - *right* – объект прикрепляется к правому краю документа
  - *top* – выравнивание по верхней границе текущей строки текста
  - *middle* – середина объекта должна находиться на уровне базовой линии строки текста
  - *bottom* – выравнивание нижней границы объекта производится по базовой линии текущей строки. Это значение используется по умолчанию
- **id** – задает имя объекта в пределах документа;
- **tabindex** – номер объекта в последовательности перехода по клавише "Tab";
- **style** – стилевые установки объекта;
- **usemap** – этот атрибут связывает данный объект с навигационной картой. В качестве значения этого атрибута указывается имя навигационной карты;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

С помощью тегов PARAM задается список параметров и их значений, которые будут переданы подключенному модулю для задания параметров обработки содержимого объекта. Внутри одного элемента OBJECT может быть определено любое количество тегов PARAM и в любом порядке. Начальный тег PARAM является обязательным, конечный тег – запрещен.

Список атрибутов:

- **name** – в качестве значения этого атрибута указывается имя параметра, которое должно быть понятно подключенному модулю. Учитывает ли это имя регистр, определяется модулем, которому он предназначен;

- **value** – задает значение параметра, чье имя задано атрибутом **name**. Принимаемые атрибутом **value** значения не определены в HTML, а определяются подключенным модулем;
- **valuetype** – указывает тип атрибута **value**. Может принимать следующие значения:
  - *data* – задание этого типа говорит о том, что указанное для атрибута **value** значение будет определяться и передаваться подключенному модулю в виде строки. Это значение используется по умолчанию.
  - *ref* – указывает на то, что значением атрибута **value** является URL-адрес файла, в котором хранятся значения параметров
  - *object* – реализует ссылку на другой объект в этом же документе. В качестве значения указывается **id** другого объекта.
- **type** – указывает тип содержимого (content-type) ресурса, чей URL-адрес указан атрибутом **value**. Используется только в том случае, если для атрибута **valuetype** указано значение "*ref*"
- **id** – задает имя элемента в пределах документа.

Вообще возможны три варианта обработки тега OBJECT и его содержимого браузером:

- браузер может определить тип содержимого объекта и у него имеется необходимый подключенный модуль. В этом случае браузер включает объект в документ и обрабатывает все теги PARAM, заданные в содержимом тега OBJECT. Прочие элементы, имеющиеся в содержимом этого элемента и являющиеся его альтернативным содержимым, игнорируются.
- браузер может определить тип содержимого объекта, но у него отсутствует необходимый подключаемый модуль. В такой ситуации, браузер пытается, с предварительного согласия пользователя, подгрузить требуемый модуль с адреса, указанного атрибутом **classid** или в установках самого браузера. Если ему это удается, то ситуация переходит в описанную выше. Если браузеру не удается подгрузить нужный модуль, то тег OBJECT и содержащиеся в нем теги PARAM игнорируются. Отображению подлежат все остальные элементы, находящиеся в содержимом тега OBJECT и являющиеся его альтернативным содержимым.
- браузер изначально не может определить тип содержимого объекта. Здесь все совсем просто: тег OBJECT вместе со своими тегами PARAM игнорируется, а отображается его альтернативное содержимое.

В качестве альтернативного содержимого тега OBJECT может быть использован другой тег OBJECT. При этом тег PARAM альтернативного тега OBJECT игнорируются первичным тегом OBJECT.

Тег EMBED

Использование этого тега является предпочтительнее ввиду более корректной поддержки его наибольшим количеством браузеров, и более простой работой с ним.

При использовании тега EMBED задание начального тега является обязательным, конечный тег запрещен.

Список атрибутов тега EMBED:

- **src** – указывает URL-адрес мультимедиа-файла, в котором хранятся данные объекта;
- **type** – значением этого атрибута указывается MIME-тип содержимого мультимедиа-файла. Этот атрибут является необязательным, но его использование настоятельно рекомендуется. Задание атрибута **type** позволяет браузеру вообще загружать мультимедиа-файлы, обработать которые он не в состоянии. Иначе он сначала загрузит этот файл, а потом попытается определить его MIME-тип;
- **border** – атрибут, задающий толщину рамки в пикселах вокруг объекта;
- **frameborder** – логический атрибут, который указывает наличие или отсутствие рамки вокруг объекта. Может принимать значения "Yes" и "No". По умолчанию используется значение "No";
- **height** – задает вертикальный размер прямоугольной области в пикселах, отводимой под данный объект в документе;
- **width** – ширина прямоугольной области в пикселах, отводимой под данный объект в документе;
- **hspace** – атрибут, указывающий размеры в пикселах горизонтальных отступов между объектом и остальным содержимым документа;
- **vspace** - атрибут, указывающий размер в пикселах вертикальных отступов между объектом и остальным содержимым документа;
- **hidden** – логический атрибут, наличие которого делает объект невидимым;

- **pluginspage** – указывает URL-адрес ресурса, в котором содержатся инструкции по инсталляции модуля;
- **align** - выравнивание объекта. Принимаемые значения такие же, как и для тега OBJECT;
- **id** – задает имя объекта в пределах текущего документа.

Пример простейшего использования:

```
<!DOCTYPE html>
<html>
   <head>
   <title>
   Пример простейшего использования элемента EMBED
   </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center>
         Всероссийский хакатон hackRussia 2016
```

Рис. 10.10. Пример документа с простейшим использованием элемента EMBED

```
      </H2>
      <HR>
      <HR color=lightgrey size=20>
      <CENTER>
         <EMBED src="hackrussia.mp4">
      </CENTER>
   </BODY>
</HTML>
```

С помощью атрибутов **width** и **height** нельзя добиться искажения видеоклипа, как это можно сделать с изображением. Размеры кадра видеоклипа изменяются пропорционально друг друга, то есть если один из них, например, увеличивается в два раза, то и второй должен увеличиться в два раза. Если второй этого сделать не может из-за ограничения пространства, то и первый тогда увеличиваться не будет. Проще говоря, с помощью атрибутов **width** и **height** можно осуществлять только увеличение или уменьшения масштаба видеоизображения.

Например, если в предыдущем примере строку

```
<EMBED src="hackrussia.mp4">
```

заменить на строку

```
<EMBED src="hackrussia.mp4" width=100>
```

то получится следующее:

Рис. 10.11. Использование атрибута width в значении 100

Если заменить на

```
<EMBED src="hackrussia.mp4" width=500>
```

то получится следующее:

Рис. 10.12. Использование атрибута width в значении 500

Чтобы гарантированно увеличить видеоизображение надо задать оба атрибута: **width** и **height**.

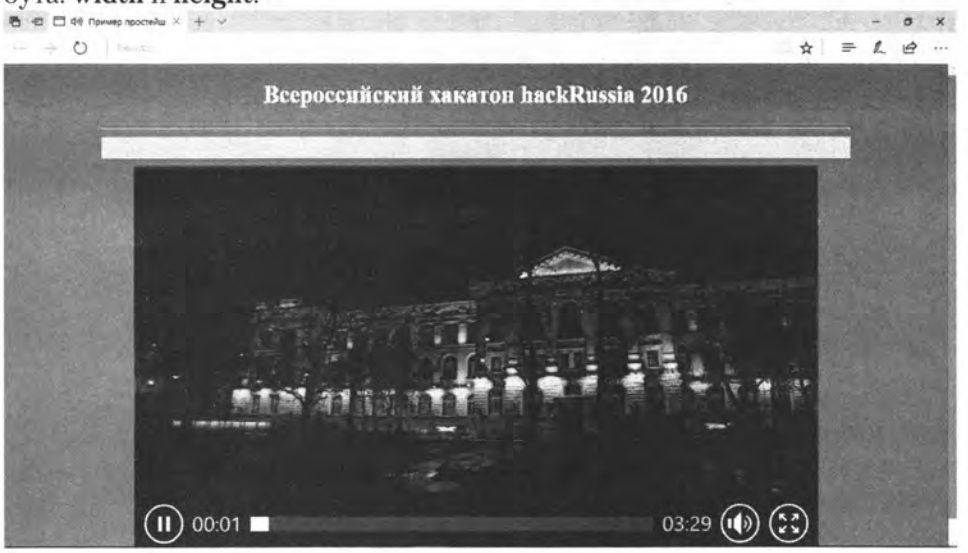


Рис. 10.13. Пример использования атрибутов width и height для увеличения видеоизображения

Параметры, передаваемые подключенному модулю вместе с мультимедиа-файлом указываются в самом теге элемента EMBED. Например:

```
<EMBED src= "hackrussia.mp4" border=2 autostart=false>
```

В этом случае подключенному модулю будет передан параметр **autostart** со значением *false*. При использовании элемента OBJECT для этих же целей понадобилась бы следующая запись:

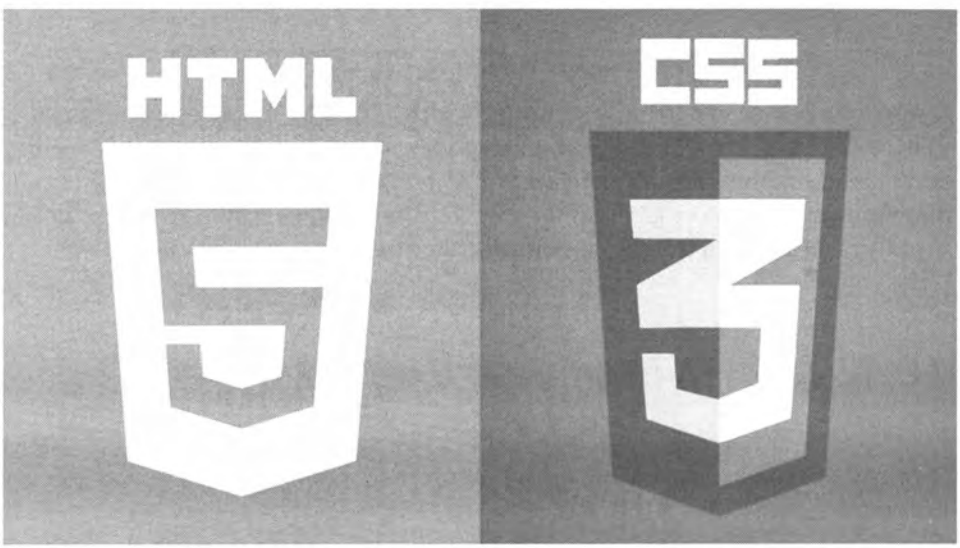
```
<OBJECT data= "hackrussia.mp4"border=2>
<PARAM name="autostart" value="false">
</OBJECT>
```

Стоит отметить, что атрибуты **align**, **width** и **height** применительно к звуковым файлам, относятся к панели управления, задавая, соответственно, ее выравнивание, ширину и высоту. Однако размеры консоли являются фиксированными для каждого ее вида. Поэтому пространство, выделяемое атрибутами **width** и **height** должно быть не меньше размеров консоли.

10.5. Как группировать объекты?

Для группировки любых объектов, например, изображений и подписей к ним используется тег FIGURE.

Чтобы задать подпись к изображению необходимо использовать тег FIGCAPTION.


HTML
5
CSS
3

HTML5

Глава 11.

Макет страницы и навигационные карты

В этой главе вы узнаете:

- *Что структура страницы?*
- *Что такое навигационные карты-изображения?*
- *Что такое серверные навигационные карты?*
- *Как создать клиентскую навигационную карту?*

11.1. Структура страницы

Для создания "шапки" страницы, иными словами раздела, в котором находится заголовок, используется тег HEADER. Открывающий и закрывающий теги обязательны.

Пример:

```
<body>
   <header>
      <h1> Добро пожаловать на сайт </h1>
   </header>
….текст сайта…
</body>
```

Для создания "футера" или "подвала" страницы используется тег FOOTER. В этом разделе обычно отображается контактная информация.

Для задания содержания страницы, например, новостей, статей, используется тег ARTICLE.

Разделы документа могут быть обозначены тегом SECTION. Допускается использование тегов SECTION, вложенных друг в друга.

Для создания боковой панели или "сайдбар", в которой зачастую отображается меню, основные разделы сайта, используется тег ASIDE.

Основное содержимое документа обычно располагается в теге MAIN.

Для навигации по сайту используется тег NAV, внутри которого располагаются ссылки.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Команда "Победители во всём!" </title>
      <base href= "http://techrussia.org">
      <meta charset="UTF-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <header>
         <h1> Информация о команде "Победители во всём!" </h1>
      </header>
      <section>
         <h2> Иван Смирнов <br> Капитан команды </h2>
         <p> Выпускник по специальности "Инноватика". Обладает
5-летним практическим опытом руководства техническими
проектами. </p>
      </section>
      <section>
         <h2> Пётр Демидов </h2>
         <p> Окончил МГТУ им. Баумана по специальности
"Робототехника". Имеет огромный опыт конструирования роботов.
</p>
      </section>
      <section>
         <h2> Елизавета Казакова </h2>
         <p> Окончила Санкт-Петербургский политехнический
университет по направлению "Графический дизайн". После
знакомства с Петром стала активно интересоваться промышленным
дизайном. </p>
      </section>
      <section>
         <h2> Алексей Сидоров </h2>
         <p> Отвечает за программирование роботов. </p>
      </section>
   <p class="ssyl"><a href="/#directions"> Ссылка на
направления TechRussia </a><p>
   <div id="cite"> "Мы живем в эпоху, когда расстояние от самых
безумных фантазий до совершенно реальной действительности
сокращается с невероятной быстротой." Максим Горький </div>
```

```
    <footer>
      <form action="mailto:partners@a-techevent.ru"
method="post"enctype="text/plain">
      <p><input type="submit" value="Отправить письмо
организаторам"></p>
      </form>
    </footer>
    </body>
</html>
```

11.2. Что такое навигационные карты-изображения?

Принцип работы навигационной карты заключается в использовании различных частей одного изображения для организации ссылок. При этом гипертекстовые ссылки привязываются к определенным областям изображения. Использование специально разработанных для этих целей изображений позволяет значительно увеличить привлекательность и эффективность навигационного интерфейса. Иными словами, карты-изображения представляют собой графическую альтернативу текстовым меню.

При работе с навигационными картами пользователю предлагается интуитивно понятный, дружественный интерфейс. Однако все вышеперечисленное еще не означает, что навигационные карты должны использоваться везде, где присутствуют ссылки. Использование больших изображений замедляет загрузку документа и увеличивает время ожидания первого представления документа. Поэтому в целях увеличения производительности рекомендуется их не использовать. К тому же это изображение должно быть разработано так, чтобы пользователь мог разобраться, какая его часть на что ссылается. Поэтому, частое использование обычного текста для привязки к нему ссылок, является более эффективными и понятным. К тому же не следует забывать о возможности использования пользователем режима отключенной графики. Надо помнить, что отдельные области карты-изображения должны выделяться (например, рамкой или цветовым контрастом), иначе пользователь может в них запутаться.

Таким образом, можно выделить следующие преимущества и недостатки использования навигационных карт.

Преимущества:

- создание наглядного, интуитивно понятного интерфейса;

- более эффективная (по сравнению с текстовыми ссылками) реализация ссылок, связанных между собой пространственно. Например, выбрать определенную страну щелкнув по ее изображению на карте гораздо проще, чем выбрать ее из списка стран.

Недостатки:

- увеличение времени загрузки Web-документа и, соответственно, снижение производительности.

Создание навигационной карты происходит путем назначения изображения с указанием соответствующих геометрических областей. Используемые для этих целей изображения могут иметь любой допустимый графический формат: jpeg, gif, png и т.п.

К изображениям, служащими основой навигационных карт, можно предложить следующие рекомендации:

- Файл изображения должен иметь желательно формат GIFJPEG
- При создании изображения в виде gif-файла, можно рекомендовать сохранять его в этом формате с использованием чередованием строк. Это позволит пользователю гораздо быстрее увидеть изображение пусть и в нечетком, схематичном виде, но еще до того, как оно загрузится полностью. По мере загрузки графического файла, изображение будет принимать свой оригинальный вид
- Следует по возможности уменьшать размер памяти, занимаемой изображением. Это осуществимо путем уменьшения разрешения изображения и количества присутствующих в нем цветов.

Применимость вышеуказанных рекомендаций не носит абсолютный характер и определяется разработчиком в соответствии с особенностями проектируемого им документа.

Существует два типа навигационных карт:

- клиентская (client-side)
- серверная (server-side)

Рассмотрим более подробно оба эти типа.

11.3. Что такое серверные навигационные карты?

При использовании серверной навигационной карты, когда пользователь щелкает мышью на какой-либо части изображения, координаты курсора передаются серверу. Последний определяет, к какой выделенной области эти координаты принадлежат, и какая ссылка (URL-адрес) ей соответствует. Затем сервер пересылает URL-адрес клиентскому браузеру, который ее и открывает. Когда пользователь щелчком мыши активизирует какую-либо область клиентской навигационной карты, координаты курсора обрабатываются браузером на стороне клиента. Он сам определяет ссылку, привязанную к активизированной области изображения, и открывает ее.

Применение серверных навигационных карт имеет смысл в том случае, когда карта слишком сложна. Причем, связать серверную карту можно только с содержимым тега IMG. В случае с тегом IMG, он должен быть задан внутри элемента A. Для тега IMG должен быть установлен логический атрибут **ismap**.

Для реализации серверной навигационной карты необходимо чтобы на сервере, помимо самого HTML-документа, была сконфигурирована поддержка CGI-сценариев. Они обрабатывают данные, поступающие на сервер и являющиеся результатом взаимодействия пользователя с серверной навигационной картой. К тому же на сервере, для каждой навигационной карты должен находиться соответствующий ей файл, в котором определены активные области с привязанными к ним ссылками.

Таким образом, схема работы серверной навигационной карты принимает следующий вид: при щелчке мыши в пределах изображения карты, браузер пересылает координаты курсора серверу. Затем сервер, с помощью CGI-сценариев, обращается к конфигурационному файлу активных областей, в соответствие с которым определяет принадлежность присланных координат к какой-либо активной области. Результатом такой проверки является URL-адрес целевого ресурса ссылки, привязанной к активной области, которой принадлежат присланные координаты. Если такой активной области не найдено, то в качестве результата выдается соответствующее сообщение. Результат проверки пересылается браузеру, который либо открывает документ по присланному URL-адресу, либо игнорирует щелчок мыши, если нет соответствующей ему активной области.

Существует два формата конфигурационных файлов активных областей: CERN и NSCA, созданные одноименными создателями. В обоих случаях в

файле содержится одна и та же текстовая информация. Различие заключается в ее преставлении (порядке задания в пределах файла). И хотя файлы обоих форматов поддерживают одинаковые типы активных областей (rect, circle, poly и default), они являются несовместимыми.

11.4. Как создать клиентскую навигационную карту?

Создание клиентской навигационной карты аналогично созданию серверной. Отличие заключается только в том, что конфигурация активных областей осуществляется не в отдельном конфигурационном файле, а непосредственно в HTML-документе. И для них не нужен CGI-сценарий. Браузер сам обрабатывает навигационную карту.

Применение клиентских навигационных карт предпочтительно из-за быстроты и простоты их работы. К тому же они позволяют незамедлительно определять, находится ли курсор в активной области (как в случаях с текстовыми ссылками курсор из "стрелки" превращается в "ладошку").

Описание активных областей клиентских навигационных карт осуществляется в содержимом тега MAP с помощью использования тегов AREA и имеет следующий синтаксис:

```
<MAP name="имя навигационной карты">
<AREASHAPE= "форма_области"COORDS="x, y,.....координаты"
HREF="URL-адрес целевого документа">
<AREASHAPE="форма_области"COORDS="x, y,.....координаты"
HREF="URL-адрес целевого документа">
.....
<AREASHAPE="форма_области"COORDS="x, y,.....координаты"
HREF="URL-адрес целевого документа">
</MAP>
<IMG src="место расположения изображения" USEMAP=#имя_
навигационной_карты>
```

Тег MAP является контейнером, поэтому задание начального и конечного тегов при его использовании, является обязательным. Тег MAP имеет только один атрибут, являющийся обязательным:

- **name** – задает имя навигационной карте.

Установление связи каких-либо HTML-элементов с навигационной картой осуществляется с помощью атрибута **usemape**, задаваемого для этих элементов. В качестве значения этого атрибута указывается имя навигационной карты, с которой устанавливается связь.

Например:

```
<MAP name="My General Map">
<AREA ..... .>
<AREA ..... .>
.....
<AREA ..... .>
</MAP>
<IMG src="map1.jpg" usemap=# My General Map>
```

Каждая активная область задаётся индивидуальным тегом AREA.

Тег AREA имеет следующие атрибуты:

- **shape** - этот атрибут указывает какую форму имеет активная область. Возможные значения:
  - *rect* - активная область будет имеет форму прямоугольника. Данное значение используется по умолчанию
  - *circle* - активная область будет выполнена в виде круга
  - *poly* - активная область будет иметь форму многоугольника
  - *default* - в качестве активной области принимается все изображение

  Примечание. Некоторые браузеры, в том числе и Internet Explorer, не поддерживают атрибут shape в значении default. Поэтому для этих целей рекомендуется использовать значение rect, с размерами всего изображения

- **coords** - этот атрибут указывает положение и размеры активной области на экране. Задание и интерпретация его значений зависит от формы активной области. Исходя из этого, возможны следующие варианты:
  - Для прямоугольной активной области (shape = rect) в качестве значения атрибута **cords** указывается через запятую: X-координата и У-координата левого верхнего угла, X-координата и У-координата правого нижнего угла.

- Для круглой активной области (shape = circle) в качестве значения атрибута **coords** через запятую указываются: X и У–координаты центра круга и его радиус.
- Для активной области в виде многоугольника (shape = poly) значением атрибута **coords** будет являться список координат вершин : X1, Y1, X2, Y2, X3, Y3, ..., Xn, Yn. Причем последняя вершина в списке не обязательно должна совпадать с первой. Браузер сам их соединит.

Примечание. Все координаты задаются в пикселах относительно левого верхнего угла изображения.

- **nohref** - логический атрибут, задание которого указывает браузеру на то, что с активной областью не связана ни одна ссылка.
- **href** - указывает URL-адрес целевого ресурса ссылки, прикрепленной к данной активной области.

  Примечание. Если для активной области, задаваемой тегом AREA не указаны оба атрибута href и nohref, то по умолчанию браузером используется атрибут nohref.

- **target** – атрибут, употребляемый при работе с фреймами. При этом он указывает имя фрейма, в котором должен отображаться документ, который открывается по ссылке, связанной с данной активной областью.

Примечательно, что активные области могут перекрываться. При этом высший приоритет имеет область, чье задание было раньше по HTML-тексту документа.

В этой связи задание shape = default (или shape = rect с последующим указанием координат границ изображения) рекомендуется использовать после задания всех остальных активных областей. Тогда благодаря этому будет определена активная область, в которую будет входить все пространство изображения, не вошедшее в остальные активные области.

Для связи изображения с навигационной картой используется атрибут usemap тега IMG. В качестве его значений указывается имя (значение атрибута name) навигационной карты. Стоит отметить, что одна навигационная карта может быть связана с несколькими изображениями (или объектами).

Пример задания навигационной карты:

```
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE> Пример задания навигационной карты </TITLE>
</HEAD>
<BODY leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <MAP name="My Map">
   <AREA shape="rect" coords="0,0,120,80" href="http://
techrussia.org/">
   </MAP>
   <IMG src="map.jpg" usemap=#My Map>
</BODY>
</HTML>
```

HTML5

Глава 12.

Фреймы

В этой главе вы узнаете:

- *Зачем использовать фреймы?*
- *Как создать фрейм?*

12.1. Зачем использовать фреймы?

Использование фреймов (frame – рамка) при разработке документа HTML позволяет авторам представлять его в нескольких окнах, на которые разбивается окно браузера. При этом все эти окна видимы. Благодаря такому качеству достигается основное преимущество использования фреймов, а именно: обеспечивается постоянная видимость некоторой информации, в то время как другая прокручивается или загружается. На практике эта возможность обычно реализуется в следующем исполнении (особенно популярно среди начинающих разработчиков): в одном фрейме (как правило, левом) размещается навигационное меню, а во втором, занимающем все остальное пространство, фрейм с открывающимися по ссылкам из меню страницами.

Фреймы также часто используются для статического размещения баннеров в окне пользователя.

Вообще принцип работы фреймов аналогичен принципу работы таблиц. Если таблицы организуют информацию в документе путем разбивки его на насколько областей (ячеек), то фреймы служат для разбивки окна браузера на несколько окон (фреймов). Информация, помещаемая в каждый из фреймов, представляет собой отдельный HTML-файл. Таким образом, основной, корневой HTML-документ представляет собой информацию о количестве и размерах фреймов, а также содержит сведения о том, в какой фрейм какой файл загружать.

12.2. Как создать фрейм?

Язык HTML позволяет вставлять фреймы в документ, не производя разбивки всего документа на фреймы. Другими словами, фрейм может быть вставлен в обычный текстовый документ, как один из его элементов (то есть точно так же, как вставляются, например, изображения в документы). В интернете этот прием используется часто для включения инструкций, текста соглашения, информативных списков и т.п.

Реализуется эта возможность с помощью элемента разметки IFRAME

Элемент IFRAME

Начальный тег обязателен, конечный может не задаваться.

Список атрибутов:

- **name** – задает имя фрейма
- **width** – ширина встраиваемого фрейма
- **height** – высота встраиваемого фрейма
- **scr** – задает URL-адрес информационного ресурса загружаемого в данный фрейм
- **align** – задает горизонтальное и вертикальное выравнивание фрейма в рамках документа. С помощью этого атрибута, опять же, осуществляется привязка фрейма к краям документа, что делает возможным обтекание его текстом
- **scrolling** – указывает браузеру о наличии у данного фрейма полосы прокрутки. Принимает значения:
  - *yes* – полоса прокрутки отображается в любом случае;
  - *no* - полоса прокрутки не отображается в любом случае;
  - *auto* – полоса прокрутки отображается при необходимости. Это значение используется по умолчанию.
- **marginwidth** – задает пространство в пикселах, оставляемого во фрейме в качестве левого и правого полей. Другими словами, указывает размер отступа содержимого фрейма от его боковых границ. Значение должно превышать один пиксел. Значение по умолчанию задается браузером.

- **marginheight** – задает размер в пикселах верхнего и нижнего полей отступа фрейма. Значение должно превышать один пиксел. Значение по умолчанию задается браузером.
- **frameborder** – задает вид границ вокруг данного фрейма. Возможные значения:
  - *1* – браузер должен отображать объемные границы между текущим и всеми прилегающими к нему фреймами. Используется по умолчанию.
  - *0* – браузер должен изображать разделители плоскими: в виде постой линии определенного цвета и толщины

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
   <title> Пример документа с вложенным фреймом </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <H2 align=center>
   Пример обычного текстового документа со встроенным фреймом
</H2>
   <DIV align=center>
   ..... текст документа ..... .<BR>
   ..... текст документа ..... .<BR>
   ..... текст документа ..... .<BR>
   ..... текст документа ..... .<BR><BR>
   <IFRAME src="http://techrussia.org" width="400" height="200"
           scrolling="yes">
   </IFRAME>
   <P>
   ..... текст документа ..... .<BR>
   ..... текст документа ..... .<BR>
   ..... текст документа ..... .<BR>
   </DIV>
   </BODY>
</HTML>
```

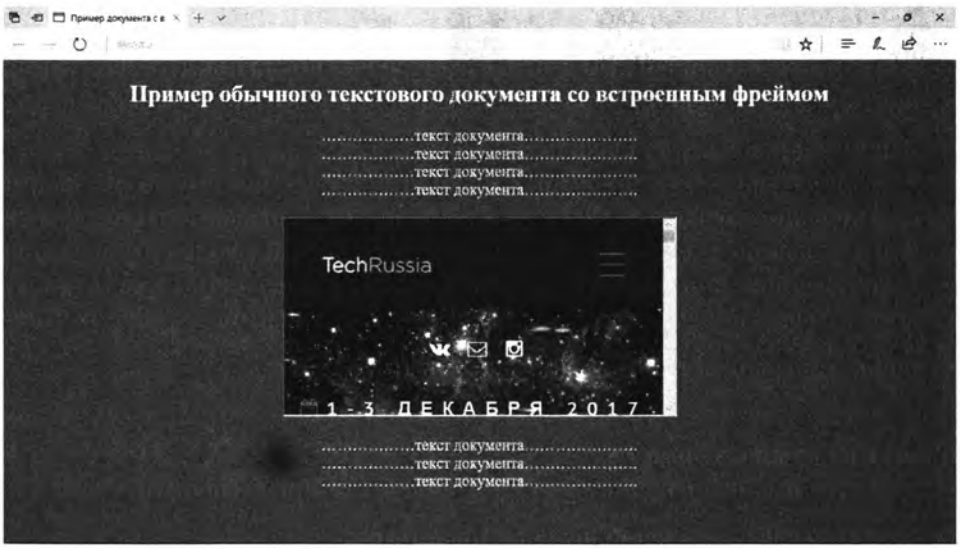


Рис. 12.1. Пример документа с вложенным фреймом

HTML5

Глава 13.

Формы

В этой главе вы узнаете:

- *Что такое формы и для чего они используются?*
- *Как использовать тег FORM?*
- *Как использовать тег SELECT?*
- *Как использовать тег OPTION?*
- *Как использовать тег TEXTAREA?*
- *Как использовать тег INPUT?*
- *Как использовать тег Button?*
- *Как использовать тег LABEL?*
- *Как использовать теги LEGEND и FIELDSET?*
- *Как использовать тег DATALIST?*
- *Как управлять фокусом?*

13.1. Что такое формы и для чего они используются?

Формы в HTML-документах используются для пересылки данных серверу от удаленного пользователя. Таким образом, организуется простейший диалог клиент-сервер, например: on-line регистрация пользователя на сервере (заполнение анкеты), внесение записи в гостевую книгу, реализация покупательской корзины в on-line магазине.

Формы состоят из одного или нескольких полей ввода, в которые пользователь заносит информацию вручную или выбирает из предложенного списка значений. После заполнения формы она отсылается серверу и обрабатывается серверными скриптами. Скрипты при этом могут быть самые разные. Один документ может содержать несколько форм. Пользователь взаимодействует с формами с помощью именованных управляющих элементов.

Управляющие элементы

Форма в HTML-документе реализуется тегом-контейнером FORM, в котором задаются все управляющие теги. Если управляющие теги указаны вне содержимого тега FORM, то они не создают форму, а используются для построения пользовательского интерфейса на Web-странице, то есть для привнесения в нее различных кнопок, флажков, полей ввода, обработка которых производится индивидуально, в рамках самого HTML-документа, с

помощью включенных в него клиентских скриптов. А могут вообще никак не обрабатываться. Например, управляющий тег TEXTAREA часто используется для создания окна с полосой прокрутки внутри документа для вывода большого текста, который играет второстепенную роль. Обычно так отображаются тексты лицензионных соглашений, тексты больших комментариев или правила пользования данным Web-ресурсом. Выглядит это так же, как и встроенный фрейм.

Имена управляющим тегам присваиваются через их атрибут name.

Каждый управляющий тег имеет начальное, используемое по умолчанию, и конечное значения, которые являются символьными строками. Начальные значения управляющих тегов не меняются, благодаря чему может осуществляться сброс значений, указанных пользователем. Результатом этого действия будет установка всех управляющих тегов формы в своих первоначальных, используемых по умолчанию, значениях.

В HTML 5 определены следующие типы управляющих элементов:

- **кнопки** – задаются с помощью тегов Button и Input. Различают:
  - *кнопки отправки* - при нажатии на них осуществляют отправку формы серверу;
  - *кнопки сброса* – при нажатии устанавливают управляющие элементы в первоначальные значения;
  - *прочие кнопки* - кнопки, для которых не указано действие, выполняемое по умолчанию при их нажатии.

- **радиокнопки (кнопки с зависимой фиксацией)** - задаются тегом INPUTи представляют собой переключатели "вкл \ выкл". Если несколько радиокнопок заданы управляющими элементами с одинаковыми именами, то они являются взаимоисключающими. Это значит, что если одна из них ставится в положение "вкл", то все остальные автоматически в положение "выкл". Именно это и является преимуществом их использования
- **флажки** - задаются элементом INPUT и представляют собой переключатели "вкл \ выкл", но в отличие от радиокнопок, "включенные" флажки могут иметь одинаковые имена. Примечание: имена задаются атрибутом name
- **меню** – реализуется с помощью тегов SELECT, OPTGROUP и OPTION. Меню предоставляют пользователю список возможных вариантов выбора

- **ввод текста** - реализуется тегами INPUT, если вводится одна строка, и тегами TEXTAREA- если несколько строк. В обоих случаях введенный текст становится текущим значением управляющего элемента
- **выбор файлов** - позволяет вместе с формой отправлять выбранные файлы, реализуется HTML-тегом INPUT
- **скрытые управляющие элементы** - создаются управляющим тегом INPUT

13.2. Как использовать тег FORM?

Тег FORM указывает адрес, куда будет послана форма, способ пересылки и характеристику данных, содержащихся в форме. Тег FORM своим начальными и конечными тегами задает границы формы, поэтому их указание является обязательным. Без тега FORM нет формы.

Список атрибутов тега FORM:

- **action** - обязательный атрибут. В качестве его значения указывается URL-адрес запрашиваемой CGI-программы, которая будет обрабатывать данные, содержащиеся в форме. Допустимо использовать запись mailto:URL, благодаря которой форма будет послана по электронной почте. Если атрибут **action** все-таки не указан, то содержимое формы будет отправлено на URL-адрес, с которого загружалась данная Web-страница;
- **method** - определяет метод HTTP, используемый для пересылки данных формы от браузера к серверу. Атрибут **method** может принимать два значения:
  - *get* - при использовании HTTP-метода "get"данные формы пересылаются в первой строке HTTP-заголовка. В этом случае они добавляются к URL, указанному атрибутом Action, и пересылаются вместе с ним. Присоединение осуществляется с помощью символа "?", после которого следуют данные формы в виде последовательности пар "имя управляющего элемента = значение", разделенных символом "&". При получении данных сервер отделяет их от URL и присваивает переменной окружения Query-string, которая и используется CGI–программой. Метод *get* используется по умолчанию.
  - *post* - при использовании HTTP - метода "post" набор данных формы пересылается серверу в тексте HTTP-сообщения. Метод " get" рекомендуется использовать, если форма не вызывает каких-либо измене-

ний на сервере (изменяет базу данных, вносит в реестр подписчиков на услуги т.п.). Идеальным приложением метода "get" является поиск в базе данных. В противном случае рекомендуется использовать метод "post". К тому же при использовании метода "get" в полях формы допустимо написание только символов ASCCI.

- **enctype** - указывает тип содержимого формы, используемый для определения формата кодирования при ее пересылке. В HTML определены три возможных значения для атрибута enctype: application/x-www-form-urlencoded (используется по умолчанию), multipart/form-data, text/plain;
- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, onkeypress, onkeydown, onkeyup – onsubmit, onreset – внутренние события.

Пример задания:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Команда "Победители во всём!" </title>
      <base href= "http://techrussia.org">
      <meta charset="UTF-8">
      <style type="text/css">
      h1{color:#fe4918; border-width:3; border-style:solid;
text-align:center; border-color:white}
      #cite{text-align:right; color:#45525b; font-style:italic}
      p.ssyl{text-align:center}
      </style>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232 scroll=no link=#fe4918 vlink=#ffffff
alink=#ffffff>
      <h1> Информация о команде "Победители во всём!" </h1>
```

```
    <h2> Иван Смирнов <br> Капитан команды </h2>
  <p> Выпускник по специальности "Инноватика". Обладает
5-летним практическим опытом руководства техническими
проектами.</p>
    <h2> Пётр Демидов </h2>
  <p> Окончил МГТУ им. Баумана по специальности
"Робототехника". Имеет огромный опыт конструирования
роботов.</p>
    <h2> Елизавета Казакова </h2>
  <p> Окончила Санкт-Петербургский политехнический университет
по направлению "Графический дизайн". После знакомства с Петром
стала активно интересоваться промышленным дизайном.</p>
    <h2> Алексей Сидоров </h2>
  <p> Отвечает за программирование роботов. </p>
  <p class="ssyl"><a href="/#directions"> Ссылка на
направления TechRussia </a><p>
```

Рис. 13.1. Пример использования тега FORM

```
  <divid="cite"> "Мы живем в эпоху, когда расстояние от самых
безумных фантазий до совершенно реальной действительности
сокращается с невероятной быстротой." Максим Горький </div>
  <form action="mailto:partners@a-techevent.ru"
method="post"enctype="text/plain">
  <p><input type="submit" value="Отправить письмо
организаторам"></p>
  </form>
```

```
    </body>
</html>
```

13.3. Как использовать тег SELECT?

В формах HTML-документа выбор из списка предлагаемых значений может осуществляться двумя способами: либо с использованием радиокнопок и флажков, либо с использованием тега SELECT. Оба способа имеют свои достоинства и недостатки: радиокнопки и флажки применяются для осуществления выбора в том случае, если вариантов выбора немного. В противном случае, большое количество радиокнопок и флажков сделает форму громоздкой и неудобной в работе. Для реализации выбора из многовариантного списка значений применяется тег SELECT. Его основное преимущество - компактность, является также и недостатком, т.к. плотно скомпонованный список в исполнении тега SELECT выглядит менее привлекательно, чем список, созданный с применением флажков и радиокнопок. Другим недостатком тега SELECT является то, что множественный выбор (выбор сразу нескольких значений) может содержать в себе только рядом расположенные в списке варианты. Флажки позволяют осуществлять множественный выбор среди элементов списка независимо от порядка их следования. Тег SELECT отображается браузерами либо в виде ниспадающего меню, либо в виде списка с вертикальной полосой прокрутки.

Начальный и конечный теги SELECT являются обязательными. Этот тег имеет следующие атрибуты:

- **name**–присваивает имя содержащему его элементу;
- **size** – задает число одновременно видимых элементов выбора. В качестве значений принимает натуральные числа. Если атрибут **size** установлен в значении *1*, то список предлагаемых значений выбора изображается в виде ниспадающего меню. В том случае, если значение атрибута **size** меньше количества элементов списка выбора, то при его отображении используется вертикальная полоса прокрутки;
- **multiple** – логический атрибут, указание которого позволяет осуществлять множественный выбор сразу нескольких предлагаемых вариантов. Если этот атрибут отсутствует, то, по умолчанию, множественный выбор в данном списке значений невозможен;
- **tabindex** – порядковый номер в последовательности перехода по клавише "TAB";

- **disabled** – логический атрибут, наличие которого "отключает" данный элемент формы;
- **onfocus**, **onblur**, **onchange** – внутренние события.

Тег SELECT создает меню, в котором каждый вариант выбора задается тегом OPTION, используемом в содержимом тега SELECT (также как тег LI является элементом списка). Порядок следования тегов OPTION определяет порядок, в котором будут отображены пункты списка выбора.

13.4. Как использовать тег OPTION?

Присутствие начального тега OPTION является обязательным, конечный тег не обязателен и обычно не указывается.

Список атрибутов:

- **value** - указывает значение, которое пересылается серверу, если из списка предлагаемых значений выбран содержащий его элемент. В том случае, если атрибут **value** не задан, при отправке формы, в качестве результата выбора серверу будет отсылаться содержимое тега OPTION. Например, при выборе любого их обоих нижеуказанных элементов, серверу будет отослано значение *Red*:

```
<SELECT>
<OPTION value = "red"> зеленый цвет
<OPTION>Red
</SELECT>
```

- **selected** - логический атрибут, наличие которого делает содержащий его тег OPTION выбранным по умолчанию;
- **disabled** – логический атрибут, наличие которого отключает данный элемент;
- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);

- **style** – встроенная информация о стиле;
- **onfocus**, **onblur**, **onchange**, **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример простейшего задания тега SELECT </title>
      <meta charset="UTF-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h1> Пример простейшего задания тега SELECT</h1>
   <form>
      <select>
      <option value="VR"> Виртуальная и дополненная реальность
</option>
      <option value="IoT"> Интернет-вещей </option>
      <option value="BPLA"> Беспилотные летательные аппараты </
option>
```

Рис. 13.2. Пример простейшего задания тега SELECT: первоначальный вид после загрузки

```
      <option value="Space"> Космические технологии </option>
      <option value="Robots"> Робототехника </option>
      <option value="ASU"> Автоматизированные системы
управления </option>
      <option value="Design"> Промышленный дизайн </option>
      <option value="Finance"> Финансовые технологии </option>
      <option value="Neuro"> Нейротехнологии </option>
      </select>
   </form>
   </body>
</html>
```

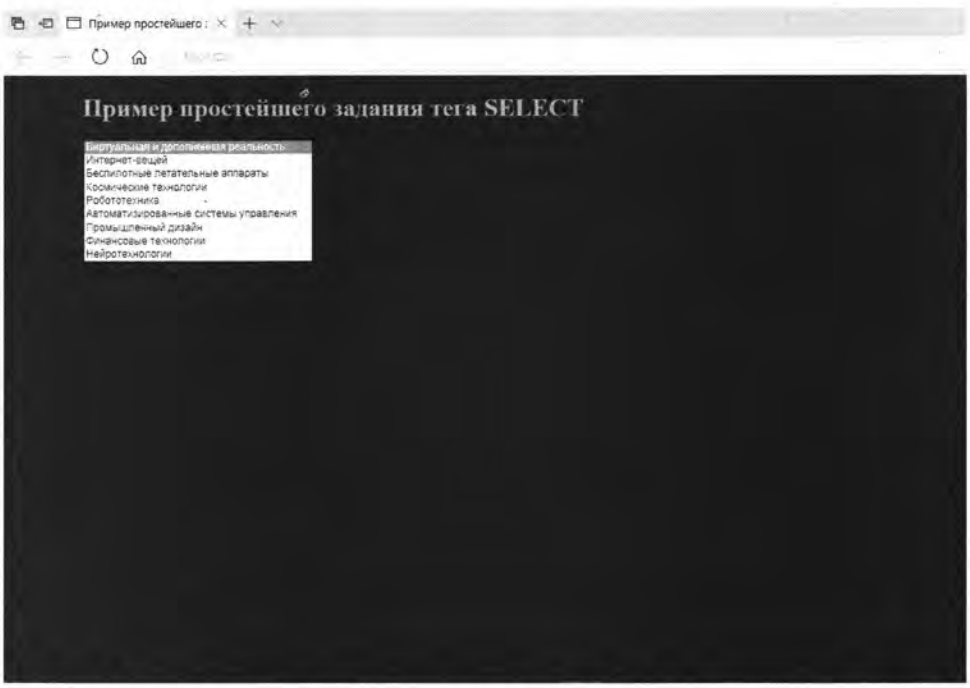


Рис. 13.3. Пример простейшего задания тега SELECT: вид, при нажатии на стрелочку в теге SELECT (раскрытии списка)

Если немного изменить HTML-код документа, как это показано ниже, то можно получить следующий результат:

Листинг документа, отображенного на рис.13.4:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример простейшего задания тега SELECT </title>
```

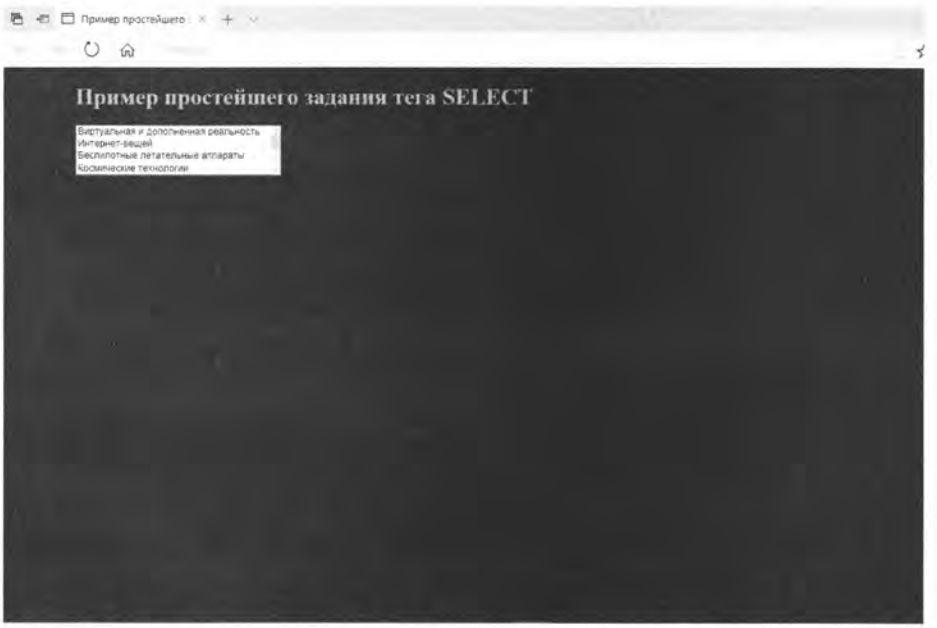


Рис. 13.4. Пример задания элемента SELECT с возможностью многовариантного выбора и со значением, установленным по умолчанию

```
    <meta charset="UTF-8">
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
    <h1> Пример простейшего задания тега SELECT </h1>
  <form>
    <select multiple>
    <option value="VR"> Виртуальная и дополненная реальность
</option>
    <option value="IoT"> Интернет-вещей </option>
    <option value="BPLA"> Беспилотные летательные аппараты </
option>
    <option value="Space"> Космические технологии </option>
    <option value="Robots" selected> Робототехника </option>
    <option value="ASU"> Автоматизированные системы
управления </option>
    <option value="Design"> Промышленный дизайн </option>
    <option value="Finance"> Финансовые технологии </option>
    <option value="Neuro"> Нейротехнологии </option>
    </select>
  </form>
  </body>
</html>
```

Как видно из рисунка 13.4, по умолчанию, при многовариантном режиме выбора, единовременно отображаются четыре элемента списка выбора. Чтобы изменить этот показатель используйте атрибут **size** тега SELECT.

Выше было сказано, что при многовариантном выборе за раз могут быть выбраны несколько только рядом расположенных элементов выбора. Однако, это не совсем так. Выбор размещенных в разных местах списка элементов может осуществляться по умолчанию, с помощью многократного использования атрибута **selected** элемента OPTION.

Например:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример простейшего задания тега SELECT </title>
      <meta charset="UTF-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h1> Пример простейшего задания тега SELECT</h1>
   <form>
```

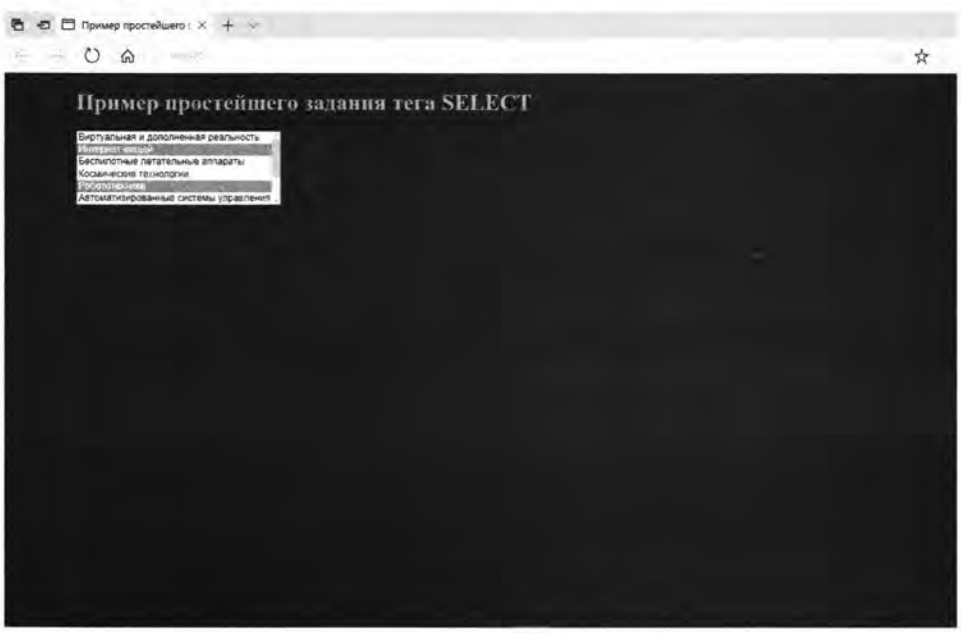


Рис. 13.5. Пример использования элемента SELECT для осуществления выбора нескольких, размещенных в разных местах списка элементов

```
    <select multiple size=6>
    <option value="VR"> Виртуальная и дополненная реальность
</option>
    <option value="IoT" selected> Интернет-вещей </option>
    <option value="BPLA"> Беспилотные летательные аппараты </
option>
    <option value="Space"> Космические технологии </option>
    <option value="Robots" selected> Робототехника </option>
    <option value="ASU"> Автоматизированные системы
управления </option>
    <option value="Design"> Промышленный дизайн </option>
    <option value="Finance"> Финансовые технологии </option>
    <option value="Neuro" selected> Нейротехнологии </option>
    </select>
  </form>
  </body>
</html>>
```

13.5. Как использовать тег TEXTAREA?

Тег TEXTAREA создает управляющий элемент для многострочного ввода текста. Визуально он задает внутри формы прямоугольное поле ввода с горизонтальной и вертикальной полосами прокрутки.

Начальный и конечный теги TEXTAREA являются обязательными.

Атрибуты тега TEXTAREA:

- **name** - присваивает имя элементу;
- **rows** - задает число видимых текстовых строк. В качестве значений принимает натуральные числа;
- **cols** - задает видимую ширину в виде количества символов средней ширины;
- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **readonly** – логический атрибут, наличие которого не позволяют пользователю изменять поле;

- **tabindex** – порядковый номер в последовательности перехода по клавише "TAB";
- **disabled** – логический атрибут, наличие которого "отключает" данный элемент формы;
- **onfocus**, **onblur**, **onchange**, **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Примечание. *В том случае, если вводимый текст не помещается в поле ввода указанных размеров, автоматически начинает использовать полосы прокрутки.*

По умолчанию тег TEXTAREA формирует поле для ввода, состоящее из четырех строк по 40 символов в каждой. Поэтому рекомендуется все-таки явно указывать размеры поля ввода через задание атрибутов **cols** и **rows**.

Текстом, используемым по умолчанию, является текст, заданный между начальным и конечным тегами TEXTAREA.

Пример задания:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Команда "Победители во всём!" </title>
      <base href= "http://techrussia.org">
      <meta charset="UTF-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <form>
      <h1> Информация о команде "Победители во всём!" </h1>
      <h2> Иван Смирнов <br> Капитан команды </h2>
   <textarea cols=50 rows=7> Выпускник по специальности
"Инноватика". Обладает 5-летним практическим опытом
руководства техническими проектами.
   </textarea>
      <h2> Пётр Демидов </h2>
   <textarea cols=50 rows=7> Окончил МГТУ им. Баумана
по специальности "Робототехника". Имеет огромный опыт
конструирования роботов.</textarea>
      <h2> Елизавета Казакова </h2>
```

```
   <textarea cols=50 rows=7> Окончила Санкт-Петербургский
политехнический университет по направлению "Графический
дизайн". После знакомства с Петром стала активно
интересоваться промышленным дизайном. </textarea>
      <h2> Алексей Сидоров </h2>
   <textarea cols=50 rows=7> Отвечает за программирование
роботов. </textarea>
   </form>
   </body>
</html>
```

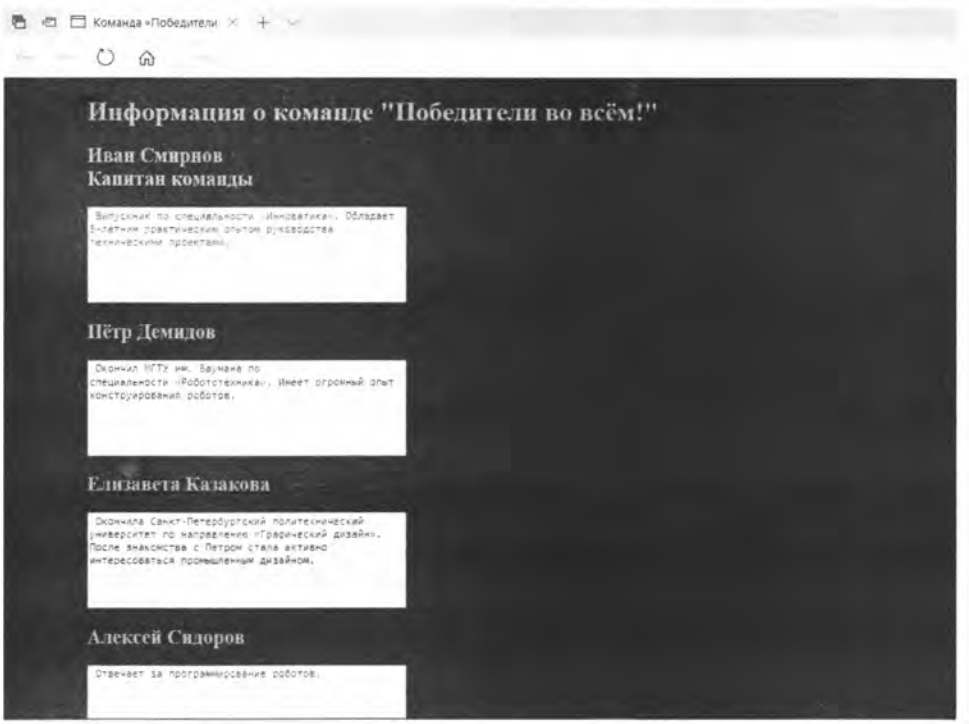


Рис. 13.6. Пример документа, в котором используется тег TEXTAREA

Очень красочно смотрится, если для поля ввода задать свое фоновое изображение. Средствами языка HTML это сделать нельзя, но с помощью использования каскадных таблиц стиля CSS это становится возможным. Сами таблицы стилей будут рассмотрены в следующих уроках. Здесь же приведу только один пример их использования. Для того, чтобы задать фоновое изображение для поля ввода, используйте следующую запись:

```
<TEXTAREA cols=…
```

```
raws=…
style="background-image:URL(… здесь указывается URL-адрес
изображения…">
……..текст……..
……..текст……..
……..текст……..
</TEXTAREA>
```

Кстати говоря, текст, заключенный в содержимом элемента TEXTAREA, воспринимается браузерами как отформатированный, то есть отображается точно в таком виде, в котором он задан.

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Команда "Победители во всём!" </title>
      <base href= "http://techrussia.org">
      <meta charset="UTF-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <form>
      <h1> Информация о команде "Победители во всём!" </h1>
      <h2> Иван Смирнов <br> Капитан команды </h2>
   <textarea cols=50 rows=7> Выпускник по специальности
"Инноватика".
   Обладает 5-летним практическим опытом руководства
техническими проектами.
   </textarea>
      <h2> Пётр Демидов </h2>
   <textarea cols=50 rows=7> Окончил МГТУ им. Баумана по
специальности "Робототехника".
   Имеет огромный опыт конструирования роботов. </textarea>
      <h2> Елизавета Казакова </h2>
   <textarea cols=50 rows=7> Окончила Санкт-Петербургский
политехнический университет по направлению "Графический
дизайн".
   После знакомства с Петром стала активно интересоваться
промышленным дизайном.</textarea>
      <h2> Алексей Сидоров </h2>
   <textarea cols=50 rows=7> Отвечает за программирование
роботов. </textarea>
   </form>
   </body>
</html>
```

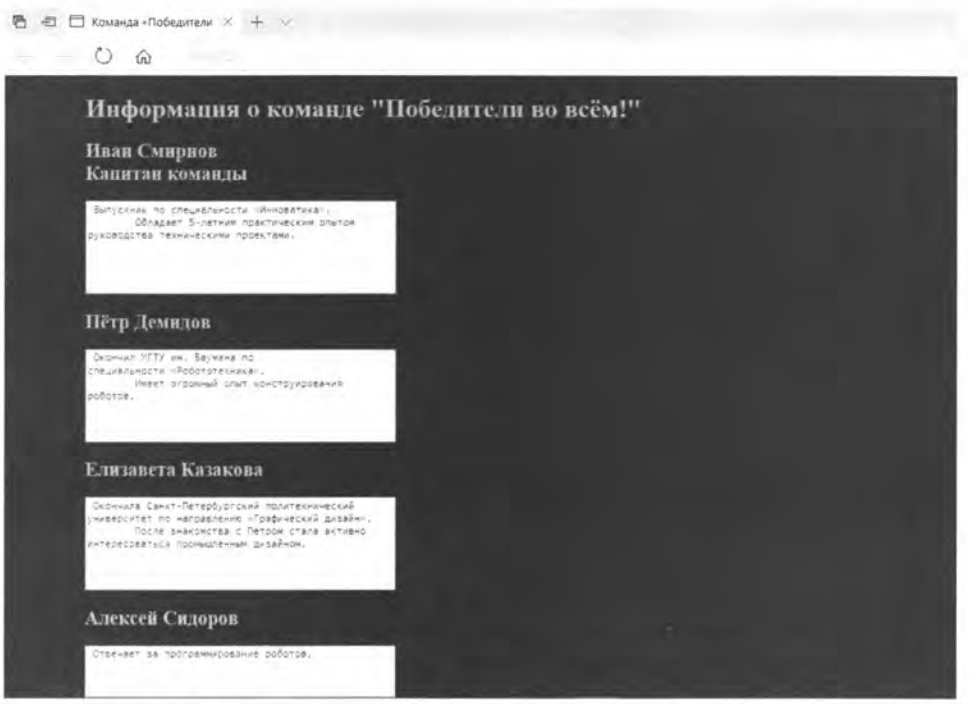


Рис. 13.7. Пример документа, с форматированием тега TEXTAREA

С помощью тега TEXTAREA можно сделать довольно оригинальное дизайнерское решение страницы.

13.6. Как использовать тег INPUT?

Тег INPUT является наиболее употребительным элементом, т.к. с помощью него реализуются основные функции формы. Он позволяет создавать внутри формы поля ввода строки текста, имени файла, пароля др.

Начальный тег при задании элемента INPUT обязателен, конечный тег запрещен.

Список атрибутов элемента INPUT.

- **name** – значением этого атрибута является имя управляющего элемента, образованного данным элементом INPUT. Указание атрибута **name** яв-

ляется обязательным, так как без него невозможно будет переслать значение управляющего элемента;

- **type** - обязательный атрибут, определяющий тип управляющего элемента. В HTML возможны следующие типы управляющих элементов, задаваемых элементом Input с соответствующим значением атрибута **type**:
  - *text* - создает элемент для ввода строки;
  - *password* - аналогичен значению "text" с той разницей, что вводимый текст будет отображаться в виде ряда звездочек.

Примечание: *Использование этого механизма обеспечивает слабую защиту введенной информации. Защитившись от случайных наблюдателей, информация передается на сервер в виде открытого текста, что делает ее доступной для любого пользователя сети на низком уровне*;

  - *checkbox* - создает флажок "вкл\выкл" (on\off). По умолчанию установлен в положении "выкл.". Допускается использование группы из нескольких флажков с одинаковым именем, то есть с одинаковым значением атрибута **name**;
  - *radio* - создает радиокнопку (кнопку с зависимой фиксацией). Используется в составе группы подобных одноименных элементов, из которых выбран (установлен в положение "вкл.") может быть только один. Браузеры отображают их в виде круглой кнопки;
  - *submit* - создает кнопку отправки. Нажатие на эту кнопку отсылает все содержимое формы на сервер. По умолчанию отображается в виде прямоугольной кнопки с надписью "Отправить";
  - *image* - создает кнопку отправки в виде графического изображения;
  - *reset* – создает кнопку сброса. Нажатие этой кнопки сбрасывает установленные пользователем значения для всех элементов формы. Они устанавливаются в значении, используемом по умолчанию. Кнопка сброса по умолчанию изображается с надписью *reset* (Сброс), которую можно изменить при помощи атрибута **value** элемента Input. Ввиду того, что значение кнопок сброса никогда не пересылается на сервер, для них не задается атрибут **name**;
  - *file* - создает управляющий элемент выбора файла. Сам выбор файла осуществляется либо путем ввода с клавиатуры имени файла, либо из диалогового окна, открываемого кнопкой "Browse", которая всегда располагается рядом с полем ввода имени файла. Выбранный таким образом файл присоединяется к содержимому формы и пересылается вместе с ней на сервер. Для передачи присоединенных файлов необ-

ходимо при задании формы указать атрибуты Enctype = "multipart/form-data" и Method = post. В противном случае серверу будет пересылаться не файл, а только его имя. Применительно к элементам выбора файла могут быть применены атрибуты Maxlength и Size;

- *hidden* - создает скрытый элемент, не отображаемый браузером. Содержащаяся в поле неотображаемого управляющего элемента информация пересылается серверу при отправке формы и не может быть изменена ни пользователем, ни браузером;

- **size** – задает начальную ширину управляющего элемента. Значения указывается в пикселах, за исключением управляющих элементов типа **text** и **password**. Для них атрибут **size** задает в количество символов в строке;
- **align** – выравнивание;
- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **alt** – в качестве значения этого атрибута указывается краткая поясняющая информация для данного управляющего элемента;
- **usemap** – используется для подключения клиентской навигационной карты. В качестве значения указывается имя этой карты;

- **readonly** – логический атрибут, наличие которого не позволяет пользователю изменять поле;
- **tabindex** – порядковый номер в последовательности перехода по клавише "TAB";
- **disabled** – логический атрибут, наличие которого "отключает" данный элемент формы;
- **accesskey** – "горячие клавиши" доступа к данному управляющему элементу;
- **onfocus**, **onblur**, **onselect**, **onchange**, **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – внутренние события.

Атрибуты тега INPUT, используемые для управляющего элемента типа **text** (type=text):

- *maxlenght = n* – задает максимальное количество символов в строке, допустимое в поле ввода. По умолчанию число символов не ограничено;
- *size = n* –указывает браузеру количество отображаемых символов в строке ввода (ширину поля ввода в количестве символов);
- *value = text* – задает начальное значение текстового поля, используемое по умолчанию.

Пример задания:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML//EN">
<HTML>
   <HEAD>
      <TITLE> Пример использования элемента INPUT типа text </
TITLE>
   </HEAD>
   <BODY leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Пример простейшей анкеты </H2>
      <FORM>
      Введите Ваше имя <INPUT type=text name="name"><BR>
      <HR>
      <I> Введите Вашу фамилию </I>
      <INPUT type=text name="surname" size=15><BR>
      <HR>
      Введите Ваш адрес электронной почты
      <INPUT type=text name="Email" size=20><BR>
      <HR>
      <B> Введите номер Вашего телефона </B>
      <INPUT type=text name="Phone" size=10 maxlength=13><BR>
      <HR>
      Введите телефонный код города, в котором Вы живете
      <INPUT type=text name="city_code"><BR>
      </FORM>
   </BODY>
</HTML>
```

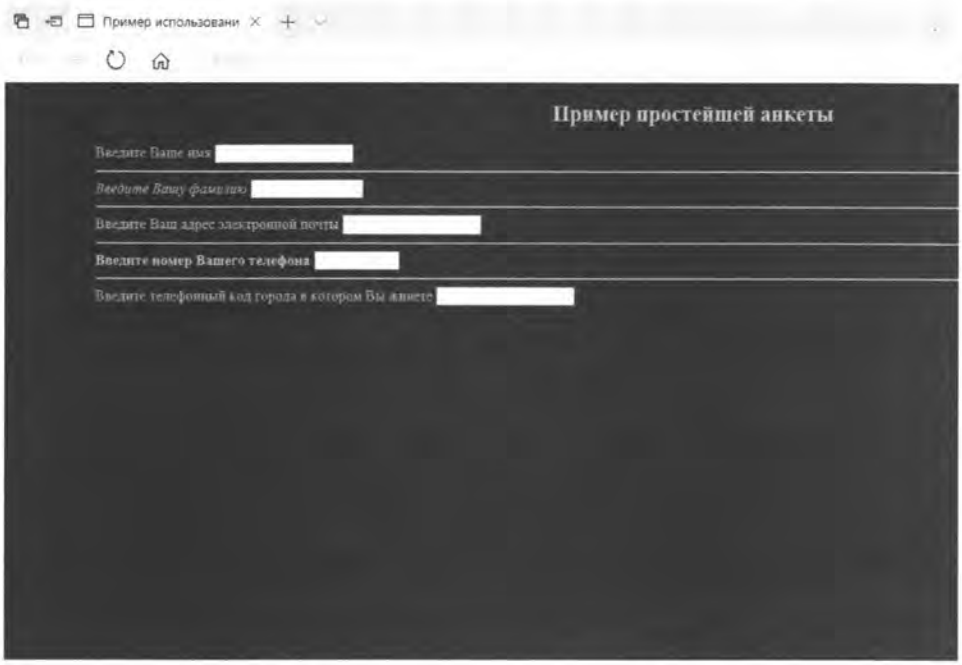


Рис. 13.8. Пример использования тега INPUT типа text

Как видно из примера, содержимое формы можно форматировать как обычный текст. Нельзя форматировать текст, вводимый в поля формы. А все остальное - пожалуйста.

Кроме этого рекомендуется применять наиболее широко используемые и адекватные имена управляющих элементов. Например, если используется управляющий элемент ввода адреса электронной почты, то очень желательно присвоить ему имя “email” (name=“email"), а не какие-нибудь “pochta", “electronn_pochta" и т.п. Браузеры кэшируют тексты, вводимые в поля форм под их именами, что позволяет в дальнейшем просто выбирать значение из ранее вводившихся. Для этого надо просто при вводе нажать клавишу “вниз" или два раза щелкнуть мышкой на требуемом поле. При этом под простым текстовым полем ввода (тег INPUT типа text) возникнет список ранее вводившихся в одноименные поля значений.

Тегу INPUT, задающим управляющий элемент для ввода пароля (type=password), могут быть указаны такие же атрибуты что тегу INPUT типа **text**. А именно: **size**, **maxlength**, **value**.

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
   <TITLE> Пример использования элемента INPUT типа text </
TITLE>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Пример аутентификации </H2>
      <HR size=20 color=lightgrey>
      <form action="www.hthy.net/cgi/autocgi.pl">
      <H3> Введите параметры доступа </H3>
      Введите имя пользователя
      <input type=text value="Николай"><BR><BR>
      Введите пароль
      <input type=password size=8 maxlength=8><BR>
      <I> Примечание: длина пароля не должна превышать восемь
символов </I>
      <HR size=20 color=lightgrey>
      </form>
   </body>
</html>
```

Рис. 13.9. Пример использования тега INPUT типа password

Дополнительные атрибуты тега INPUT, используемые для управляющего элемента типа **checkbox** (type=checkbox):

- **value = text** - обязательный атрибут для флажков. Содержит текстовую строку, которая будет послана серверу, если при заполнении формы данный флажок будет установлен в положение "вкл.". Если флажок в заполненной форме будет установлен в положение "выкл.", то при пересылке формы серверу текстовая строка **value** пересылаться не будет. Если в форме присутствует несколько флажков с одинаковым именем, то пересылаемым значением будет строка разделенных запятыми значений атрибутов **value** одноимённых флажков, установленных в положение "вкл.".
- **checked** - необязательный логический атрибут, задание которого устанавливает флажок по умолчанию в положение "вкл.". Если он не указан, то флажок по умолчанию установлен в положение "выкл."

Пример использования флажков.

```
<!DOCTYPE html>
<html>
   <head>
   <title> Пример использования элемента INPUT типа checkbox </
title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H3 align=center>
      Укажите, произведения каких русских писателей Вы
читали:</H3>
      <form>
      <input type=checkbox name="russian_writer"
value="Pushkin" checked> А.С.Пушкин <br>
      <input type=checkbox name="russian_writer"
value="Lermontov" checked > М.Ю.Лермонтов <br>
      <input type=checkbox name="russian_writer"
value="Tolstoj"> Л.Н.Толстой <br>
      <input type=checkbox name="russian_writer" value="Gogol">
Н.В. Гоголь <br>
      <input type=checkbox name="russian_writer"
value="Nekrasov"> Н.А Некрасов <br>
      <input type=checkbox name="russian_writer" value="Kuprin"
> А.И.Куприн <br>
```

```
     <input type=checkbox name="russian_writer"
value="Pasternak"> Б.Л.Пастернак <br>
     <h3 align=center>
     Укажите, произведения каких зарубежных писателей Вы
читали:</H3>
     <input type=checkbox name="foreign_writer" value="Tomas_
Mann"> Томас Манн <br>
     <input type=checkbox name="foreign_writer"
value="Dickens"> Чарльз Диккенс <br>
     <input type=checkbox name="foreign_writer" value="Show">
Бернард Шоу <br>
     <input type=checkbox name="foreign_writer"
value="Kipling" checked > Редьярд Киплинг <br>
     <input type=checkbox name="foreign_writer" value="Bell">
Генрих Белль <br>
     <input type=checkbox name="foreign_writer"
value="Ficgerald"> Скотт Фицджеральд <br>
     <input type=checkbox name="foreign_writer"
value="Selinger"> Джером Селинджер <br>
     <input type=checkbox name="foreign_writer" value="Twen"
checked > Марк Твен
     </form>
   </body>
</html>
```

Рис. 13.10. Пример использования флажков (тег INPUT типа checkbox)

Дополнительные атрибуты тега INPUT, используемые для радиокнопок (type=radio):

- **value = text** – обязательный атрибут, значение которого передается серверу, если при заполнении формы данная радиокнопка будет установлена в положение "вкл.". Каждая радиокнопка должна иметь уникальное имя (атрибут name) в пределах формы.
- **checked** - необязательный логический атрибут, задание которого устанавливает радиокнопку по умолчанию в положение "вкл.". Если он не указан, то радиокнопка по умолчанию установлена в положение "выкл."

Если флажки позволяют осуществлять "включение" сразу нескольких одноименных управляющих элементов, то радиокнопки наоборот, исключают эту возможность.

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
   <title> Пример использования элемента INPUT типа radio </
title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H3 align=center>
      Выберите из списка своего наиболее любимого русского
писателя </H3>
      <form>
      <input type=radio name="russian_writer" value="Pushkin"
checked> А.С.Пушкин <br>
      <input type=radio name="russian_writer" value="Lermontov"
checked > М.Ю.Лермонтов <br>
      <input type=radio name="russian_writer" value="Tolstoj">
Л.Н.Толстой <br>
      <input type=radio name="russian_writer" value="Gogol">
Н.В. Гоголь <br>
      <input type=radio name="russian_writer" value="Nekrasov">
Н.А.Некрасов <br>
      <input type=radio name="russian_writer" value="Kuprin" >
А.И.Куприн <br>
      <input type=radio name="russian_writer"
value="Pasternak"> Б.Л.Пастернак <br>
      <h3 align=center >
```

```
    Выберите из списка своего наиболее любимого зарубежного
писателя</H3>
    <input type=radio name="foreign_writer" value="Tomas_
Mann"> Томас Манн <br>
    <input type=radio name="foreign_writer" value="Dickens">
Чарльз Диккенс <br>
    <input type=radio name="foreign_writer" value="Show">
Бернард Шоу <br>
    <input type=radio name="foreign_writer" value="Kipling" >
Редьярд Киплинг <br>
    <input type=radio name="foreign_writer" value="Bell">
Генрих Белль <br>
    <input type=radio name="foreign_writer"
value="Ficgerald"> Скотт Фицджеральд <br>
    <input type=radio name="foreign_writer" value="Selinger">
Джером Селинджер <br>
    <input type=radio name="foreign_writer" value="Twen"
>МаркТвен
    </form>
  </body>
</html>
```

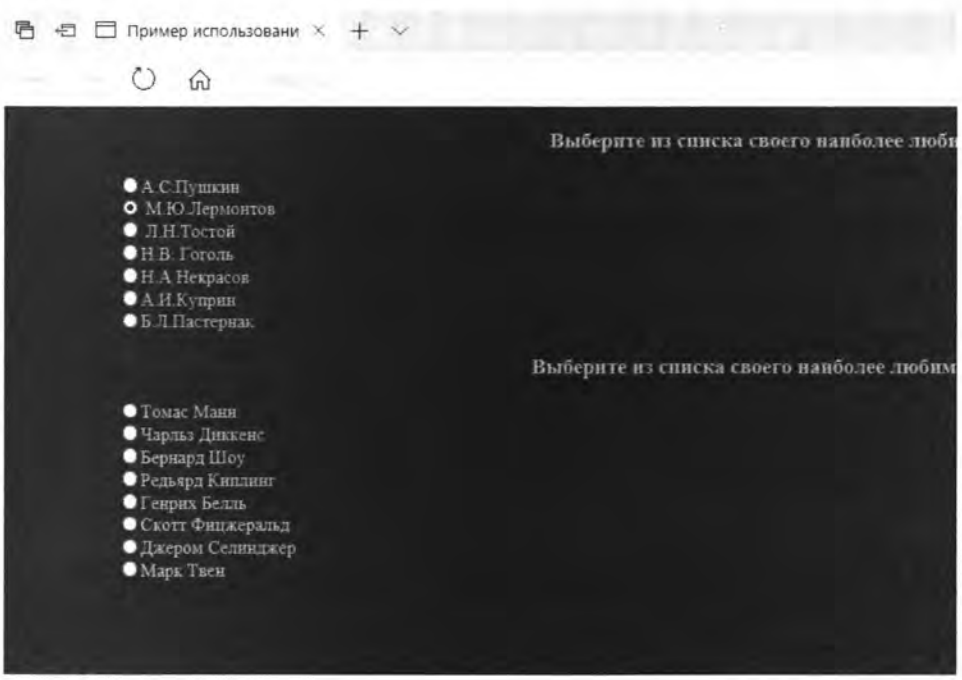


Рис. 13.11. Пример использования радиокнопок.

Таким образом, в данном случае может быть выбран только один русский писатель и только один иностранный. Кстати, что касается иностранных писателей, то среди них может быть не выбран ни один писатель, так как ни один из них не установлен выбранным по умолчанию. И если теперь вообще не трогать иностранных писателей, то при отправке ни одно из их имен не будет послано.

Атрибуты тега Input, используемые для кнопок отправки (type=submit):

- **value = text** - указывает пользовательское название кнопки отправки. Данный атрибут позволяет изменить надпись на кнопке. Имя кнопок отправки (атрибут **name**) можно не указывать. В этом случае, значение кнопки отправки не передается на сервер. Если для нее заданы оба атрибута **name** и **value**, то в качестве отправляемого значения будет использоваться название кнопки.

Чтобы отправить форму, необходимо в рамках нее задать кнопку отправки. То есть, чтобы все формы, заданные в вышеуказанных примерах были отосланы на сервер и там обработаны, необходимо задать для них кнопку отправки. Иначе они являются просто предметом интерфейса, и тег FORM, в таком случае, во всех вышеприведенных примерах можно не использовать.

Итак, возьмем один из примеров и вставим в него кнопку отправки.

```
<!DOCTYPE html>
<html>
   <head>
   <TITLE> Пример использования элемента INPUT типа text </
TITLE>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Пример аутентификации </H2>
      <HR size=20 color=lightgrey>
      <form action="www.hthy.net/cgi/autocgi.pl">
      <H3> Введите параметры доступа </H3>
      Введите имя пользователя
      <input type=text value="Николай"><BR><BR>
      Введите пароль
      <input type=password size=8 maxlength=8><BR>
      <I> Примечание: длина пароля не должна превышать восемь
символов </I>
      <HR size=20 color=lightgrey>
```

```
      <input type=submit><br>
      <input type=submit value="Отправить"><br>
      </form>
   </body>
</html>
```

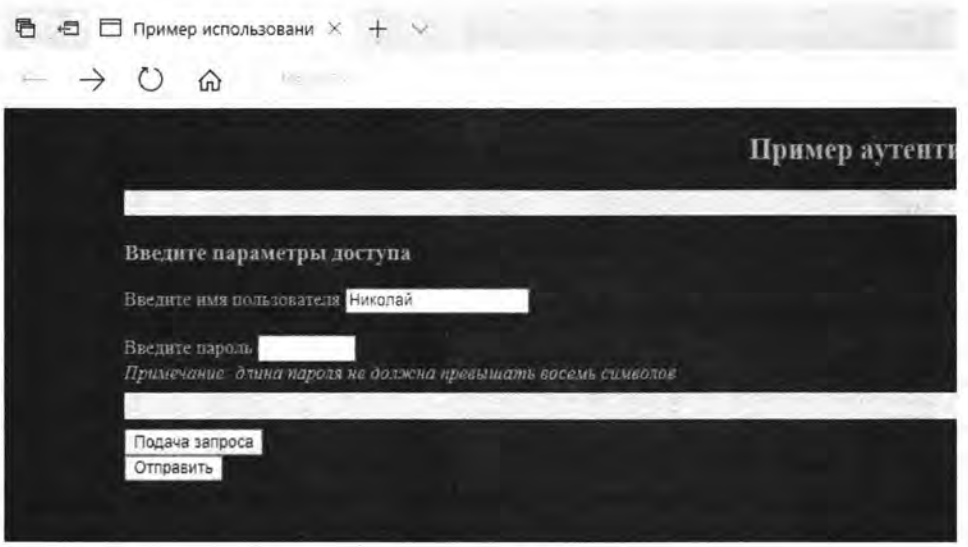


Рис. 13.12. Пример задания формы с кнопками передачи.

В рамках одной формы допустимо задание нескольких кнопок передачи, что и продемонстрировано в вышеприведенном примере. Обе кнопки являются абсолютно равноправными и приводят к передаче содержимого формы сценарию, расположенному по адресу www.hthy.net/cgi/autocgi.pl.

Дополнительные атрибуты для тега INPUT, используемые для кнопок отправки в виде графического изображения:

- src - указывает URL-адрес изображения
- align - задает тип выравнивания изображения

В качестве значения такой кнопки будут отосланы координаты указателя мыши при ее нажатии:

- имя_ кнопки x = значение координаты x
- имя _кнопки y = значение координаты y

Отправка координат, как и в случае с простой кнопкой передачи, будет производиться только если задано имя управляющего элемента.

Из соображений доступности разработчикам рекомендуется указывать альтернативное текстовое содержимое графической кнопки через атрибут **alt**.

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

<!DOCTYPE html>
<HTML>
   <HEAD>
      <TITLE> Пример использования элемента INPUT типа text </
TITLE>
   </HEAD>
   <BODY leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Пример простейшей анкеты </H2>
      <FORM>
      Введите Ваше имя <INPUT type=text name="name"><BR>
      <HR>
      <I> Введите Вашу фамилию </I>
      <INPUT type=text name="surname" size=15><BR>
      <HR>
      Введите Ваш адрес электронной почты
      <INPUT type=text name="Email" size=20><BR>
```

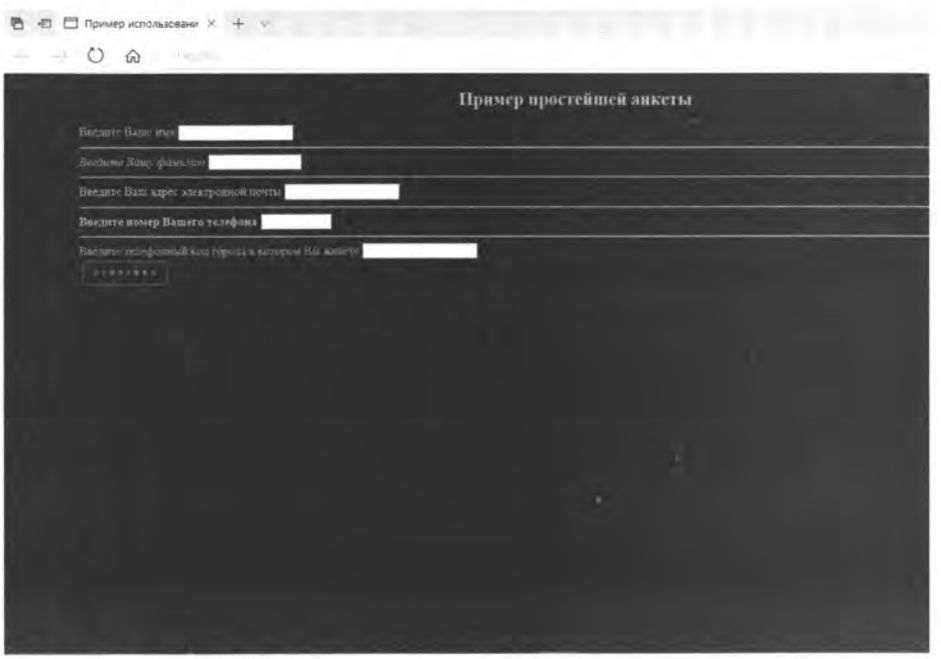


Рис. 13.13. Пример формы с графической кнопкой отправки

```
      <HR>
      <B> Введите номер Вашего телефона </B>
      <INPUT type=text name="Phone" size=10 maxlength=13><BR>
      <HR>
      Введите телефонный код города, в котором Вы живете
      <INPUT type=text name="city_code"><BR>
      <input type=image src="submit.jpg" alt="Отправить форму">
      </FORM>
   </BODY>
</HTML>
```

Аналогично кнопки отправки создается кнопка сброса (type=reset). При нажатии на эту кнопку, все управляющие элементы содержащей его формы устанавливаются в своем первоначальном значении (значении по умолчанию)

Пример задания:

```
<!DOCTYPE html>
<html>
   <head>
   <title>Пример задания кнопок сброса</title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
```

Рис. 13.14. Пример задания кнопок сброса

```
     <h2 align=center> Пример задания кнопок сброса </h2>
     <hr size=20 color=lightgrey>
     <input type=reset> - кнопка сброса со стандартной
надписью
     <br><br>
     <input type=reset value="Очистить форму">
     - кнопка сброса с измененной надписью
     <hr size=20 color=lightgrey>
   </body>
</html>
```

Теперь рассмотрим на примере использование управляющего выбора файла (type=file):

```
<!DOCTYPE html>
<html>
   <head>
   <title> Пример управляющего элемента выбора файла </title>
   </head>
     <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
     <H2 align=center> Выберите файл </H2>
     <form  action="www.ewyye.spb.ru/cgi-bin/fyu.pl"
```

Рис. 13.15. Пример использования управляющего элемента выбора файла.

```
    Enctype = "multipart/form-data" Method = post>
    <HR size=20 color=lightgrey>
    <input type=file name="app_file">
    <HR size=20 color=lightgrey>
    <input type=submit>
    <input type=reset>
  </body>
</html>
```

Как и во всех предыдущих случаях, имя файла, используемое по умолчанию, можно задать через атрибут **value**.

13.7. Как использовать тег Button?

Тег BUTTON создает кнопки, аналогично кнопкам сброса и отправки, определяемые тегом INPUT. Отличие кнопок, задаваемых тегом BUTTON в том, что они могут использоваться для различных целей. Тег BUTTON обычно применяется вместе с атрибутом **onclick**, в значении которого указывается адрес клиентского сценария, который будет активизироваться при нажатии на кнопку.

Преимуществом этих кнопок также является более богатые возможности их визуального представления. Например, кнопка, задаваемая тегом BUTTON, может иметь вид графической кнопки, но не может быть при этом кнопкой отправки, т.е. может принимать значения, отличные от координат курсора. Этот элемент используется для создания красочного пользовательского интерфейса.

Для тега BUTTON определены следующие атрибуты:

- **name** – в качестве его значения указывается имя данного управляющего элемента;
- **value** – через этот атрибут задается значение, которое будет передано сценарию после нажатии на кнопку (если при этом задан атрибут **name**);
- **type** – этот атрибут указывает тип данной кнопки BUTTON, при этом возможны следующие варианты:
  - *submit* – создается кнопка отправки, аналогичная по своей функции кнопке отправки INPUT. Это значение используется по умолчанию;
  - *reset* – создается кнопка сброса, аналогичная по своей функции кнопке сброса INPUT;

- *button* – создается уникальная пользовательская кнопка, функциональное предназначение которой должен задать сам разработчик. Именно в сочетании с этим типом кнопок BOTTON используется атрибут **onclick**.

- **disabled** – логический атрибут, указание которого делает данную кнопку отключенной;
- **accesskey** – в качестве значения этого атрибута указываются "горячие клавиши" для быстрого доступа к данной кнопке;
- **usemap** – этот атрибут связывает данную кнопку с навигационной картой. В качестве значения атрибута **usemap** указывается имя навигационной карты, с которой связывается данная кнопка. Другими словами, значение атрибута **usemap** тега BUTTON должно совпадать с атрибутом **name** тега MAP, между которыми устанавливается связь;
- **tabindex** – этот атрибут определяет положение данного управляющего элемента в последовательности перехода по клавише TAB для текущего документа;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – **onsubmit**, **onreset** – внутренние события, для обработки которых указывается имя соответствующего сценария.

Задание начального и конечного тегов BUTTON обязательно. Тег BUTTON является контейнером. Все, что задано в содержимом этого элемента отображается на кнопке, размер которой подгоняется под размер располагаемой на ней информации. Если между начальным и конечным тегами BUTTON ничего не задано, то отображена будет пустая кнопка без надписи небольших размеров.

Пример задания:

```
<!DOCTYPE html>
<html>
   <head>
   <title> Примеры задания элемента BUTTON </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Примеры задания элемента BUTTON </H2>
      <button></button> - Пример пустой кнопки,
```

```
     определяемой тегом BUTTON, в содержимом которого ничего
не задано <BR><BR>
     <button type=submit> Кнопка отправки </button>
     <BR><BR>
     <button type=submit><IMG src="submit.jpg"></button> - еще
одна кнопка отправки
     <BR><BR>
     <button type=buttom onclick="mycgiscript.pl">
     <font     size=+1>
     <I><B> Уникальная кнопка, определенная разработчиком </
B></I>
     </font>
     </button>
     <BR><BR>
     <button type=reset>
     <IMG src="reset.jpg"></button>
     <BR><BR>
     <button>
     <IMG src="pol.jpg">
     На поверхности кнопки можно выводить все что угодно
     <IMG src="pol.jpg">
     </button>
   </body>
</html>
```

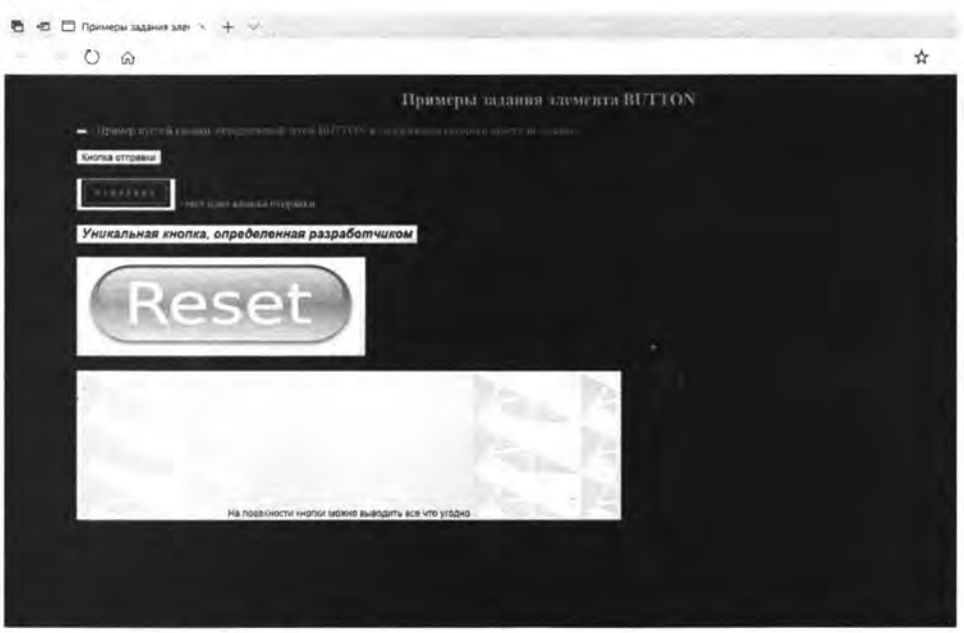


Рис. 13.16. Примеры кнопок, задаваемых тегом BUTTON

Заслуживает внимания тот факт, что задание графической кнопки сброса (reset) с помощью тега INPUT не предусмотрено. Это упущение может быть восполнено с помощью тега BUTTON, что и продемонстрировано в вышеприведенном примере: одна из кнопок является графической кнопкой сброса.

13.8. Как использовать тег LABEL?

При отображении флажков или радиокнопок пользователю представляется круглая или квадратная кнопка без каких-либо пояснений. Можно рядом с кнопкой вывести поясняющий текст, но он и радиокнопка (флажок) воспринимаются браузером как абсолютно обособленные, ничем не связанные элементы. Использование меток позволяет связывать с управляющими элементами дополнительную информацию. И события, совершающиеся при взаимодействии пользователя с метками, интерпретируются браузером как события, связанные с управляющими элементами, с которыми связаны данные метки.

В случае с флажками, если поясняющий текст заключен в содержимом метки флажка, то нажатие мышки над этим текстом эквивалентно нажатию мышки над самой кнопкой флажка и приведет к его переключению. Если, например, метка связана с кнопкой отправки формы, то щелчок над отображенным содержимым метки приведет к отправке формы на сервер.

В HTML5 метки реализуются тегом LABEL. Указание начального и конечного тегов является обязательным.

Список атрибутов тега LABEL:

- **for** - обязательный атрибут, связывающий метку с управляющим элементом. В качестве значения этого атрибута указывается заключенное в кавычки **id**-имя управляющего элемента, находящегося с меткой в пределах одного документа;

  Примечание. *В данном случае у управляющего элемента должно быть задано два имени: атрибут* name, *с которым связывается значение управляющего элемента, и атрибутом* id, *служащем для обращения к управляющему элементу.*

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;

- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;
- **accesskey** – в качестве значения этого атрибута указываются "горячие клавиши" для быстрого доступа к данной кнопке;
- **tabindex** – этот атрибут определяет положение данного управляющего элемента в последовательности перехода по клавише TAB для текущего документа;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – **onsubmit**, **onreset** – внутренние события, для обработки которых указывается имя соответствующего сценария.

С одним управляющим элементом может быть связано несколько меток. Причем они могут быть расположены по тексту как до, так и после управляющего элемента.

Пример задания:

```
<!DOCTYPE html>
<html>
   <head>
   <title> Примеры использования элемента Label </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Примеры использования меток </H2>
      <form action="www.erwsd.com/cgi-bin/grdj.pl">
      <input type=checkbox name="flag1" value="true" id="first
flag">
      <label for="first flag">
      Нажатие на этот текст приведет к переключению
расположенного слева флажка
      </label>
      <BR><BR>
      <input type=checkbox name="flag2" value="true">
      Нажатие на этот текст  ни к чему не приведет
      <BR><BR>
      <input type="text" name="name">
      <BR><BR>
      <label for="my reset">
```

```
      <B> Сбросить введенную информацию </B>
      </label>
      <BR><BR>
      <input type=radio name="radio" value="coca-cola"
      id="first radio" checked>
      <label for= "first radio">
      Установить  расположенную слева радиокнопку в положение "вкл."
      </label>
      <BR><BR>
      <input type=radio name="radio" value="sprite" id="second
radio">
      <label for= "second radio">
      Установить  расположенную слева радиокнопку в положение "вкл."
      </label>
      <BR><BR>
      <input type=radio name="radio" value="fanta" id="third radio">
      <label for="third radio">
      Установить  расположенную слева радиокнопку в положение "вкл."
      </label>
      <BR><BR>
      <input type=reset id="my reset">
```

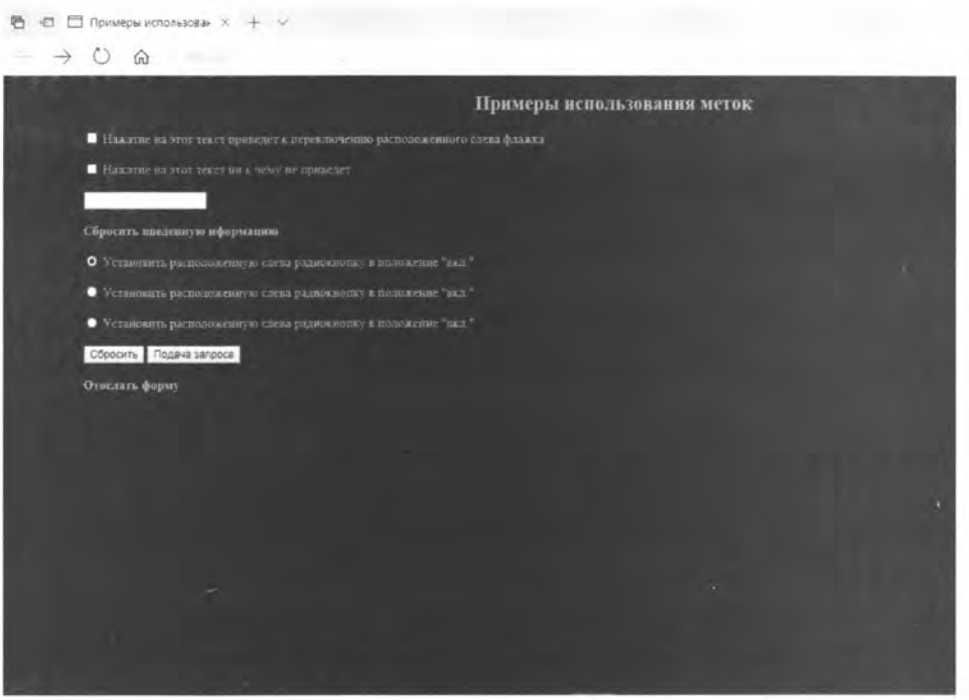


Рис. 13.17. Пример формы с использованием меток

```
      <input type=submit id="my submit">
      <BR><BR>
      <label for="my submit">
      <B></I> Отослать форму </I></B>
      </label>
      </form>
   </body>
</html>
```

13.9. Как использовать теги LEGEND и FIELDSET?

Тег FIELDSET позволяет разработчикам группировать связанные метки и управляющие элементы. На экране монитора его использование приводит к тому, что управляющие элементы и связанные с ними метки, расположенные между начальным и конечным тегами FIELDSET, обводятся рамкой. Этот элемент обычно используется в сочетании с тегом LEGEND.

Тег LEGEND служит для задания заголовка группе выделенных тегом FIELDSET управляющих элементов и их меток.

Указание начального и конечного тегов является обязательным.

Тег LEGEND имеет необязательный атрибут **align**, который позволяет выводить заголовок либо справа, либо слева на верхнем сегменте рамки. Для этого он может принимать следующие значения:

- *left* – заголовок выводится слева на верхнем сегменте рамки. Это значение используется по умолчанию;
- *right* – заголовок выводится справа на верхнем сегменте рамки.

И тег LEGEND, и тег FIELDSET имеют следующие стандартные атрибуты (все необязательные):

- **id**, **class** – идентификаторы в пределах документа;
- **lang**, **dir** – информация о языке и направленности текста;
- **title** – заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);
- **style** – встроенная информация о стиле;

- **accesskey** – в качестве значения этого атрибута указываются "горячие клавиши" для быстрого доступа данной выделенной группе управляющих элементов;
- **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup** – **onsubmit**, **onreset** – внутренние события, для обработки которых указывается имя соответствующего сценария.

Теги FIELDSET могут быть вложенными, однако каждый из них может содержать только один заголовок LEGEND.

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
      <title>
      Пример использования тегов FIELDSET и LEGEND для разметки
формы
      </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Заполните анкету </H3>
      <form action="www.dsdr.da.ru/cgi-bin/hyh.pl">
      <P>
      <fieldset>
      <legend><B><I> Личная информация </I></B></legend>
      Фамилия: <input name="last name" type="text" size=30>
      Имя: <input name="first name" type="text">
      Телефон: <input name="phone" type="text"><BR>
      Адрес: <input name="address" type="text" size=90>
      </fieldset>
      <fieldset>
      <legend align=right> Опыт работы Web-мастером </legend>
      <input name="Experience"
            type="radio"
            value="none"
      checked> Нет опыта
      <input name="Experience"
            type="radio"
            value="<1"> Меньше 1 года
      <input name="Experience"
           type="radio"
```

```
          value="1-3"> от 1 до 3 лет
      <input name="Experience"
            type="radio"
            value=">3"> более 3 лет
      </fieldset>
      <BR>
      <fieldset>
        Есть ли у Вас готовые работы или законченные проекты, в
которых Вы учавствовали?
      <BR>
      <input name="work"
            type="radio"
            value="Yes"> Да
      <input name="work"
            type="radio"
            value="No" checked>Нет
      Если да, то перечислите их ниже:<BR>
      <textarea name="current_medication"
             rows="10" cols="50" tabindex="40">
      </textarea>
      </fieldset>
      <input type=reset>
      <input type=submit>
      </form>
   </body>
</html>
```

Рис. 13.18. Пример использования тегов FIELDSET и LEGEND для структурирования и разметки формы

13.10. Как использовать тег DATALIST?

Использование тега DATALIST позволяет создать список вариантов, который сможет ввести пользователь в поле INPUT.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Пример простейшего задания тега DATALIST</title>
      <meta charset="UTF-8">
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <h1> Пример простейшего задания тега DATALIST </h1>
      <p> Укажите свою роль в команде </p>
      <p><input list="role">
      <datalist id="role">
         <option value="Инженер"></option>
         <option value="Программист"></option>
```

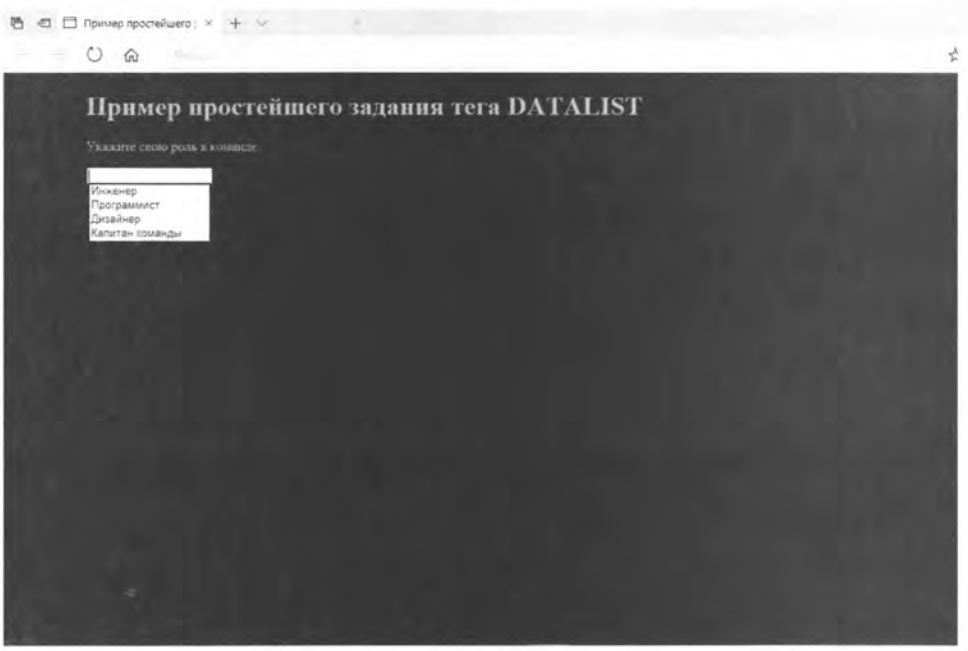


Рис. 13.19. Использование тега DATALIST

```
        <option value="Дизайнер"></option>
        <option value="Капитан команды"></option>
    </body>
</html>
```

13.11. Как управлять фокусом?

В HTML-документах чтобы реализовать связанные с каждым элементом возможности (ввести текст, переключить флажок и т.п.), пользователь должен перевести на отображаемую часть элемента фокус, т.е. сделать элемент активным. Например, чтобы ввести текст в поле тега TEXTAREA, необходимо сначала перевести на него фокус. На тег TEXTAREA фокус переведен тогда, когда в его поле ввода мигает курсор.

Браузеры поддерживают несколько способов передачи фокуса элементам:

- Указание элемента с помощью щелчка мыши над ним;
- Указание элемента посредством последовательного перевода фокуса с одного элемента на другой с помощью клавиши TAB;
- Переход к элементу с помощью набора " горячих " клавиш.

Работа с мышью, а надеюсь, ни у кого никаких трудностей не вызывает, поэтому не будем на этом останавливаться.

Перевод фокуса при помощи клавиши TAB

Практически все управляющие теги в списке своих атрибутов содержат атрибут **tabindex**, который определяет положение элемента в последовательности перехода по клавише "Tab" для текущего документа. В качестве значений атрибут **tabindex** принимает целые числа от 0 до 32767. По умолчанию этот атрибут установлен в значении "0" и переход с одного элемента на другой осуществляется в порядке их следования по тексту HTML- документа. Переход по элементам, поддерживающих атрибут **tabindex** (а к ним относятся теги A, AREA, BUTTON, INPUT, SELECT, TEXTAREA, OBJECT) осуществляется в следующем порядке:

- Если в документе имеются элементы с ненулевым значением атрибута **tabindex**:

- сначала производится последовательный переход фокуса от одного элемента с ненулевым значением **tabindex** к другому в порядке возрастания атрибута **tabindex**. Причем последовательность значений этого атрибута может быть дискретной (не непрерывной), и начинаться с любого числа. Например: 5, 6, 32, 87.
- затем осуществляется последовательный переход фокуса по элементам с нулевым значением атрибута **tabindex** в порядке их следования по тексту.

- Если в документе отсутствуют элементы с ненулевым значением атрибута **tabindex**, то переход по клавише "Tab" осуществляется в порядке задания элементов в тексте HTML-документа.

Множество значений **tabindex** распространяется на весь документ и поэтому допускается последовательный переход фокуса на элементы, вложенные в разные элементы: например, в разные формы. Отключенные управляющие элементы, для которых установлен логический атрибут **disabled**, не принимают участия в последовательности перехода.

Перевод фокуса при помощи набора "горячих" клавиш.

При использовании этого способа передачи элементу фокуса, активизация элемента осуществляется посредством нажатия набора "горячих" клавиш. Для тегов, поддерживающих данный способ фокусировки (A, AREA, BUTTON, INPUT, LABEL, LEGEND и TEXTAREA) горячие клавиши устанавливаются атрибутом **accesskey**.

Задание атрибута выглядит, например, следующим образом:

```
<INPUT type = "reset" accesskey = "U">
```

При этом клавиша сброса может быть активирована нажатием на клавиатуре комбинации Alt-U. Маленькая деталь: в этот момент на компьютере должна использоваться английская клавиатура.

В общем случае действие, происходящее при получении элементом фокуса, зависит от самого элемента. Например, при передаче фокуса элементу **A** автоматически происходит переход по задаваемой им ссылке. В случае получения фокуса элементом-радиокнопкой, происходит ее переключение и т.п.

Пример задания:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
    Пример использования атрибутов TABINDEX,ACCESSKEY и
DISABLED
    </title>
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
    <H3 align=center>
    Пример использования атрибутов TABINDEX,ACCESSKEY и
DISABLED </H3>
    Имя Вашего любимого писателя
    <input type=text name="writer" tabindex=1><BR><BR>
    Его лучшее, на Ваш взгляд, произведение
    <input type=text name="roman" tabindex=10><BR><BR>
    Имя Вашего любимого киноактера
    <input type=text name="writer" tabindex=100><BR><BR>
```

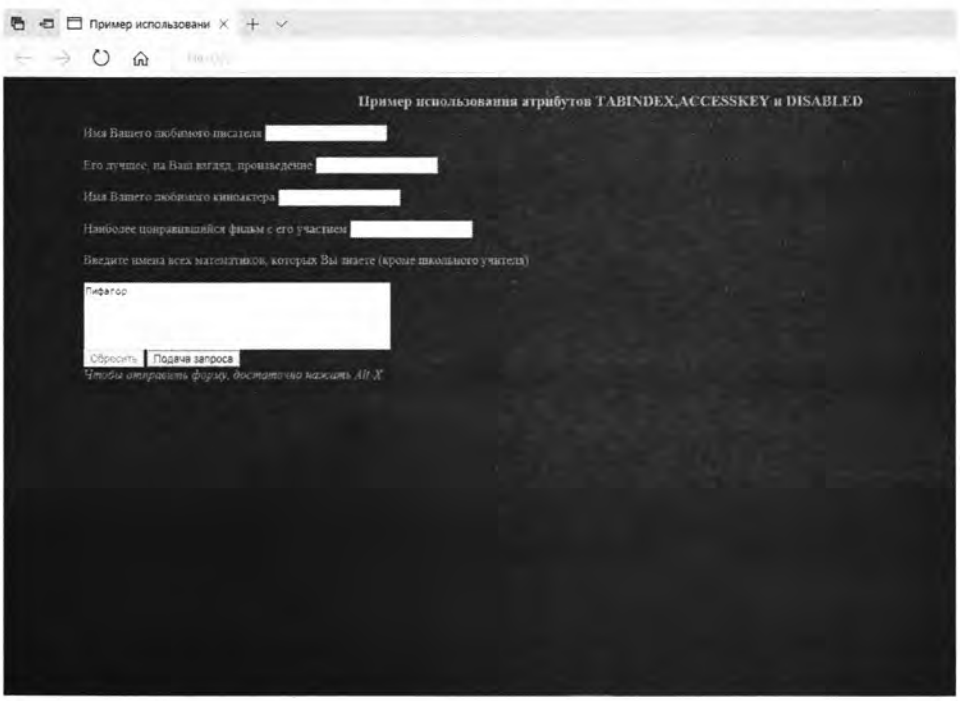


Рис. 13.20. Документ, в котором используются атрибуты tabindex, accesskey и disabled

```
      Наиболее понравившийся фильм с его участием
      <input type=text name="writer" tabindex=1000><BR><BR>
      Введите имена всех математиков, которых Вы знаете (кроме
школьного учителя)
      <BR><BR>
      <textarea rows=5 cols=50 name="math" tabindex=1001>
Пифагор </textarea><BR>
      <input type=reset disabled>
      <input type=submit accesskey="X"><br>
      <I> Чтобы отправить форму, достаточно нажать Alt-X. </I>
   </body>
</html>
```

Таким образом, переход по клавише Tab осуществляется в следующем порядке: сначала курсор помещается в поле ввода имени писателя, затем в поле ввода его произведения, поле ввода имени актера, поле ввода фильма с его участием, а затем фокус передается кнопке отправки. Кнопка сброса вообще отключена, а вместо кнопки отправки можно использовать комбинацию клавиш Alt-X.

Управляющие теги, задаваемые HTML-элементами BUTTON, INPUT, SELECT, OPTION и TEXTAREA могут быть указаны как отключенные. В этом случае доступ к управляющему элементу делается невозможным. Например, можно отключить кнопку отправки формы до тех пор, пока необходимые поля формы не будут заполнены. Снять отключение с управляющих тегов можно только с помощью скриптов.

Отключенные управляющие элементы. Атрибуты disabled и readonly

Отключение элементов осуществляется выставлением у них логического атрибута **disabled**. Если это имеет место, то элемент нельзя активизировать (передать ему фокус) никаким способом. Значения отключенных управляющих элементов при отправке формы серверу не пересылаются.

В том случае, если необходимо передать значение управляющего элемента вместе с формой и одновременно закрепить доступ к нему пользователя, используется логический атрибут **readonly**. Наличие атрибута **readonly** определяет, может ли пользователь изменить значение управляющего элемента. Тем из них, для которых установлен атрибут **readonly**, может осуществляться передача фокуса, но изменяться в значении такие управляющие элемен-

ты не могут. Атрибут **readonly** могут содержать в списке своих атрибутов следующие HTML-теги: INPUT и TEXTAREA.

13.11. Как визуально форматировать формы?

Привлекательность документа, в котором используются формы, во многом зависит от размеров полей ввода и компоновки элементов внутри формы. Радующего глаз дизайна можно достигнуть и применяя простые форматирующие HTML-элементы разметки; такие как BR, PRE, P и т.п.

Пример использования тега PRE:

```
<!DOCTYPE html>
<html>
   <head>
   <title>
   Пример использования тега PRE для форматирования формы
   </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Ввод личной информации </H2>
      <form>
      <pre>
      Фамилия <input type=text name="lastname">
      Имя     <input type=text name="firstname">
      Электронная почта<input type=text name="email">
      </pre>
      <fieldset>
      <legend><I> Адрес </I></legend>
      <pre>
      Улица        <input type=text name="street">
      Дом        <input type=text name="home" size=5>
      Город        <input type=text name="city">
         Страна    <input type=text name="country">
         Планета   <input type=text name="planet">
      Планетная система    <input type=text name="planetary
sistem">
      Галактика    <input type=text name="galaxy">
      </pre>
```

```
      </fieldset>
      <BR>
      <input type=reset>
      <input type=submit>
      </form>
   </body>
</html>
```

Однако визуальное представление сложных форм рекомендуется разрабатывать с помощью списков и таблиц с невидимыми границами. В этом случае управляющие элементы формы задаются как элементы списка или содержимое ячеек таблицы соответственно.

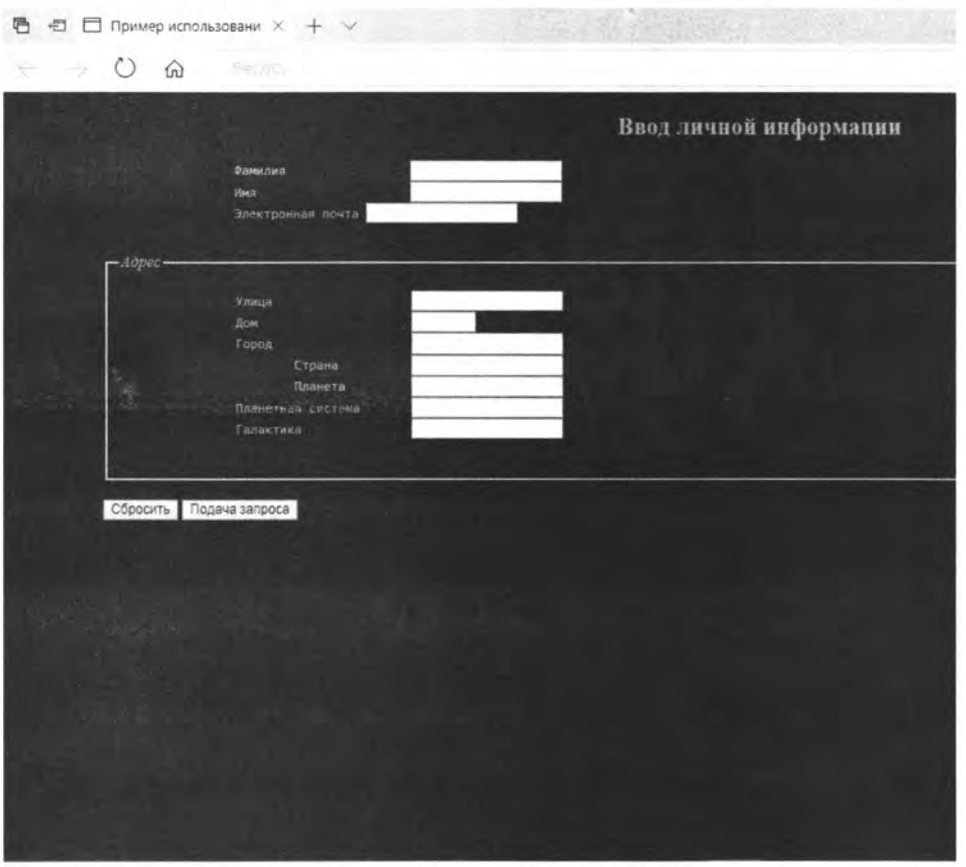


Рис. 13.21. Пример использования тега PRE для форматирования формы

Пример использования элементов списка определений для форматирования формы:

```
<!DOCTYPE html>
<html>
  <head>
  <title>
  Пример использования элементов списка определений для
форматирования формы
  </title>
  </head>
  <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
     <H2 align=center> Заполните небольшую анкету </H2>
     <form>
     <DL>
     <fieldset>
     <legend>
     <I><FONT size=-1><B> Читали ли Вы Ф.М.Достоевского? </
B></FONT></I>
     </legend>
     <DT><input type=radio name="dost" value="none"
onclick="closedost.js">
     Я не читал ни одного произведения Ф.М. Достоевского.
     <DT>
     <input type=radio name="dost" value="yes"
onclick="opendost.js" checked>
     Я читал произведения Ф.М.Достоевского. Далее укажите
какие именно:
     <DD><input type=checkbox name="Idiot" value="yes">
"Идиот"
     <DD><input type=checkbox name="Karamazov" value="yes">
"Братья Карамазовы"
     <DD><input type=checkbox name="Poor_people" value="yes">
"Бедные люди"
     <DD><input type=checkbox name="Bes" value="yes"> "Бесы"
     <DD><input type=checkbox name="Crime" value="yes"
checked> "Преступление и наказание"
     <DD><input type=checkbox name="White_night" value="yes">
"Белые ночи"
     <DD><input type=checkbox name="Junior" value="yes">
"Подросток"
     </fieldset>
     <fieldset>
     <legend>
```

```
     <I><FONT size=-1><B> Читали ли Вы И.С.Тургенева? </B></
FONT></I>
     </legend>
     <DT>
     <input type=radio name="turg" value="none"
onclick="closeturg.js" checked> Я не читал ни одного
произведения И.С.Тургенева.
     <DT><input type=radio name="turg" value="yes"
onclick="openturg.js"> Я читал произведения И.С.Тургенева.
Далее укажите какие именно:
     <DD><input type=checkbox name="Slip" value="yes">
"Записки охотника"
     <DD><input type=checkbox name="Fathers" value="yes">
"Отцы и дети"
     <DD><input type=checkbox name="Mumu" value="yes" checked>
"Муму"
     <DD><input type=checkbox name="Jack" value="yes">
"Дворянское гнездо"
     <DD><input type=checkbox name="Rudin" value="yes">
"Рудин"
```

Рис. 13.22. Пример формы, отформатированной с помощью элементов списка определений

```
      <DD><input type=checkbox name="Acja" value="yes"> "Ася"
      </fieldset>
      <input type=submit>
      </form>
   </body>
</html>
```

Примечание: opendost.js и closedost.js – вымышленные скрипты, которые включают и отключают (устанавливают и отменяют атрибут **disabled**) все управляющие элементы указания прочитанных произведений Ф.М.Достоевского. Аналогично используются скрипты openturg.js и closeturg.js для включения и отключения флажков выбора произведений И.С.Тургенева.

Пример форматирования формы средствами упорядоченных списков:

```
<!DOCTYPE html>
<html>
      <head>
      <title>
      Пример использования элементов упорядоченного списка
для форматирования формы
      </title>
      </head>
      <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
            <H2 align=center> Укажите десятку своих самых
любимых актеров </H2>
            <form>
            <OL type=I>
            <LI><B>Русские актеры:</B>
            <OL>
            <LI><input type=text name="1rus">
            <LI><input type=text name="2rus">
            <LI><input type=text name="3rus">
            <LI><input type=text name="4rus">
            <LI><input type=text name="5rus">
            </OL>
            <LI><B>Иностранные актеры</B>
            <OL>
            <LI><input type=text name="1foreign">
            <LI><input type=text name="2foreign">
            <LI><input type=text name="3foreign">
            <LI><input type=text name="4foreign">
            <LI><input type=text name="5foreign">
```

```
            </OL>
            </OL><BR>
            <input type=reset>
            <input type=submit>
            </form>
      </body>
</html>
```

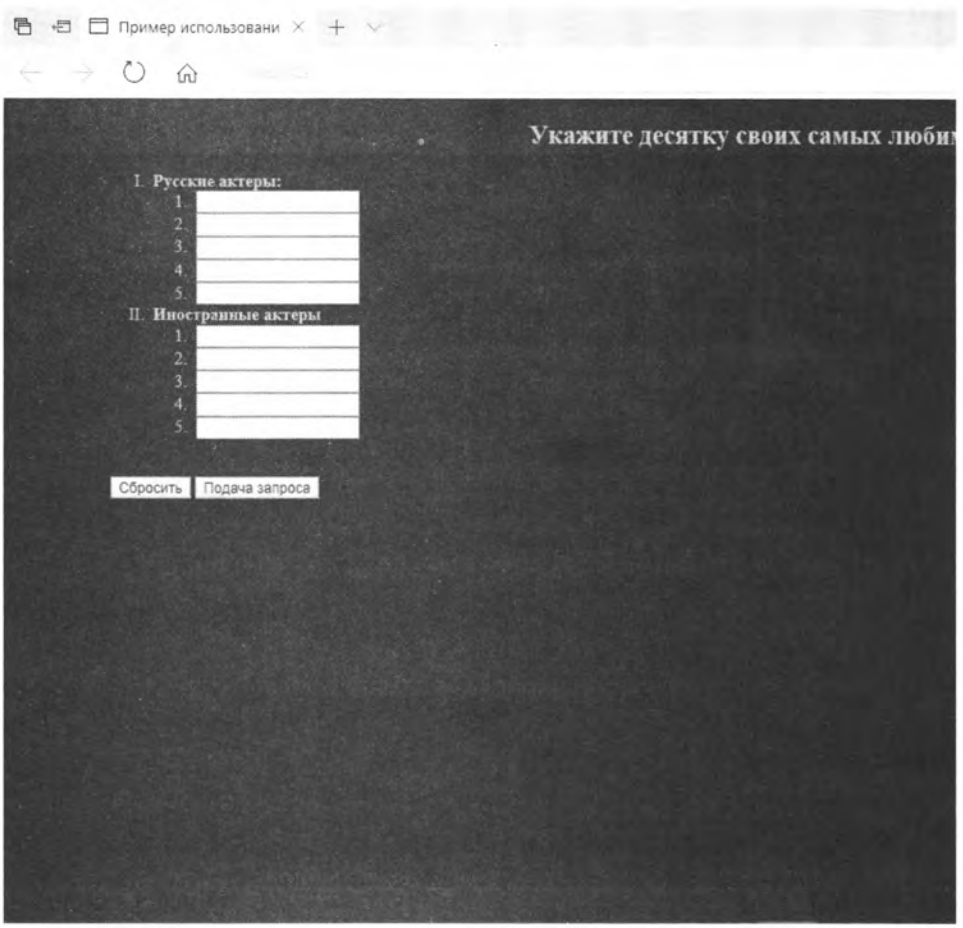


Рис. 13.23. Пример формы, отформатированной с помощью HTML-элементов упорядоченного списка

Использовав в вышеприведенном примере еще и таблицы, можно достичь большей привлекательности формы:

```
<!DOCTYPE html>
<html>
   <head>
   <title>
   Пример использования таблиц для форматирования форм
   </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
      <H2 align=center> Укажите десятку своих самых любимых
актеров </H2>
      <form>
      <center>
      <table Cellspacing=20 Cellpadding=10 bgcolor=darkgray>
      <TR>
      <TH bgcolor=white>
       I. Русские актеры:<BR>
      <OL>
      <LI><input type=text name="1rus">
      <LI><input type=text name="2rus">
      <LI><input type=text name="3rus">
      <LI><input type=text name="4rus">
      <LI><input type=text name="5rus">
      </OL>
      <TH bgcolor=white><BR>
      II.  Иностранные актеры:<BR>
      <OL>
      <LI><input type=text name="1foreign">
      <LI><input type=text name="2foreign">
      <LI><input type=text name="3foreign">
      <LI><input type=text name="4foreign">
      <LI><input type=text name="5foreign">
      </OL>
      <BR>
      <TR >
      <TH colspan=2>
      <input type=reset>
      <input type=submit>
      </table>
      </center>
      </form>
   </body>
</html>
```

Рис. 13.24. Пример простейшего использования таблиц для форматирования форм

Как уже упоминалось, в документе может присутствовать несколько форм. Продемонстрируем этот момент на примере:

```
<!DOCTYPE html>
<html>
   <head>
   <title>
   Пример использования двух форм в одном документе.
   </title>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <form>
   <fieldset>
   <legend><B><I> Форма 1 </I></B></legend>
   <center><B> Личная информация </B></center>
   <PRE>
```

```
    Фамилия <input type=text name="lastname">
    Имя     <input type=text name="fistname">
    Город     <input type=text name="city">
    Адрес     <input type=text name="adress">
    Телефон <input type=text name="phone">
</PRE>
<center>
<input type=submit><input type=reset>
</center><BR>
</fieldset>
</form>
<form>
<fieldset>
<legend><B><I>Форма 2</I></B></legend>
<center><B>Служебная информация</B></center>
<PRE>
Название фирмы    <input type=text name="firma">
Юридический адрес      <input type=text name="Ur_adress">
```

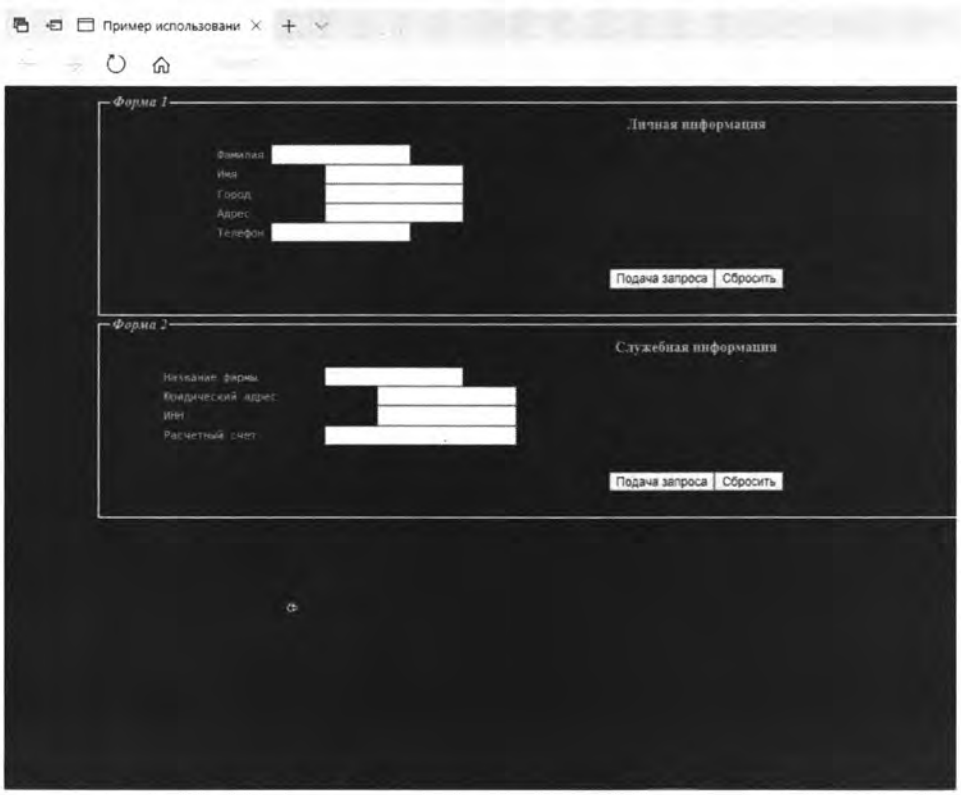


Рис. 13.25. Пример документа с двумя формами

```
    ИНН          <input type=text name="INN">
    Расчетный счет    <input type=text name="account" size=30>
    </PRE>
    <center>
    <input type=submit><input type=reset>
    </center><BR>
    </fieldset>
    </form>
    </body>
</html>
```

CSS3

Глава 1.

Типы данных и синтаксис CSS3

В этой главе вы узнаете:

- *Что такое CSS?*
- *Как подключить каскадные таблицы стилей к HTML-документам?*
- *Что такое синтаксические правила, присутствующие в CSS3?*
- *Как обрабатываются синтаксические ошибки?*
- *Про допустимые значения величин, используемых в CSS3*

1.1. Что такое CSS?

Для начала попробуем понять, что такое каскадные таблицы стилей и для чего они нужны. Таблица стилей представляет собой написанный на языке CSS3 набор правил, применяемых к HTML-тегам документа, к которому эта таблица подключена. Правила эти задают визуальное представление содержимого HTML-тегов. Или, иначе говоря, то, как они будут выглядеть на экране монитора (цвет, рамки, размер, выравнивание, шрифт, отступы, видимость и т.п.). Благодаря CSS3 эти параметры не надо задавать отдельно для каждого тега через его атрибуты. Таким образом, каскадные таблицы стилей позволяют отделять содержимое HTML-документа от описания его внешнего вида. К тому же, благодаря этому достигается значительное удобство, компактность описания визуальных свойств HTML-элементов и оперативность их изменения.

Рассмотрим это на примере.

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

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

Сравните:

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

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

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

Ко всему прочему умелое использование возможностей CSS3 (обособленно или в сочетании с DHTML) позволяет реализовать оригинальные визуальные эффекты, описание которых произведено в следующих уроках.

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

Каскадные таблицы стилей могут располагаться либо в заголовке HTML-документа (в содержимом тега STYLE), либо во внешнем файле с расширением .css. В этом случае, подключение таблицы стилей осуществляется тегом заголовка LINK.Использование внешних таблиц стилей особенно ак-

туально для многостраничных сайтов, все страницы которого должны быть выдержаны в одном дизайнерском решении. Поэтому вместо того, чтобы в коде каждой страницы писать одинаковые стилевые настройки, достаточно написать их один раз, поместить во внешний css-файл и подключить ко всем страницам сайта.

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

1.2. Как подключить каскадные таблицы стилей к HTML-документам?

Для того, чтобы браузер применял правила какой-либо таблицы стилей к HTML-документу, необходимо привязать таблицу и документ друг к другу.

Существует четыре метода, которыми это можно сделать:

- *Внедрение* - задание таблицы стилей непосредственно в заголовке самого HTML- документа, в качестве содержимого тега STYLE;
- *Присоединение* - таблица стилей находится во внешнем файле и присоединяется к HTML-документу через тег LINK. При этом CSS-файл с внешней таблицей стиля всегда сопровождает HTML-файл документа;
- *Импортирование* - суть этого метода заключается в том, что текст таблицы стилей, находящейся во внешнем файле на сервере, импортируется с помощью Css-свойства @import внутрь текста HTML-файла;
- *Поэлементное задание стиля*, когда для всех HTML-тегов определен атрибут STYLE. Через него, используя синтаксис CSS3, можно задавать (или переопределять) стиль для каждого тега индивидуально.

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

Например:

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

Присоединение внешних таблиц стилей к HTML-документу осуществляется тегом LINK и имеет следующий вид:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<TITLE> Пример подключения таблицы стиля </TITLE>
<LINK rel ="stylesheet" type ="text/css" href = "file_of_style.
css">
</HEAD>
<BODY>
.....текст документа.....
.....текст документа.....
.....текст документа.....
</BODY>
</HTML>
```

Где *file_of_style.css* – имя файла, в котором хранится подключаемая таблица стилей. Файл, по сути, является текстовым и содержит перечень правил CSS3.

Пример текста файла с внешней таблицей стиля:

```
body {font-family:sans-serif
font-size:14 pt}
p {background - color: yellow}
```

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

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

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

Например:

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

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

1.3. Синтаксические правила, присутствующие в CSS3

В рамках данного урока будет рассмотрена грамматика написания таблиц стилей CSS3. В основе всех версий CSS лежит общий для всех набор синтаксических правил. Именно благодаря этому обстоятельству обеспечивается преемственность различных версий CSS. В том случае, если браузер поддерживает ранние версии CSS и не поддерживает новые возможности,

то последние будут просто им проигнорированы. Однако правила, которые браузер в состоянии применить, он применит.

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

Эй-ти правила начинаются с ключевого символа @, непосредственно за которым следует указание имени правила. Затем правило *Эй-ти* включает в себя все, что находится до первого символа точки с запятой или другого правила.

Следует помнить, что все правила @import, которые находятся внутри блока определений другого @правила, будут проигнорированы браузером.

Пример недопустимого задания:

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

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

Пример неправильного задания:

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

Пример правильного задания:

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

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

селектор {блок определений}

Селектором служит название HTML-тега, или комбинация названий HTML-тегов, для которых и задается правило форматирования. Блок определений начинается с левой фигурной скобки "{" и заканчивается правой

фигурной скобкой "}". Все, что находится перед левой фигурной скобкой, считается селектором.

Пример:

```
H1,H2    {color: blue; font family: Times New Roman }
селектор                    блок определений
```

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

название свойства: значение свойства

Причем между "названием", двоеточием и "значением" может находиться любое количество пробелов.

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

Например, следующий набор правил:

```
p{font-size:14 pt}
p{font-family: Arial}
p{font-color: yellow}
p{background-color: blue}
```

Эквивалентен одному такому правилу

```
p     { font-size:14 pt;
font-family: Arial;
font-color: yellow;
background-color: blue}
```

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

Например,

```
IMG {}
IMG - селектор, {}- блок определений
```

Что касается регистра, то каскадные таблицы стилей CSS3 не учитывают регистр. Это значит, что названия правил, селекторов и их значения можно указывать как прописными, так и строчными буквами. Чувствительными к регистру могут оказаться объекты, присутствующие в таблице стилей, но не являющиеся объектами языка CSS.

При написании текста таблиц стилей допустимо использование комментариев.

Они должны начинаться символом "/*" и заканчиваться символом "*/".

Например,

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

В CSS допустимо также использование комментариев <!---и--->, установленных в HTML.

1.4. Как обрабатываются синтаксические ошибки?

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

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

```
b {font-color: blue;
thick:15; /*thick -  несуществующее свойство*/
font-size:14 pt}
```

Будет воспринято браузером как:

```
b {font-color: blue;
font-size:14 pt}
```

- Если для определенного в CSS3 свойства указано недопустимое значение, то также будет проигнорировано содержащее его объявление. Например, фрагмент таблицы стилей:

```
p{font-color:black}
p{background-color:"lightgrey"} /* некорректное задание,
т.к. ключевое слово заключено в кавычки */
p{font-size:A3x} /* неверное задание размера шрифта*/
```

после синтаксической проверки примет следующий вид:

```
P{font-color: black}
P{}
P{}
```

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

```
@superstyle {
P { font-color: black;
background-color: lightgrey}
H3{font-color: yellow}
}
BODY {font-family:Arial}
```

Будет воспринято браузером просто как:

```
BODY {font-style:Arial}
```

так как @ правило @superstyle в CSS3 не определено.

1.5. Допустимые значения величин, используемых в CSS3

В этом разделе будут описаны значения, которые могут принимать те или иные свойства. А также их размерности и допустимые интервалы. При первом ознакомлении с CSS3 этот раздел можно пропустить и возвращаться к нему по мере надобности. Но для более продуктивного ознакомления с возможностями CSS3 рекомендуется этот раздел прочесть предварительно.

Числа

Здесь ничего нового. В CSS3 могут использоваться целые и вещественные числа, представленные в десятичной форме исчисления. Допустимые интервалы значений индивидуальны для каждого свойства.

Единицы измерения длины

В CSS3 допустимо использование двух типов задание длины: относительное задание и абсолютное.

Относительное задание длины подразумевает ее задание относительно чего-то. К относительным единицам относятся:

- **em** - 1em равен используемому значению свойства font-size элемента, к которому данный em-размер применяется;
- **ex** - 1ex равен размеру высоты строчной буквы x для используемого шрифта;
- **px** - указывает размер в пикселях и, соответственно, зависит от разрешения экрана;
- **rem** – равен используемому значению размера шрифта, заданного для элемента html.

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

- **cm** - сантиметр
- **mm**– миллиметр
- **in** - дюйм (1 дюйм равен 2.54 см)
- **pt** - пункт (point) В CSS2 1pt =1/72 дюйма
- **pc** - пика. В CSS2 1 пика =12 пунктам

Задание URL в контексте CSS3

Указание URL-адреса в CSS3 осуществляется в следующем формате:

```
url (http://techrussia.org/)
```

Например, указать адрес фонового изображения можно следующим способом:

```
BODY {background-image: url (http://techrussia.org/assets/img/
Robots/VR.png)}
```

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

Указание цвета

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

- указание цвета с помощью ключевых слов

  Пример: H3{color: blue}

- указание цвета с помощью шестнадцатеричного задания его RGB-кода. Например, H3 {color: # FFFFFFF}

- указание цвета с помощью десятеричного задания его RGB-кода.

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

  При этом используются целые числа в диапазоне от 0 до 255

- указание цвета с помощью процентного задания насыщенности каждого из цветов в RGB-коде.

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

  При этом используются вещественные числа в интервале от 0% до 100%

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

Например, H3 {color: rgb (0,300,0)}будет уменьшено до

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

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

Углы

Углы в каскадных таблицах стилей, написанных на языке CSS3, могут быть заданы в следующих единицах измерения:

deg - градусы

rad - радианы

grad – грады

turn– поворот круга

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

Время

Задание времени (не путать с датой) может осуществляться либо в секундах (s) либо в миллисекундах (ms).1s=1000ms.

CSS3

Глава 2.

Селекторы, псевдоэлементы и псевдоклассы

В этой главе вы узнаете:

- *Что такое простой селектор?*
- *Что такое универсальный селектор?*
- *Что такое селектор классов?*
- *Что такое селектор ID-имен?*
- *Что такое селекторы контекстного окружения?*

2.1. Что такое простой селектор?

Как уже упоминалось, все обычные правила CSS состоят из селекторов, и, соответствующих им блоков определений. В этом разделе будет произведено детальное рассмотрение используемых в CSS селекторов. Простейший селектор представляет собой название HTML-тега. Такой селектор называется "простым селектором".

Применение простых селекторов является наиболее широко употребляемым. Однако язык CSS3 позволяет использовать более широкие возможности для создания таблиц стилей. Например, можно описать свойства HTML-тега, который является дочерним по отношению к другому HTML-тегу. Также можно использовать селектор, благодаря которому задаваемые параметры будут применяться при определенных действиях пользователя. Определенные селекторы позволяют обращаться к различным группам HTML-элементов. Например, ко всем элементам с определенным значением атрибута **id**.

2.2. Что такое универсальный селектор?

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

Например, записи:

*. myclass и .myclass – эквиваленты

*# id_name и # id_name – так же эквивалентны

Пример задания:

```
<!DOCTYPE html>
<html>
   <head>
      <title>
      Пример заключения таблицы стиля в SGML-комментарии
      </title>
      <style type ="text/css">
      * {color=olive}       /* правило с использованием
универсального селектора*/
      H1 {color=red}  /* правило с использованием простого
селектора*/
      *.myclass {font-color=blue}       /*использование
универсального селектора в сочетании с селектором классов*/
      </style>
   </head>
   <body>
   ..... .текст документа.....
   ..... .текст документа.....
   ..... .текст документа.....
   </body>
</html>
```

2.3. Что такое селектор классов?

Благодаря использованию селектора классов разработчик может обращаться к группе разнородных HTML-элементов, принадлежащим к одному классу (имеющим одноименное значение атрибута **class**). Селектор классов имеет следующий общий вид:

- сначала, через запятую, указываются названия HTML-элементов, а затем, через точку, следует имя класса, к которому эти элементы должны принадлежать, чтобы к ним было применено данное правило.
- тег.имя класса {определение; определение; определение}

Например, чтобы правило применялось ко всем элементам, принадлежащим к классу **superclass**, используется следующая запись: ".superclass {определение}"

Или другой пример:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
    Пример применения селектора классов
    </title>
    <style type ="text/css">
    h3{color:green}
    h3.titlepage{color:red}
    h3.indexpage{color:blue}
    </style>
```

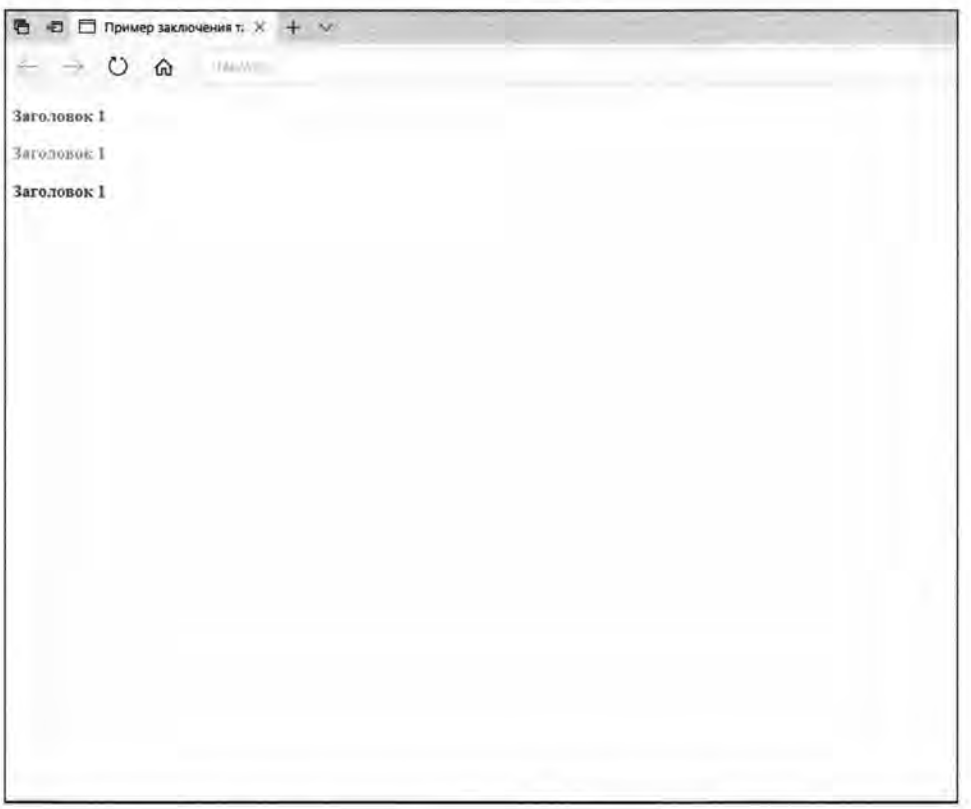


Рис. 2.1. Пример применения селектора классов

```
  </head>
  <body>
    <h3> Заголовок 1</h3>
    <h3 class=titlepage> Заголовок 1</h3>
    <h3 class=indexpage> Заголовок 1</h3>
  </body>
</html>
```

При таком задании, все заголовки H3, принадлежащие к классу **titlepage** (class = titlepage) будут отображаться красным цветом. Принадлежащие к классу **indexpage** (class = indexpage) – синим цветом. Все остальные H3-заголовки будут иметь зеленый цвет.

Можно также использовать классы и без указания тега. При такой записи класс можно применять к любому тегу:

.Имя класса {определение; определение; определение}

2.4. Что такое селектор ID-имен?

Селектор ID-имен в своем применении аналогичен селектору классов. Отличие заключается в том, что селектор ID-имен фильтрует HTML-теги не по классу (атрибуту **class**) , а по **id**-именам (атрибуту **id**).

Синтаксис селектора ID-имен имеет следующий вид:

сначала пишется имя HTML-тега, а затем, через символ "#", прописывается его **id**-имя.

тег#id_имя {определение; определение; определение}

Например, правило:

```
P#famous {font-family: Heretz}
```

будет применено только к тем HTML-тегам P, у которых для атрибута id указано значение "famous".

Чтобы обратиться ко всем HTML-тегам, имеющим атрибут **id** = "famous", достаточно следующей записи:

#famous{определение;

определение;

.....

определение}

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

Например,

```
H1, P, Q {color: lightgrey;
font-family: Arial}
.my-class, B {color: black}
```

2.5. Что такое селекторы контекстного окружения?

Возможности CSS3 позволяют учитывать расположение HTML-элементов в иерархическом дереве документа при задании их визуальных характеристик, а также учитывать их контекстное окружение.

HTML-тег, заданный содержимым другого HTML-тега, является его потомком. Исходя из этого правила, каждый HTML-документ имеет свое иерархическое дерево.

В CSS возможно использование правил, которые будут применяться только к HTML-тегам, являющихся потомками других определенных тегов. При этом, к таким же HTML-тегам, не являющимся потомками указанных тегов данное правило применяться не будет.

При задании такого правила, через пробел указывается селектор родительского тега и селектор тега-потомка.

Например, разработчику нужно установить синий цвет для цитат, только если они содержатся в тексте тега PRE, то есть если тег CITE входит в состав тега PRE:

```
<pre>
отформатированный текст
<cite> цитата </cite>
```

```
отформатированный текст
</pre>.
```

Для этого достаточно использовать следующее правило:

```
PRECITE {color: blue}
```

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

Например, правило

```
DIVPEM {color: yellow;
    font-family: Italic}
```

будет применено только к тем тегам EM, которые являются потомками тега **P**, который, в свою очередь, должен являться потомком тега DIV. Чтобы обратиться ко всем тегам EM, являющимися потомками второго уровня элемента, используется запись DIV * B. Символ "*" при этом с обеих сторон выделен пробелами.

Таким образом, правило, использующее иерархию HTML-тегов документа, имеет следующий общий вид:

селектор селектор ... селектор {определение; определение; определение}

CSS3 также предоставляет возможность учитывать расположение HTML-тегов в контексте HTML-документа. А именно для того, чтобы правило применялось к определенному HTML-тегу только в том случае, если он следует вслед за другим определенным HTML-тегом. Например, чтобы обратиться ко всем неупорядоченным спискам одного HTML-тега, которым предшествует заголовок H3, достаточно сделать следующую запись:

```
H3+UL {объявление;
    объявление;
        .....
объявление}
```

Теги H3 и UL называются сестринскими тегами. Символ "+" с обеих сторон выделяется пробелами.

Таким образом, правило учитывающее предыдущий HTML-элемент для описываемого HTML-элемента, имеет следующий общий вид:

селектор + селектор + ... + селектор {определение; определение; определение}

2.6. Что такое псевдоэлементы и псевдоклассы?

В рамках спецификации CSS3 определено некоторое количество псевдоэлементов и псевдоклассов, которые представляют собой элементы и классы, не входящие в иерархическое дерево HTML-документа, и не образующие в нем структурных единиц. Иными словами, они являются специфичными элементами и классами языка CSS3 и используются им только в своих целях для задания специальных визуальных эффектов. Например, псевдоэлемент: first-letter представляет собой первую букву текста фрагмента: first-line - первую строку и т.п.

Условно все псевдоклассы делятся на три группы:

- определяющие состояние элементов;
- имеющие отношение к дереву элементов (класс first-child);
- указывающие язык текста (класс lang).

Все псевдоклассы, определяющие состояние элементов принадлежат одному из пяти псевдоклассов:

- классу *visited* посещенных ссылок
- классу *link* не посещенных ссылок
- классу *active* активных ссылок
- классу *focus* для ссылок, при получении им фокуса
- классу *hover* ссылок, над видимым содержимым которых, находится курсор мыши

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

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

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

```
<!DOCTYPE html>
<html>
   <head>
      <title>
      Пример использования псевдоклассов
      </title>
      <base href= "http://techrussia.org">
      <style type ="text/css">
      a:visited {color:lightgrey}
      a:link {color:blue}
      a:active {color:yellow}
      a:hover {color:red}
      </style>
   </head>
   <body leftmargin=100 rightmargin=100 text=#78bed0
bgcolor=#112232>
   <p><a href="/#directions"> Ссылка на направления TechRussia
</a><p>
   <p><a href="/#members"> Ссылка на информацию для участников
TechRussia </a><p>
   <p><a href="/#faq"> Ссылка на часто задаваемые вопросы о
TechRussia </a><p>
   </body>
</html>
```

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

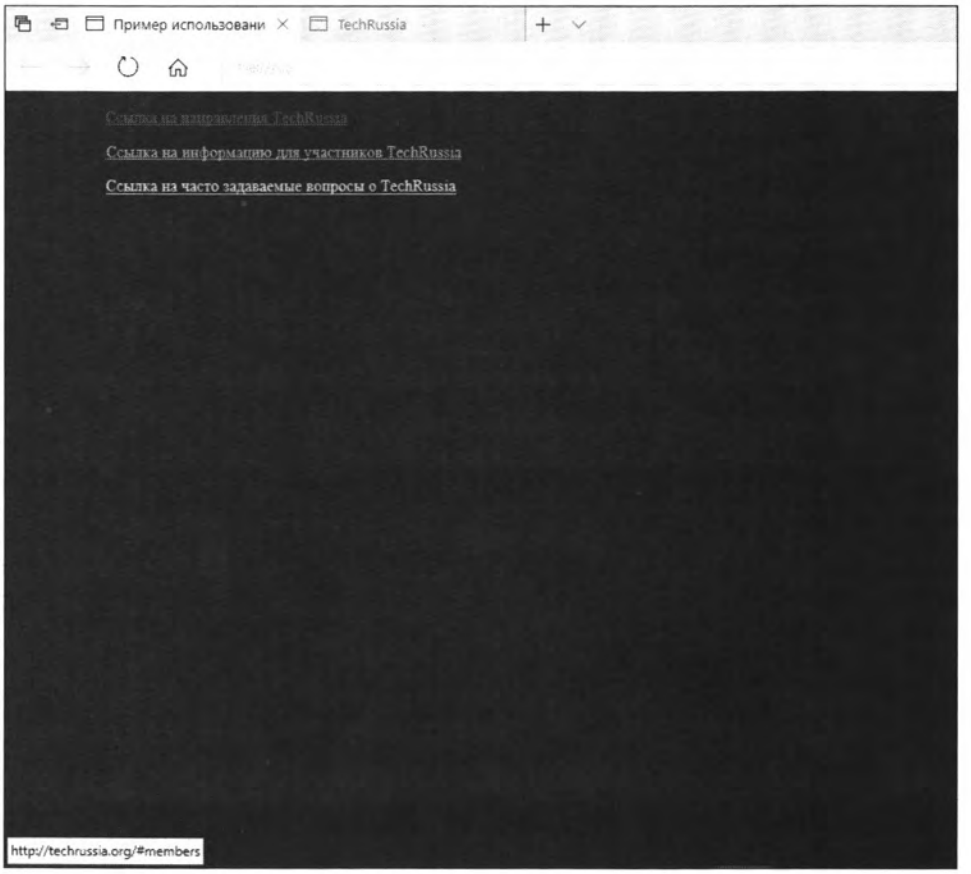


Рис. 2.2. Применение псевдоклассов, определяющих состояние элемента

CSS3

Глава 3.

Правила каскадирования и аппаратно-зависимые таблицы стилей

В этой главе вы узнаете:

- *Что такое правила каскадирования?*
- *Как создать аппаратно-зависимую таблицу стилей?*

3.1. Что такое правила каскадирования?

Как уже упоминалось, к одному HTML-документу может быть применено несколько каскадных таблиц стилей разного или одинакового типа.

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

1 этап: Синтаксический анализ документа и обработка синтаксических ошибок

2 этап: Построение иерархического дерева HTML-тегов, образующих данный документ

3 этап: Применение таблицы стилей в соответствии с правилом установки каскадирования и иерархическим деревом документа.

Иерархическое дерево документа используется для передачи установок родительского тега тегам-потомкам. То есть благодаря ему реализуется процесс наследования.

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

Если для одного HTML-тега указано несколько одинаковых правил в рамках одной таблицы стилей, то приоритет имеет правило заданное последним.

Все таблицы стилей в качестве своего источника могут иметь разработчика, пользователя или браузер. Самым низким приоритетом обладают стилевые установки, используемые браузером по умолчанию. Таблицы стилей, заданные разработчиком, имеют приоритет над таблицами, применяемыми к документу пользовательскими таблицами стилей.

При задании и подключении нескольких таблиц стилей, каждая последующая заданная (или подключенная) таблица имеет приоритет над предыдущей. Говоря точнее, каждый последующий тег STYLE или LINK имеет приоритет над предыдущим.

В этом отношении между таблицами, задаваемыми тегами STYLE и LINK, различий не делается. Т.е., если задание подключенной таблицы (тег LINK) следует после внедренной таблицы (тег STYLE), то подключенная имеет приоритет над внедренной. Верно также и обратное.

В рамках одного тега STYLE с помощью правила @import может быть импортировано несколько таблиц стилей, и после чего могут следовать другие правила, задаваемые непосредственно в этом теге STYLE. В рамках одного тега STYLE все импортированные установки имеют меньший приоритет по отношению к установкам, заданным в нем непосредственно. Надо помнить, что все правила @import задаются в таблице стилей до определения остальных правил.

Если в тег STYLE импортировано несколько CSS-таблиц, то каждая последующая импортируемая таблица обладает большим приоритетом над предыдущей.

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

<H3 class="vas" id="ret"></H3> могут быть применены следующие правила:

```
H3 {color:gray}
H3.vas {color:red}
H3#ret{color:green}
H3, H2, H1 {color:blue}
```

Правило H3 {color:gray} относится только к заголовкам третьего уровня, то есть является более специфичным по сравнению с предыдущим. Правило

H3.vas {color:red} применяется только к тегам H3, принадлежащим к классу "vas", то есть является более специфичным по сравнению с предыдущим

Самым специфичным является правило H3#ret {color:green}, которое применяется к тегам H3 с **id**-именем *ret*. Все эти правила относятся к тегу <H3 class="vas" id="ret"></H3>. Но если они заданы все вместе, то к этому элементу будет применено правило H3#ret {color:green}, независимо от того, в каком порядке эти правила заданы.

К HTML-тегам применяются относящиеся к ним правила, обладающие наибольшей специфичностью, независимо от порядка задания.

Специфичность правила определяется по следующей схеме:

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

Псевдоэлементы и псевдоклассы считаются как обычные элементы и классы.

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

Например:

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

3.2. Как создать аппаратно-зависимую таблицу стилей?

Возможности языка CSS3 позволяют разработчикам создавать аппаратно-зависимые таблицы стилей. Или, говоря другими словами, таблицы стилей для определенных устройств вывода: монитора, принтера, телевизора, синтезатора речи и т.п. В силу технических особенностей этих устройств, один и тот же документ может иметь у них разное визуальное представление внешний вид. Например, общеизвестным считается тот факт, что для отображения текста на экране монитора требуется больший размер шрифта, чем для его печати. Поэтому для достижения наилучшего качества представления документа, рекомендуется создавать стилевые таблицы под каждое устройство вывода, для которых он может предназначаться. Указать целевое устройство можно с помощью правила @media, например,

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

Внешнюю таблицу стилей можно определить для определенного целевого устройства с помощью атрибута media тега LINK.

Например,

```
<link rel = "stylesheet" type = "text/css" media = "print"
href = "my style file.css">
```

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

Например,

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

Если таблица предназначается нескольким устройствам разного типа, то их перечисление осуществляется через запятую. Например,

```
@media print, screen {
```

```
BODY {font-color: san serif;
   font-color: 12 pt;
         font-color: black}
                        }
```

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Команда "Победители во всём!" </title>
    <style type="text/css">
    body {text-align:center;
      background-color:#112232;
      color:#78bed0}
    @media print{body{color:black}}
    </style>
  </head>
  <body>
    <h1> Информация о команде "Победители во всём!" </h1>
    <h2> Иван Смирнов <br> Капитан команды </h2>
```

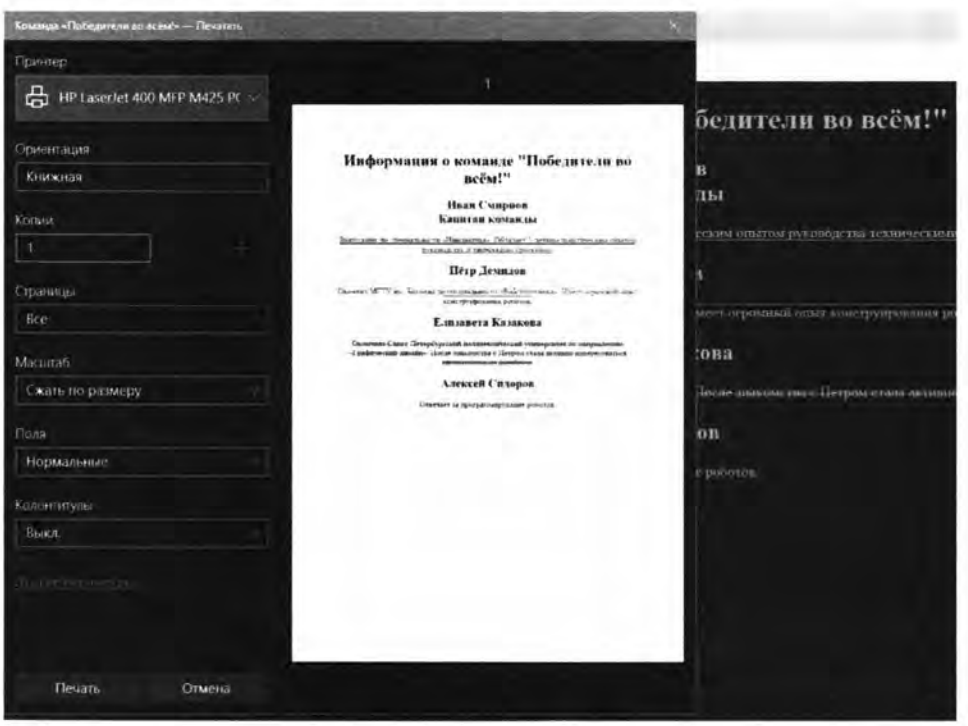


Рис. 3.1. Демонстрация аппаратно-зависимых таблиц стилей (отличается цвет текста на экране и на принтере)

```
    <p style="text-decoration:underline"> Выпускник по
специальности "Инноватика". Обладает 5-летним практическим
опытом руководства техническими проектами.</p>
       <h2> Пётр Демидов </h2>
    <p style="text-decoration:overline"> Окончил МГТУ им.
Баумана по специальности "Робототехника". Имеет огромный опыт
конструирования роботов.</p>
       <h2> Елизавета Казакова </h2>
    <p style="text-decoration:line-through"s> Окончила Санкт-
Петербургский политехнический университет по направлению
"Графический дизайн". После знакомства с Петром стала активно
интересоваться промышленным дизайном. </p>
       <h2> Алексей Сидоров </h2>
    <p> Отвечает за программирование роботов. </p>
  </body>
</html>
```

Типы устройств, распознаваемые CSS3

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

- all - Этот тип предназначен для обозначения устройств всех типов. Его задание говорит о том, что таблица стилей предназначена для воспроизведения всеми возможными устройствами вывода. Данное значение используется по умолчанию;
- print - Обозначает печатные устройства и говорит о том, что таблица стилей будет применена к документу при его печати;
- screen - обозначает цветной экран монитора;
- speech – речевые синтезаторы.

Пример использования:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Пример аппаратно-зависимого документа </title>
    <style type="text/css">
    @media screen {
    правила, применяемые к документу, при выводе его на экран
монитора
```

```
      правила, применяемые к документу, при выводе его на экран
монитора
      правила, применяемые к документу, при выводе его на экран
монитора
      }
      @media print {
      правила, применяемые к документу, при его печати
      правила, применяемые к документу, при его печати
      правила, применяемые к документу, при его печати
      }
      @media speech {
      правила, применяемые к документу, при его звуковом
воспроизведении
      правила, применяемые к документу, при его звуковом
воспроизведении
      правила, применяемые к документу, при его звуковом
воспроизведении
                  }

      правила, применяемые к документу, вне зависимости от
устройства вывода
      правила, применяемые к документу, вне зависимости от
устройства вывода
      правила, применяемые к документу, вне зависимости от
устройства вывода
      </style>
   </head>
   <body>
      .....текст документа.....
      .....текст документа.....
      .....текст документа.....
   </body>
</html>
```

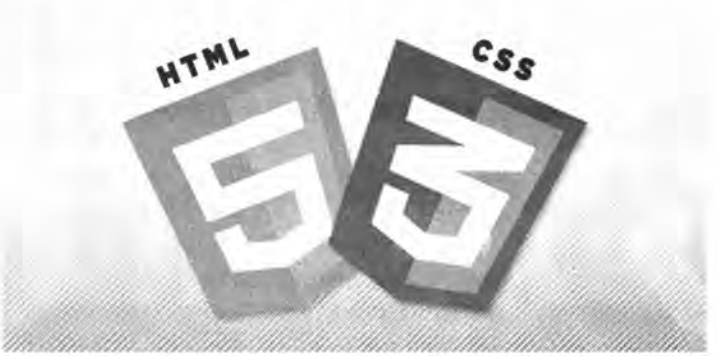
CSS3

Глава 4.

Форматирование документа средствами CSS3

В этой главе вы узнаете:

- *Про блочную модель визуального представления документа*
- *Как задать свойства полей?*
- *Как задать свойства отступов?*
- *Как задать свойства границ?*
- *Как задать тип линии границ?*
- *Как задать цвет текста и фона?*



4.1. Блочная модель визуального представления документа

Одной из основных технологий, составляющих суть языка CSS3, является блочная модель отображения документов. Согласно этой модели HTML-теги уровня блока, состоящие в иерархическом дереве документа, отображаются в виде прямоугольных блоков, к которым могут быть применены стилевые установки.

Сам блок представляет собой несколько вложенных друг в друга прямоугольных областей. В самой внутренней области и помещается информативное содержание HTML-тега.

Под HTML-теги, являющиеся строковыми, стилевые блоки не выделяются.

При этом, текст абзаца, представляющий собой информативную область блока, имеет вокруг себя три прямоугольных области: отступ, границу и поле. Каждая из них может быть разбита на четыре сегмента: левый, правый, верхний и нижний. В дальнейшем левый сегмент области отступа будем называть левым отступом, правый сегмент области границы - правой границей и т.д. Возможности языка CSS3 позволяют для каждой области устанавливать свои визуальные параметры (цвет, фон, размер и т.д.).

Стилевое оформление фона различных областей определяется следующим образом:

- Фоновое оформление информативной области задается CSS-свойствами HTML-тега, начинающимися со слова **background**;
- Область отступов также использует для своего фона свойства **background** HTML-тега, породившего данный блок;
- Область границы сама имеет свои свойства, устанавливающие ее фоновое оформление;
- Область поля всегда является прозрачной и служит только для корректировки расположения блока на Web-странице. Поэтому в качестве цвета области поля используется цвет, установленный свойством **background** родительского по отношению к текущему HTML-тега.

Универсальным элементом уровня блока является тег DIV, универсальным строковым элементом-тег SPAN. Смысл их использования заключается только в привнесении структуры в HTML-документ. Как теги HTML, они не задают никаких визуальных параметров для своего содержимого. Благодаря использованию тегов DIV и SPAN информация, между их начальным и конечным тегами, считается блоком или строкой соответственно.

В рамках блочной модели представления документов использование этих тегов принимает особое значение. Например, благодаря тегу DIV можно выделить как блок группу строковых элементов или фрагмент текста, что автоматически позволяет применять CSS-свойства, соответствующие блокообразующим элементам. Тоже самое можно было бы сделать с помощью тег **P** (элемента абзаца), но иногда, с точки зрения смысловой организации структуры документа, было бы правильнее использовать универсальный элемент блока DIV. К тому же, использование элемента абзаца, приведет к выделению его содержимого пустыми строками сверху и снизу. То есть тег **P**, как элемент с четко определенным смысловым значением (тег **P** – элемент абзаца) привносит свойственные ему особенности визуального представления. Тег DIV всего этого лишен, предоставляя разработчику самому определить его смысловое значение и особенности внешнего вида.

Размеры прямоугольной области экрана, занимаемой стилевым блоком HTML-тега, выделяются как сумма значений следующих свойств:

- горизонтальный размер:
  - левое поле (*margin-left*);
  - левая граница (*border-left*);

- левый отступ (*padding-left*);
- ширина информативной области (атрибут **width** блокообразующего элемента, который также может указываться в таблице стилей);
- правый отступ (*padding-right*);
- правая граница (*border-right*);
- правое поле (*margin-right*).

- вертикальный размер вычисляется аналогично горизонтальному с той разницей, что вместо правых и левых (*left* и *right*) суммируются верхние и нижние (*top* и *bottom*) размеры соответствующих областей.

Наглядно это представлено на рисунке.

Рис. 4.1. Схема стилевого блока, с указанием CSS-свойств, задающих размеры соответствующих областей

Надо отметить, что если два блока прилегают друг к другу, то размеры верхнего и нижнего полей указывают минимальное расстояние между границами соседних блоков. Поэтому, если для двух прилегающих блоков задано ненулевое значение вертикальных полей, то они сливаются в одно с размером, равным наибольшей среди двух блоков высоте поля. Хороший пример такого слияния можно наблюдать при отображении тегов списка LI. Вер-

тикальные поля "плавающих" элементов (для списка которых задано свойство **float**) никогда не перекрываются.

Далее, по тексту, будет произведено подробное описание свойств различных областей стилевого блока.

4.2. Как задать свойства полей?

Свойства полей задают геометрические размеры области поля стилевого блока.

Возможности CSS3позволяют задать размер каждого поля отдельно, или для всех сразу.

Следующие свойства задают размеры одного отдельного поля:

- *margin-left*: размер левого поля;
- *margin-right*: размер правого поля;
- *margin-top*: размер верхнего поля;
- *margin-bottom*: размер нижнего поля.

Размеры указываются в стандартных значениях длины, определенных для языка CSS3, или в процентах ширины ближайшего блокового элемента-родителя.

Пример:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Команда "Победители во всём!" </title>
      <style type="text/css">
      p{margin-left:100px;
         margin-right:100px;
         margin-top:50px}
      </style>
   </head>
   <body>
      <h1> Информация о команде "Победители во всём!" </h1>
      <h2> Иван Смирнов <br> Капитан команды </h2>
   <p> Выпускник по специальности "Инноватика". Обладает
5-летним практическим опытом руководства техническими
проектами.</p>
```

```
    <h2> Пётр Демидов </h2>
  <p> Окончил МГТУ им. Баумана по специальности
"Робототехника". Имеет огромный опыт конструирования
роботов.</p>
    <h2> Елизавета Казакова </h2>
  <p> Окончила Санкт-Петербургский политехнический университет
по направлению "Графический дизайн". После знакомства с Петром
стала активно интересоваться промышленным дизайном.</p>
    <h2> Алексей Сидоров </h2>
  <p> Отвечает за программирование роботов. </p>
  </body>
</html>
```

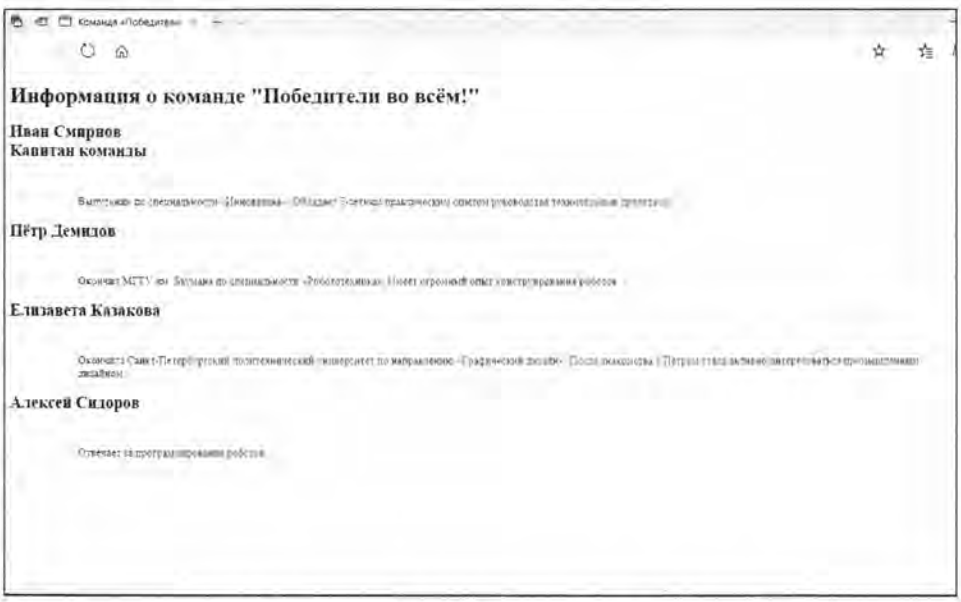


Рис. 4.2. Пример с использованием свойств полей

Как видно из примера, необязательно указывать размеры всех четырех полей стилевого блока. При этом для всех неопределенных полей будет использоваться значение, используемое по умолчанию, т.е. нулевой размер поля.

Для задания размеров сразу всех полей может быть использована сокращенная (стенографическая) форма их указания в виде свойства **margin**. При этом через пробел указываются размеры всех четырех полей. Например,

```
p {margin: 12px 10px 0px 16px}
```

Обратите внимание на порядок, в котором задаются и интерпретируются браузером размеры полей: сначала указывается размер верхнего поля, затем размер правого поля, затем размер нижнего поля и, наконец, размер левого поля.

4.3. Как задать свойства отступов?

Геометрические размеры отступов, также как и в случае с полями, можно задать либо каждый отдельным правилом, либо все сразу.

Ниже перечисленные свойства, задающие размеры каждого поля в индивидуальном порядке:

- *padding-left*: размер левого отступа;
- *padding-right*: размер правого отступа;
- *padding-top*: размер верхнего отступа;
- *padding-bottom*: размер нижнего отступа.

Размеры также указываются в стандартных для CSS3 значениях длины или в процентном задании от ширины ближайшего родительского элемента.

Сокращенная форма задания размеров всех четырех отступов осуществляется с помощью правила **padding**.

Например:

```
pre {background: lightgrey;
          font-color: blue;
          padding: 10px 15px 10px 15px}
```

Размеры отступов задаются и интерпретируются браузером в следующем порядке: сначала указывается размер верхнего отступа, затем размер правого отступа, затем размер нижнего отступа и, наконец, размер левого отступа.

4.4. Как задать свойства границ?

Границы представляют собой обыкновенные линии, ограничивающие видимую часть стилевого блока (поля являются прозрачными, и поэтому невидимы). Определенные в CSS3 свойства позволяют задавать толщину, цвет и тип этих линий.

Ширина границ

Ширина границ задается либо для каждой из них индивидуально, либо для всех четырех сразу.

Ширина отдельных границ задается следующими свойствами:

- *border-top-width*: ширина верхней границы;
- *border-bottom-width*: ширина нижней границы;
- *border-right-width*: ширина правой границы;
- *border-left-width*: ширина левой границы.

Ширина всех четырех границ может быть указана с помощью одного свойства *border-width*. Значения этих свойств либо указываются только в стандартных для CSS3 единицах длины, либо в качестве значений используются специально зарезервированные ключевые слова языка CSS3:

- **thin** - тонкая линия границы (2 пикселя);
- **medium** – средняя линия границы (4 пикселя);
- **thick** - толстая линия границы (6 пикселей).

Пример задания границ:

```
DIV {padding: 10 10 10 10;
border-top-width: 5px;
border-bottom-width: 5px;
border-right-width: thin;
border-left-width: thin}
```

Или тоже самое, но с использованием стенографического свойства border-width:

```
DIV {padding: 10px 10px 10px 10px;
        border-width: 5px thin 5px thin}
```

Порядок задания ширин границ для свойства *border-width* следующий: ширина верхней границы, ширина правой границы, ширина нижней границы, ширина левой границы, все значения, как и в предыдущих случаях, разделяются пробелами.

Пример использования:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Команда "Победители во всём!" </title>
    <style type="text/css">
      div{margin:20;
        padding:20;
        text-align:center;
        border-style:solid;
        border-color:black;
        border-width:thin}
      div.a{border-width:medium}
      div.b{border-width:thick}
      div.c{border-width:15px}
    </style>
  </head>
  <body text=#78bed0 bgcolor=#112232>
    <h1> Информация о команде "Победители во всём!" </h1>
    <div>
      Иван Смирнов <br> Капитан команды <p> Выпускник по
специальности "Инноватика". Обладает 5-летним практическим
опытом руководства техническими проектами.</p>
    </div>
    <div class="a">
```

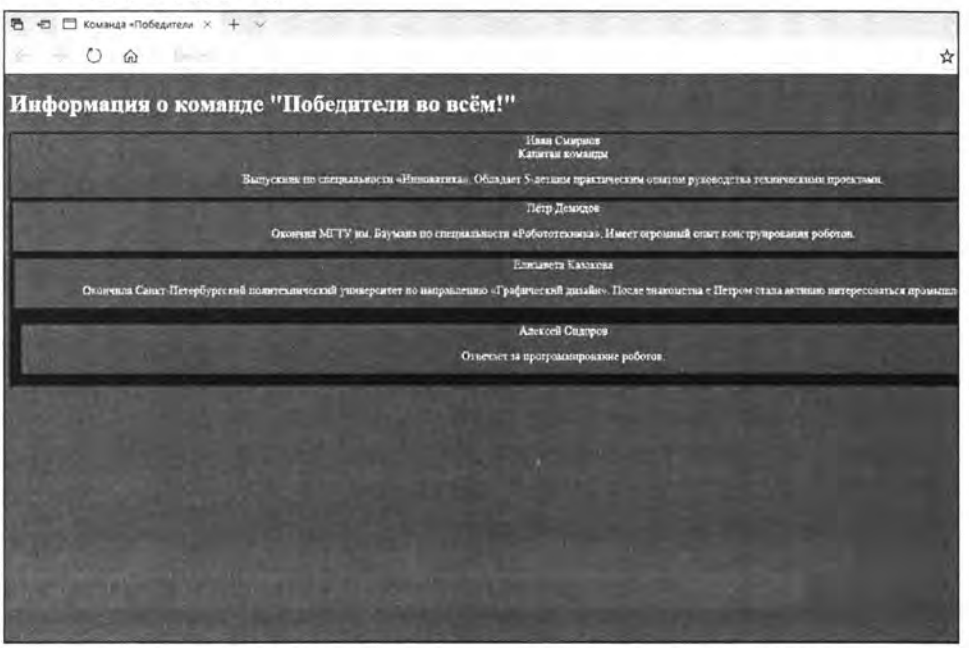


Рис. 4.3. Вид рамок с различными значениями border-width

```
        Петр Демидов <p> Окончил МГТУ им. Баумана
по специальности "Робототехника". Имеет огромный опыт
конструирования роботов.</p>
    </div>
    <div class="b">
      Елизавета Казакова <p> Окончила Санкт-Петербургский
политехнический университет по направлению "Графический
дизайн". После знакомства с Петром стала активно
интересоваться промышленным дизайном.</p>
    </div>
    <div class="c">
      Алексей Сидоров <p> Отвечает за программирование
роботов. </p>
    </div>
  </body>
</html>
```

Для того, чтобы отобразить границы со скругленными углами, используется свойство *border-radius*.

Например:

```
div {border: 1px solid #333;
     padding: 10px;
       border-radius: 0;
     }
```

Цвет границ

Цвет границ определяется следующими свойствами:

- *border-top-color*: цвет верхней границы
- *border-bottom-color*: цвет нижней границы
- *border-right-color*: цвет правой границы
- *border-left-color*: цвет левой границы

Цвет для всех четырех границ сразу можно указать с помощью свойства **border-color**. Порядок задания цветов отдельных границ при этом аналогичен порядку задания ширин для свойства *border-width*. В том случае, если цвет границ не указан, они отображаются цветом, установленным для свойства **color** блокообразующего элемента.

Пример задания стиля абзаца с указанием цвета границ:

```
p {padding: 10px 10px 10px 10px;
     border-width: 5px thin 5px thin;
     border-color: blue}
```

4.5. Как задать тип линии границ?

Для границ должен быть также задан тип линии, которой она отображается. Ниже приведены допустимые в CSS3 названия типов и описан соответствующий им видимый результат:

- **none** - граница определена как отсутствующая. При таком задании значение *border-width* устанавливается равным нулю;
- **solid** - граница будет отображаться в виде одной сплошной линии. Если атрибут *border-width* не задан, то по умолчанию используется значение *border-width: medium*;
- **double** - граница будет очерчена двумя сплошными линиями. Причем свойство *border-width* будет показывать суммарную толщину двух этих линий и расстояния между ними;
- **dotted** - граница будет отображена в виде пунктирной линии;
- **groove** - граница будет показана как вдавленная линия;
- **ridge** - граница будет отображена в виде выпуклой линии;
- **inset** - при задании такого типа границ, весь блок будет отображен как вдавленный;
- **outset** - указание этого значения при задании типа границ приведет к тому, что весь блок будет отображен как выпуклый.

Тип границы блока может быть задан с помощью определения следующих свойств:

- *border-top-style*: задает тип верхней границы;
- *border-bottom-style*: задает тип нижней границы;
- *border-left-style*: задает тип левой границы;
- *border-right-style*: задает тип правой границы

Или с помощью одного стенографического свойства **border-style** могут быть указаны типы всех четырех границ сразу

Пример задания:

```
pre (border-style: double)
div (border-top-style: dotted:
border-bottom-style: solid;
border-left-style: none;
border-right-style: none)
```

Порядок задания типов линий границ для свойства **border-style** идентичен порядку, определённому для всех предыдущих аналогичных свойств: верхняя граница, правая, нижняя и левая.

Пример использования:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Команда "Победители во всём!" </title>
    <style type="text/css">
      div{margin:20;
        padding:20;
        text-align:center;
        border-style:solid;
        border-color:black;
        border-width:medium}
    </style>
  </head>
  <bodytext=#78bed0 bgcolor=#112232>
    <h1> Информация о команде "Победители во всём!" с
примерами рамок разного стиля</h1>
    <div>
      Иван Смирнов <br> Капитан команды <p> Выпускник по
специальности "Инноватика". Обладает 5-летним практическим
опытом руководства техническими проектами.</p>
    </div>
    <div style="border-style:double">
        Пётр Демидов <p> Окончил МГТУ им. Баумана
по специальности "Робототехника". Имеет огромный опыт
конструирования роботов.</p>
    </div>
    <div style="border-width:30px; border-style:groove">
      Елизавета Казакова <p> Окончила Санкт-Петербургский
политехнический университет по направлению "Графический
```

```
дизайн". После знакомства с Петром стала активно
интересоваться промышленным дизайном.</p>
     </div>
     <div style="border-width:50px; border-
style:outset">
       Алексей Сидоров <p> Отвечает за программирование
роботов. </p>
     </div>
   </body>
</html>
```

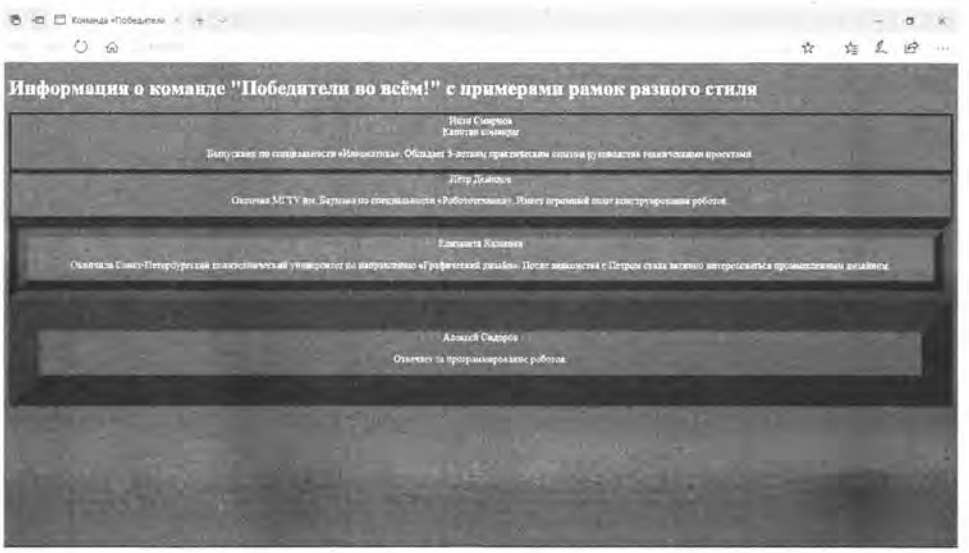


Рис. 4.4. Демонстрация границ разного стиля

Возможности языка CSS3 позволяют, кроме всего прочего, использовать другие формы описания внешнего вида границ. Это обусловлено тем, что для каждой границы, в общем случае, указывается три визуальных параметра: цвет, толщина и тип линии. Поэтому в язык CSS3 включены свойства, определяющие их вместе для каждой из границ:

- *border-top*: - указывает ширину верхней границы, тип линии и ее цвет;
- *border-bottom*: *border-left*: *border-right*: - указывают вышеперечисленные характеристики для нижней, левой и правой границ соответственно.

Самым обобщенным свойством границ является CSS-свойство **border**. Оно задает ширину, тип линии и цвет сразу для всех границ. Причем, для каж-

дого параметра указывается одно значение, которое применятся ко всем четырем границам.

Пример использования:

```
h3 {border-bottom: 10px solid blue}
```

Благодаря этому правилу, все заголовки третьего уровня будут подчеркнуты синей одинарной линией. Следует помнить о том, что если для свойства, например, **border-top**, указаны не все три визуальные характеристики границы, то для недостающего параметра будет использоваться значение, установленное для блокообразующего HTML-тега.

Например,

```
pre {border-color: green;
         border-top: double;
         border-bottom: double}
```

В этом случае задан зеленый цвет для всех границ. Однако, верхняя и нижняя границы будут показаны черным цветом (цветом, используемым по умолчанию для вывода текста). Это произойдет ввиду того, что свойства *border-top* и *border-bottom* по своему определению помимо типа линии границы должны определять ее цвет и толщину. Т.к. цвет не указан, эти свойства используют цвет блокообразующего тега PRE. А поскольку они по тексту заданы после свойства **border-color**, то они переопределяют его значение. Чтобы этого не происходило, достаточно следующим образом поменять свойства местами:

```
pre {border-top: double;
   border-bottom: double;
            border-color: green}
```

В этом случае верхняя и нижняя границы будут отображены зеленой двойной линией. Правая и левая границы не будут показаны вовсе.

4.6. Как задать цвет текста и фона?

Как уже упоминалось, каскадные таблицы стилей были задуманы для наиболее эффективного задания внешнего вида содержимого HTML-элементов, который они принимают при отображении в окне браузера. А

что же, как не цветовое оформление является основной визуальной характеристикой. Для каждого отображаемого HTML-тега может быть указано два цвета: цвет фона и цвет переднего плана. На переднем плане отображается информативное содержимое тега, проще говоря - текст. В CSS3 цвет текста задается единственным, предназначенным для этих целей свойством **color**. Цвет указывается одним из стандартных, определенных в CSS3 способов: ключевым словом или RGB- функцией.

Например:

```
P {color: olive}
B {color: RGB (255,0,0)}
```

Для описания фонового оформления элемента в CSS3 определено несколько возможностей.

Фон можно задать с помощью следующих свойств:

- **background-color**: указывает цвет фона;
- **background-image**: указывает фоновое изображение. Причем для этого случая может быть задано несколько дополнительных параметров фона:
  - *background-repeat*: указывает наличие или отсутствие дублирования фонового изображения как по вертикали, так и по горизонтали;
  - *background-attachment:* это свойство указывает на то, будет или нет фоновое изображение прокручиваться при прокрутке документа;
  - *background-position*: задает положение изображения в рамках стилевого блока, к которому оно принадлежит;
  - *background-origin* – определяет область позиционирования фонового рисунка;
  - *background-size* – масштабирует фоновое изображение, согласно заданным размерам;
  - *background-clip* – определяет, как цвет фона или фоновая картинка должна выводиться под границами.

Свойство background-color

Использование этого свойства аналогично использованию свойства **color**. Разница заключается в том, что для строковых элементов свойство **color**. определяет цвет текста, а *background-color* – цвет фона, на котором этот

текст отображается. Если свойство *background-color* установлено для блокообразующего элемента, то заданным этим свойством цветом будет залито все информационное пространство стилевого блока вместе с его отступами.

Пример использования:

```
<!DOCTYPE html>
<html>
   <head>
      <title> Команда "Победители во всём!" </title>
      <style type="text/css">
      h1{border-style:double;
         background-color:white}
      h2{margin:15px 200px 15px 100px;
         padding:15px;
         background-color:#f8b74f;
         color:black}
      </style>
   </head>
```

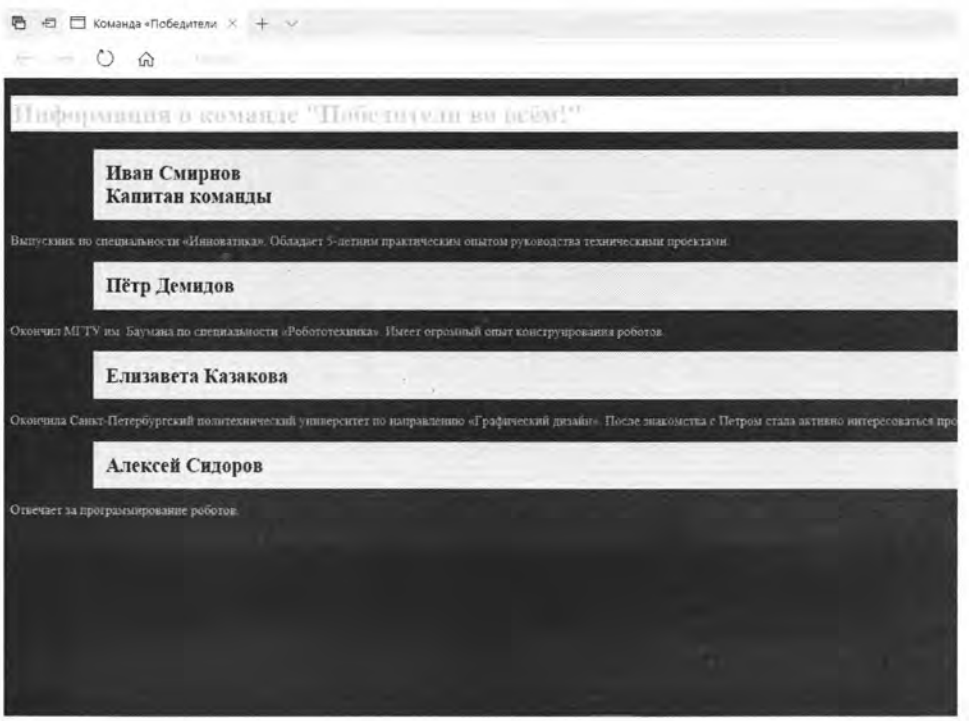


Рис. 4.5. Пример документа, в оформлении которого использовалось свойство background-color

```
  <body text=#78bed0 bgcolor=#112232>
    <h1> Информация о команде "Победители во всём!" </h1>
    <h2> Иван Смирнов <br> Капитан команды </h2>
  <p> Выпускник по специальности "Инноватика". Обладает
5-летним практическим опытом руководства техническими
проектами.</p>
    <h2> Пётр Демидов </h2>
  <p> Окончил МГТУ им. Баумана по специальности
"Робототехника". Имеет огромный опыт конструирования
роботов.</p>
    <h2> Елизавета Казакова </h2>
  <p> Окончила Санкт-Петербургский политехнический университет
по направлению "Графический дизайн". После знакомства с Петром
стала активно интересоваться промышленным дизайном.</p>
    <h2> Алексей Сидоров </h2>
  <p> Отвечает за программирование роботов. </p>
  </body>
</html>
```

Свойство background-image

В качестве значения этого свойства указывается URL- адрес изображения, которое будет использоваться в качестве фонового.

Например,

```
body {background-image: URL (my image\bg image.jpg)}
p {color: yellow;
background-image: URL (my image\aaa32.jpg)}
```

Для каждого блокообразующего HTML-тега изображение будет занимать всю информативную область блока, вместе с его отступами. Если отводимое под изображение пространство стилевого блока по своим размерам меньше самого изображения, то значит, последнее будет отображено частично. Геометрические размеры фоновой картинки никак не влияет на размеры стилевого блока.

Примечательно, что фоновое изображение может быть задано и для строкового HTML-тега. В этом случае отображается только та часть картинки, которая необходима для вывода текста. Её левый верхний угол соответствует первой букве текста строкового элемента.

Например,

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title> Команда "Победители во всём!" </title>
    <style type="text/css">
    h2{background-image:URL(fon.jpg);
      color:black}
    </style>
  </head>
  <body text=#78bed0 bgcolor=#112232>
    <h1> Информация о команде "Победители во всём!" </h1>
    <h2> Иван Смирнов <br> Капитан команды </h2>
    <p> Выпускник по специальности "Инноватика". Обладает
5-летним практическим опытом руководства техническими
проектами.</p>
      <h2> Пётр Демидов </h2>
    <p> Окончил МГТУ им. Баумана по специальности
"Робототехника". Имеет огромный опыт конструирования
роботов.</p>
      <h2> Елизавета Казакова </h2>
    <p> Окончила Санкт-Петербургский политехнический
университет по направлению "Графический дизайн". После
знакомства с Петром стала активно интересоваться промышленным
дизайном.</p>
      <h2> Алексей Сидоров </h2>
    <p> Отвечает за программирование роботов. </p>
  </body>
</html>
```

Свойство background-repeat

Это свойство используется, если задано фоновое изображение. Оно определяет, будет ли фоновое изображение дублироваться, и если будет, то каким образом. По умолчанию фоновая картинка будет в виде мозаики заполнять все пространство стилевого блока, отведенное под его информативную область и отступы.

Свойство *background-repeat* позволяет указать способ дублирования. В соответствии со своим назначением, этот свойство может принимать следующие значения:

- **repeat** - изображение дублируется в вертикальном и горизонтальном направлении;

- **repeat-x** - изображение дублируется только в горизонтальном направлении;
- **repeat-y** - изображение дублируется только в вертикальном направлении;
- **space** – изображение дублируется столько раз, чтобы полностью заполнить область;
- **round** – изображение дублируется столько раз, чтобы в области поместилось целое число рисунков. Если это не удастся, фоновое изображение будет масштабировано;
- **no-repeat** - изображение не дублируется вовсе. При этом оно будет отображено только один единственный раз.

Пример использования:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
    Примеры различного задания свойства background-repeat
    </title>
    <style type="text/css">
    p{background-image:URL(bg001.jpg);
    margin:10px;
    padding:40px;
    border:solid orange medium;
    text-align:center}
    </style>
  </head>
  <body bgcolor=white>
  <h3 align=center> Примеры различного задания свойства
background-repeat </h3>
  <p style="background-repeat:no-repeat">
  Для этого блока<b> background-repeat:no-repeat</b>
  </p>
  <p>
  Свойство background-repeat для этого блока используется в
значении, установленном по умолчанию. То есть<b> background-
repeat:repeat</b>
  </p>
  <p style="background-repeat:repeat-x">
  Для этого блока<b > background-repeat:repeat-x</b>
  </p>
  <p style="background-repeat:repeat-y">
```

```
   Для этого блока<b> background-repeat:repeat-y</b>
   </p>
   </body>
</html>
```

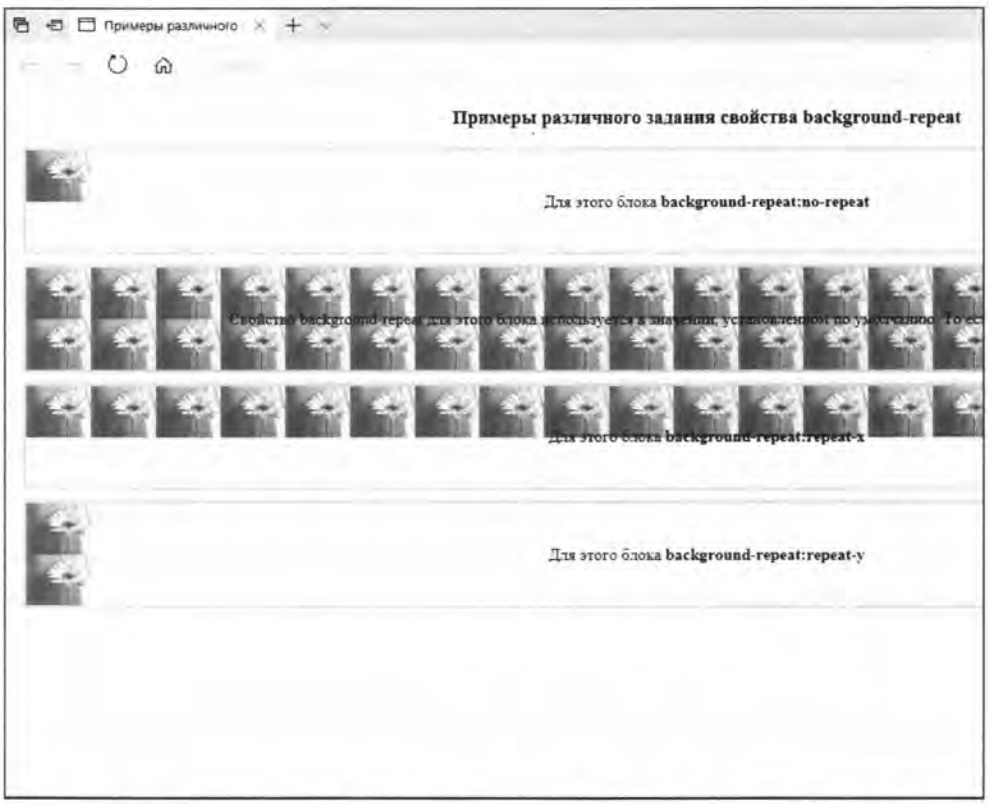


Рис. 4.7. Демонстрация различного задания свойства background-repeat

Свойство background-attachment

Свойство *background-attachment* устанавливает, будет ли фоновое изображение фиксироваться в окне браузера или будет прокручиваться вместе с документом в процессе его прокрутки. Это свойство имеет смысл только для фоновых изображений элементов, для которых предусмотрены полосы прокрутки (элементы BODY и FRAME). Если изображение указано как за-

фиксированное, то содержимое элемента как бы проплывает над ним при прокрутке.

В противном случае фоновое изображение (обычно продублированное) будет прокручиваться вместе с документом.

Для задания параметров фиксации используются следующие значения свойства *background-attachment*:

- **fixed** - фоновое изображение задается как зафиксированное;
- **scroll** - фоновое изображение будет прокручиваться вместе с документом.

По умолчанию используется значение scroll.

Пример задания:

```
BODY {background-image: url (gb706.gif);
background-repeat: no-repeat;
background-position: center;
background-attachment: fixed}
```

В этом примере изображение gb706.gif, используемое в качестве фона, будет отображено в единственном экземпляре (никакого дублирования), и расположено по центру и зафиксировано.

Свойство background-position

Это свойство своим значением определяет начальное месторасположение фонового изображения в рамках стилевого блока. Другими словами, оно показывает, где будет отображен первый экземпляр фонового изображения. Именно от него и будет производиться дублирование картинки в разные стороны.

Через свойство *background-position* положение фоновой картинки может быть задано тремя способами:

- В процентах. В этом случае имеет место следующая схема позиционирования:
  - если в качестве значения свойства *background-position* указана пара 0% 0% (*background-position*: 0% 0%), то левый верхний угол изображения помещается в левый верхний угол области отступов;

- если указана пара 30% 60%, то сначала браузером определяется точка изображения, полученная в результате смещения от левого верхнего угла изображения на 30% ширины вправо и на 60% высоты вниз. Затем изображение размещается в стилевом блоке таким образом, чтобы эта точка была совмещена с точкой блока, полученной в результате смещения вдоль области отступов на 30% вправо и на 60% вниз от верхнего левого угла области отступов;
- Если указано значение 100% 100%, то в этом случае правый нижний угол изображения совмещается с правым нижним углом области отступов стилевого блока;

- в единицах длины. Здесь все просто. Пара значений, например, 20px 15px, показывает, что левый верхний угол изображения будет смещен на 20 пикселов вправо относительно левого верхнего угла области отступов, и на 15 пикселов вниз.
- с помощью ключевых слов *top*, *center*, *bottom*, *right* и *left*. В этом случае положение фоновой картинки может быть задано одной из следующих комбинаций:
  - *topleft =topleft* - комбинация аналогична процентному значению 0%0%
  - *bottomright = rightbottom* - комбинация аналогична процентному значению 100% 100%
  - *center =centercenter* - комбинация аналогична процентному значению 50%50%
  - *top =topcenter = centertop* - комбинация аналогична процентному значению 50% 0%
  - *left =leftcenter =centerleft* - комбинация аналогична процентному значению 0% 50%
  - *righttop =topright* - комбинация аналогична процентному значению 100%0%
  - *bottomleft = leftbottom* - комбинация аналогична процентному значению 0% 100%
  - *right =rightcenter =centerright* - комбинация аналогична процентному значению 100% 50%
  - *bottom =bottomcenter =centerbottom* - комбинация аналогична процентному значению 50%100%

Это были перечислены все возможные варианты использования ключевых слов.

Для всех способов задания сначала указывается месторасположение по горизонтали, второе значение характеризует положение по вертикали.

В том случае, если для свойства *background-position* в качестве значения указано только одно процентное значение или одна длина, то это значение определяет положение изображения по горизонтали. Вертикальная составляющая принимается равной 50 %.

Допускается совместное использование в одном определении значений в единицах длины и в процентах.

Например:

```
p {background-image: url (myimage.gif);
      background-position: 10% 1cm;
       color: yellow}
```

Что касается ключевых слов, то они не могут использоваться совместно с процентным заданием или значением в единицах длины.

Как и для границ, отступов и полей, для фона в CSS3 предусмотрено стенографическое (сокращенное) свойство **background**, с помощью которого можно сразу задать все параметры фона и фонового изображения. Свойство **background** объединяет в себе все свойства фона: *background-color*, *background-image*, *background-repeat*, *background-attachment*, *background-position* и др.

Пример использования:

```
PRE {color: blue;
background: url (/image/bg.igf) 0%0% repeat scroll}
```

В том случае, если какой-то из параметров не указан, то он будет применен в значении, используемом браузером по умолчанию. Все свойства, начинающиеся со слова **background**, не наследуются. Однако значением свойства *background-color* по умолчанию является ключевое слово **transparent**. Это означает, что фон является прозрачным, и поэтому будет виден фон родительского элемента, что создает иллюзию наследования.

Функции radial-gradient и linear-gradient

Использование данных функций позволяет добавить радиальный или линейный градиент к фону.

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Команда "Победители во всём!" </title>
    <style type="text/css">
    body{text-align:center;
    background: repeating-linear-gradient(90deg, #fff, #fff
25px, #8397a9 25px, #ffc7aa 50px) fixed;
      text-color:#78bed0}
    </style>
  </head>
  <body>
    <h1> Информация о команде "Победители во всём!" </h1>
    <h2> Иван Смирнов <br> Капитан команды </h2>
    <p> Выпускник по специальности "Инноватика". Обладает
5-летним практическим опытом руководства техническими
проектами.</p>
      <h2> Пётр Демидов </h2>
    <p> Окончил МГТУ им. Баумана по специальности
"Робототехника". Имеет огромный опыт конструирования
роботов.</p>
```

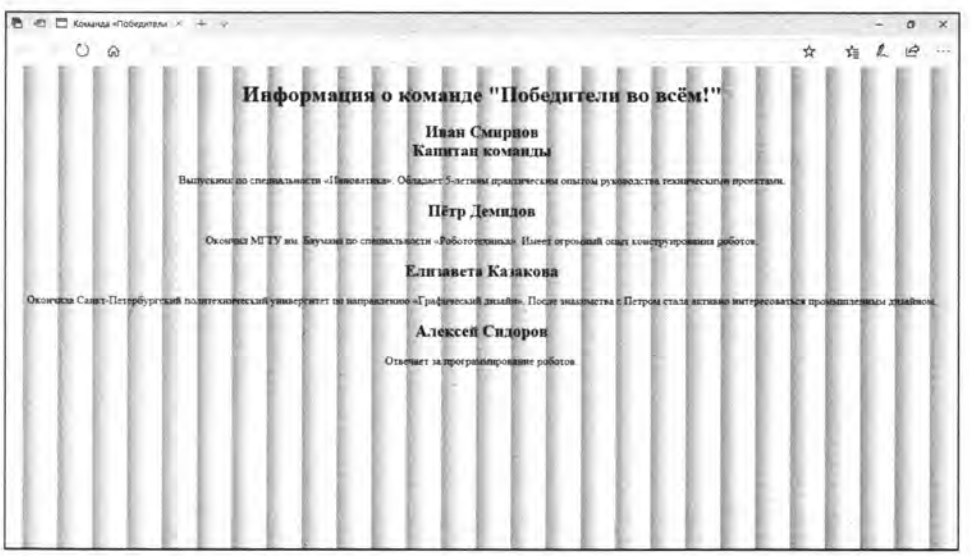


Рис. 4.8. Использование градиента

```
        <h2> Елизавета Казакова </h2>
      <p> Окончила Санкт-Петербургский политехнический
университет по направлению "Графический дизайн". После
знакомства с Петром стала активно интересоваться промышленным
дизайном.</p>
        <h2> Алексей Сидоров </h2>
      <p> Отвечает за программирование роботов. </p>
   </body>
</html>
```

CSS3

Глава 5.

Форматирование текста средствами CSS3

В этой главе вы узнаете:

- *Как задать отступы текста?*
- *Как задать выравнивание текста?*
- *Как визуально оформить текст?*
- *Как установить внутри текстовые интервалы?*
- *Как изменить регистр букв?*
- *Как создать многоколоночный текст?*

Итак, со стилевыми блоками и их цветовыми и фоновыми характеристиками разобрались. Теперь остается описать разметку текста, располагающегося в информативной области блоков. С помощью средств языка CSS3 можно задать отступы и высоту строк в текстовых фрагментах, расстояния между словами и буквами. Возможности CSS3 также позволяют трансформировать текст, написанный строчными буквами в прописной текст, и наоборот. С помощью свойства **text-decoration** можно указать подчеркивание, зачеркивание или надчеркивание текста. Описание всего этого букета возможностей и составляет этот раздел книги.

Общепринятым считается тот факт, что от выбора подходящего шрифта может во многом зависеть привлекательность документа. Каждый из них имеет свой индивидуальный внешний вид. В этом разделе будет произведено описание работы со шрифтами. Подключение определенного шрифта и задание его визуальных характеристик (прямой, курсив, жирный и т.п.) является одними их наиболее часто производимых разработчиком операций. Язык каскадных таблиц стилей CSS3 предлагает для этих целей свои возможности.

5.1. Как задать отступы текста?

С помощью свойства **text-indent** может быть задан отступ первого слова в первой строке абзаца. Или, говоря другими словами, определяет размер красной строки. Сдвиг производится относительно левого края текстового фрагмента (т.е. относительно левого края информативной области стилевого блока).

Размер отступа может быть задан либо в фиксированном виде (в единицах длины), либо в виде процентов. Процентное соотношение задается относительно ширины информативной области (ширины текстового фрагмента).

Пример использования:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
    Демонстрация применения свойства text-indent.
    </title>
    <style type="text/css">
    P {color:black;
```

```
        margin:25px;
        margin-top:5px;
        padding:15px;
        background:lightgrey;
        border:solid gray
    }
    </style>
</head>
<body text=#78bed0 bgcolor=#112232>
    <h3 align=center>
        Демонстрация применения свойства text-indent </h3>
    Обычный абзац. По умолчанию <b> text-indent:0 </b>
    <p>
    текст текст текст текст текст текст текст
    текст текст текст текст текст текст текст
```

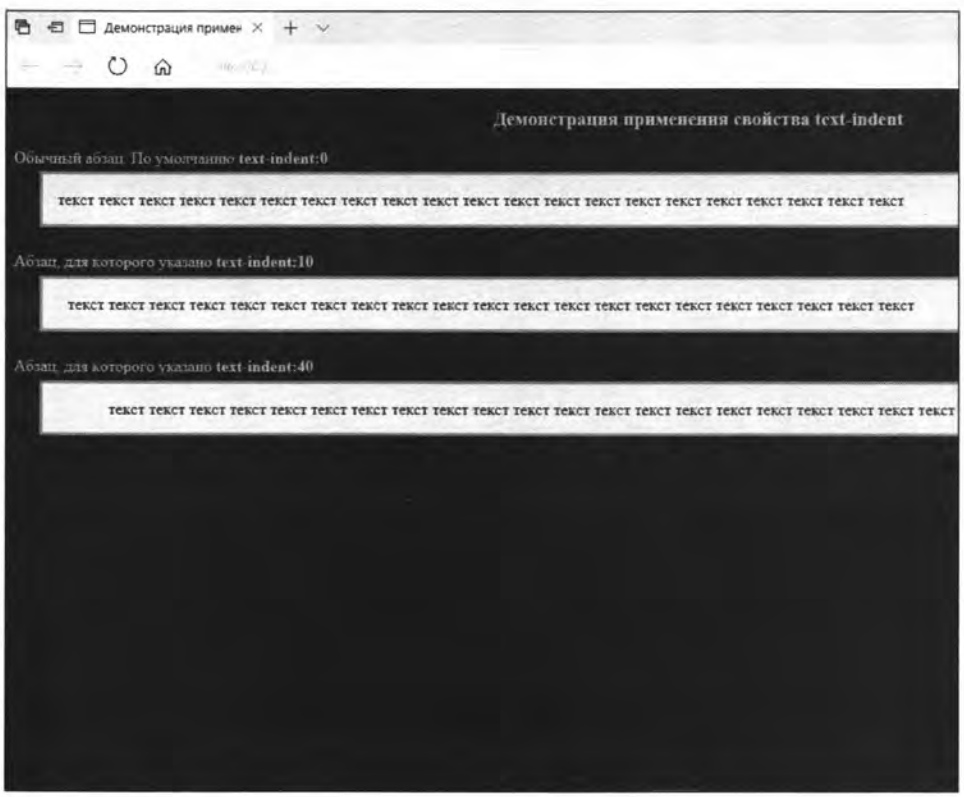


Рис. 5.1. Документ, в котором показано применение свойства text-indent

```
      текст текст текст текст текст текст текст
      </p>
      Абзац, для которого указано <b> text-indent:10 </b>
      <p style="text-indent:10px">
      текст текст текст текст текст текст текст
      текст текст текст текст текст текст текст
      текст текст текст текст текст текст текст
      </p>
      Абзац, для которого указано <b> text-indent:40 </b>
      <p style="text-indent:50px">
      текст текст текст текст текст текст текст
      текст текст текст текст текст текст текст
      текст текст текст текст текст текст текст
      </p>
   </body>
</html>
```

5.2. Как задать выравнивание текста?

Свойством **text-align** определяется выравнивание текста внутри информативной области стилевого блока. С помощью него текст может быть выровнен:

- *left* – по левому краю;
- *right* – по правому краю;
- *center* – по центру;
- *justify* – по ширине;
- *start* – аналогично значению *left*, если текст идет слева направо, и *right*, если текст идет справа налево;
- *end* – алогично значению *right*, если текст идет слева направо, и *left*, если текст идет справа налево.

Пример задания:

```
P {color: yellow;
   text-indent: 4 em;
   text-align: justify}
```

5.3. Как визуально оформить текст?

Как уже упоминалось выше, с помощью свойства **text-decoration** можно придать тексту такие визуальные характеристики как: подчеркивание, зачеркивание и надчеркивание. Для этого достаточно использовать свойство **text-decoration** с одним из следующих значений:

- *underline* – каждая строка текста будет подчеркнута;
- *overline* – все строки текста будут отображены с чертой, расположенной над каждой их них;
- *line-through* – каждая строка будет отображена перечеркнутой;
- *nont* – никакого декоративного оформления текста производиться не будет. Это значение выставлено по умолчанию.

Пример использования:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Команда "Победители во всём!" </title>
    <style type="text/css">
    body {text-align:center;
      background-color:#112232;
      color:#78bed0}
    </style>
  </head>
  <body>
    <h1> Информация о команде "Победители во всём!" </h1>
    <h2> Иван Смирнов <br> Капитан команды </h2>
    <p style="text-decoration:underline"> Выпускник по
специальности "Инноватика". Обладает 5-летним практическим
опытом руководства техническими проектами.</p>
      <h2> Пётр Демидов </h2>
    <p style="text-decoration:overline"> Окончил МГТУ им.
Баумана по специальности "Робототехника". Имеет огромный опыт
конструирования роботов.</p>
      <h2> Елизавета Казакова </h2>
    <p style="text-decoration:line-through"s> Окончила Санкт-
Петербургский политехнический университет по направлению
"Графический дизайн". После знакомства с Петром стала активно
интересоваться промышленным дизайном.</p>
      <h2> Алексей Сидоров </h2>
    <p> Отвечает за программирование роботов. </p>
  </body>
</html>
```

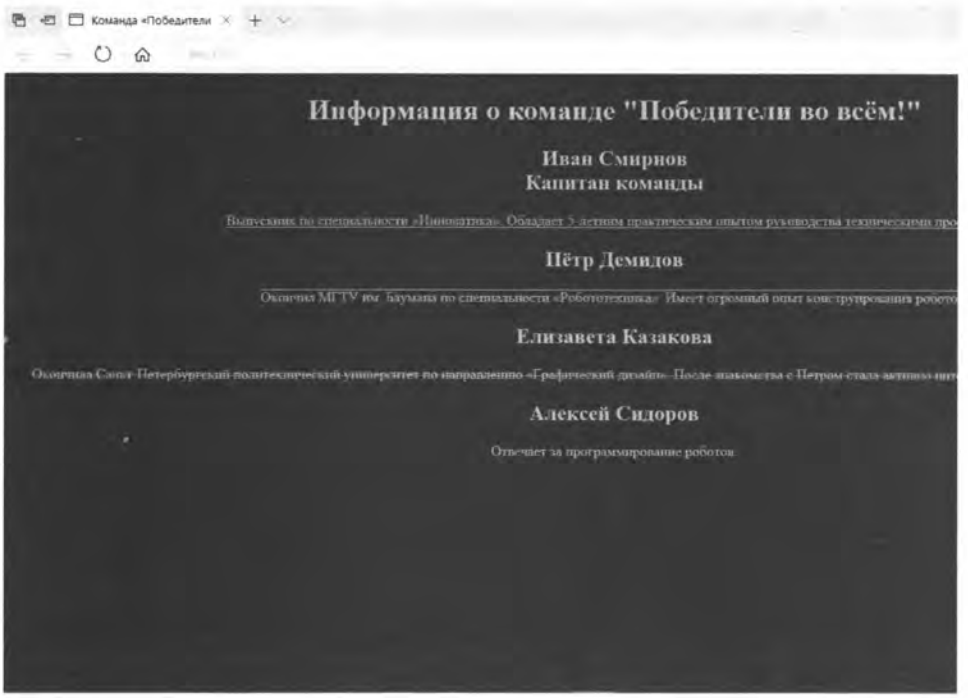


Рис. 5.2. Документ, в котором показано применение свойства text-decoration

5.4. Как установить внутри текстовые интервалы?

Важными визуальными характеристиками текста являются его межбуквенные интервалы и интервалы между словами. Управление размером межбуквенных интервалов осуществляется с помощью CSS-свойства **letter-spacing**, которое устанавливает расстояние между буквами, задаваемое в единицах длины.

Например,

```
H3 {color: blue;
letter-spacing: 13px}
```

Свойство **letter-spacing** может быть выставлено в значении *normal* (*letter-spacing: normal*), что указывает браузеру использовать стандартные интервалы текущего шрифта.

Размер интервалов между словами задается с помощью свойства **word-spacing**.

Размер междустрочных интервалов задается с помощью свойства **line-height**.

5.5. Как изменить регистр букв?

С помощью свойства **text-transform**, определённого в рамках спецификации CSS3, можно осуществлять преобразование регистра букв текста. Или, другими словами, трансформировать строчные буквы в прописные, и наоборот. Эта возможность реализуется свойством **text-transform**, которое в соответствии со своим назначением может принимать следующие значения:

- *capitalize* - первая буква каждого слова отображается прописной (заглавной);
- *uppercase* - все буквы каждого слова отображаются прописными (заглавными);
- *lowercase* - все буквы каждого слова отображаются как строчные;
- *no* - никакого преобразования регистра не производится.

Пример использования:

```
h1 {color: green;
letter-spacing: 5px;
   text-transform: uppercase}
```

5.6. Как создать многоколоночный текст?

Для создания многоколоночного текста используется универсальное свойство COLUMNS.

Дополнительные свойства:

- **column-fill** со значением *auto* позволяет заполнять колонки последовательно, а со значение *balance* – равномерно;
- **column-gap** – определяет расстояние между колонками;

- **column-rule** – рисует границу между колонок. Ширина границы настраивается свойством *column-rule-width*, стиль линии – свойством *column-rule-style*, цвет линии – свойством *column-rule-color;*
- **column-width** – определяет ширину колонки.

CSS3

Глава 6.

Форматирование шрифта средствами CSS3

В этой главе вы узнаете:

- *Как подключить шрифт?*
- *Как указать стиль шрифта?*
- *Как указать размер шрифта?*
- *Как изменить жирность текста?*
- *Про универсальное свойство шрифта*
- *Как подключить удаленные шрифты?*

6.1. Как подключить шрифт?

С помощью свойства **font-family** браузеру указывается шрифт, которым он должен отображать текстовую информацию, для которой это свойство определено (или наследуется).

Например,

```
p {font-family:Arial}
```

В результате такого задания текст абзацев будет отображаться шрифтом Arial.

Возможности свойства **font-family** таковы, что оно позволяет указывать приоритетный список названий шрифтов, каждый из которых будет пробовать подключаться в соответствии со своим приоритетом.

Например, правило

```
body {font-family: Heretet, Times New Roman, Arial}
```

означает, что сначала браузер будет пробовать использовать шрифт Heretet. Если такой шрифт отсутствует в его распоряжении (его просто нет на компьютере пользователя), то будет произведена попытка использования шрифта Times New Roman и т.д.

К тому же, для свойства **font-family** можно заказывать целое семейство шрифтов, среди которых браузер будет искать шрифт, подходящий для отображения всех символов текста. Возможность такого задания введена в язык CSS3 с целью реализации максимально корректной обработки наи-

худшего варианта, когда браузеру не удается использовать ни одного, из специально установленных разработчиком, шрифтов.

Ниже приведены пять типовых семейств шрифтов, включающие в себя все наиболее распространенные шрифты, установленные практически на всех компьютерах:

- Serif (например, Times New Roman)
- Sans-serif (например,Helvetica)
- Monospace (например,Courier New)
- Curcive (например, Zapf-Chancery)
- Fantasy (например, Western)

Полный список шрифтов, с указанием их принадлежности к различным группам приведен ниже, в разделе "6.6. Как подключить удаленные шрифты?"

6.2. Как указать стиль шрифта?

С помощью свойства **font-style** можно задавать стиль шрифта. Рассмотрим значения, принимаемые свойством **font-style** и соответствующий им эффект:

- normal – шрифт отображается в своем обычном, прямом виде. Это значение используется по умолчанию
- *italic* – шрифт будет отображен курсивом
- *oblique* – шрифт будет отображен в наклонном виде

6.3. Как указать размер шрифта?

Благодаря CSS-свойству **font-size** реализуется возможность задания размера шрифта. В качестве значения свойства **font-size** могут выступать либо ключевые слова, либо единицы длины, характеризующие высоту шрифта (ширина изменяется пропорционально высоте).

Возможно также процентное задание размера шрифта по отношению к размеру шрифта, используемому по умолчанию.

Например, правила B {font-size: 150%} и B{ font-size: 1.5 em} являются абсолютно эквивалентными .

Для свойства **font-size** определена следующая последовательность ключевых слов:

```
xx-small, x-small, small, medium, large, x-large, xx-large
```

Последовательность задана по возрастанию. Это значит, что шрифт размером *small* меньше шрифта размером *medium*.

Однако лучше задавать шрифт в типографских пунктах (pt). Именно в этих единицах измерения задается размер шрифта во всех текстовых редакторах.

Пример:

```
B {font-size: 14 pt}
```

В рамках спецификации CSS3 определено такое свойство, как **font-variant**. Через него может быть задан стиль шрифта, при котором все буквы текста будут выглядеть как прописные (заглавные), но немного меньшего размера, чем обычные прописные буквы.

Свойство **font-variant** может принимать следующие значения:

- *small-caps* - при этом происходит вышеописанные изменения шрифта
- *normal* - при таком значении свойства никаких изменений не происходит. Используется по умолчанию.

6.4. Как изменить жирность текста?

Жирность шрифта, которым отображается текст, может быть задана через свойство **font-weight**. В спецификации CSS3 регламентируется использование 9 градаций жирности для одного шрифта. Однако, это еще не означает, что сам шрифт все эти девять градаций поддерживает. Например, используемый по умолчанию большинством браузеров шрифт Times New Roman поддерживает только четыре градации жирности.

Каждая из этих девяти градаций задается одним числом: 100, 200, 300, ..., 900. Значение 100 соответствует самому бледному варианту шрифта, значение 900 – самому жирному.

Кроме этого, допустимо использование следующих ключевых слов:

- normal – задает нормальную (обычную) жирность шрифта. Это значение используется по умолчанию.
- **bold** - указывает на использование полужирного варианта шрифта.

Ключевое слово normal соответствует числовому значению 400, слово **bold** – значению 700. Как уже писалось выше, для разных шрифтов может существовать разное количество градаций жирности, и разное числовое соответствие для них.

Для шрифта Times New Roman имеет место следующее значение:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
    Демонстрация применения свойства font-weight для щрифта
Georgia
    </title>
    <style>
    body{font-family:Georgia;
    text-align:center;
      background:white}
    </style>
  </head>
  <body>
    <hr size=10 color=lightgrey>
    <H3> Демонстрация применения свойства font-weight </H3>
    <p style= "font-weight:100">
    Строка текста, для которой установлена жирность 100
    </p>
    <p style= "font-weight:200">
    Строка текста, для которой установлена жирность 200
    </p>
    <p style= "font-weight:300">
    Строка текста, для которой установлена жирность 300
    </p>
    <p style= "font-weight:400">
    Строка текста, для которой установлена жирность 400
```

```
    </p>
    <p style= "font-weight:500">
    Строка текста, для которой установлена жирность 500
    </p>
    <p style= "font-weight:600">
    Строка текста, для которой установлена жирность 600
    </p>
    <p style= "font-weight:700">
    Строка текста, для которой установлена жирность 700
    </p>
    <p style= "font-weight:800">
    Строка текста, для которой установлена жирность 800
    </p>
    <p style= "font-weight:900">
    Строка текста, для которой установлена жирность 900
    </p>
    <hr size=10 color=lightgrey>
  </body>
</html>
```

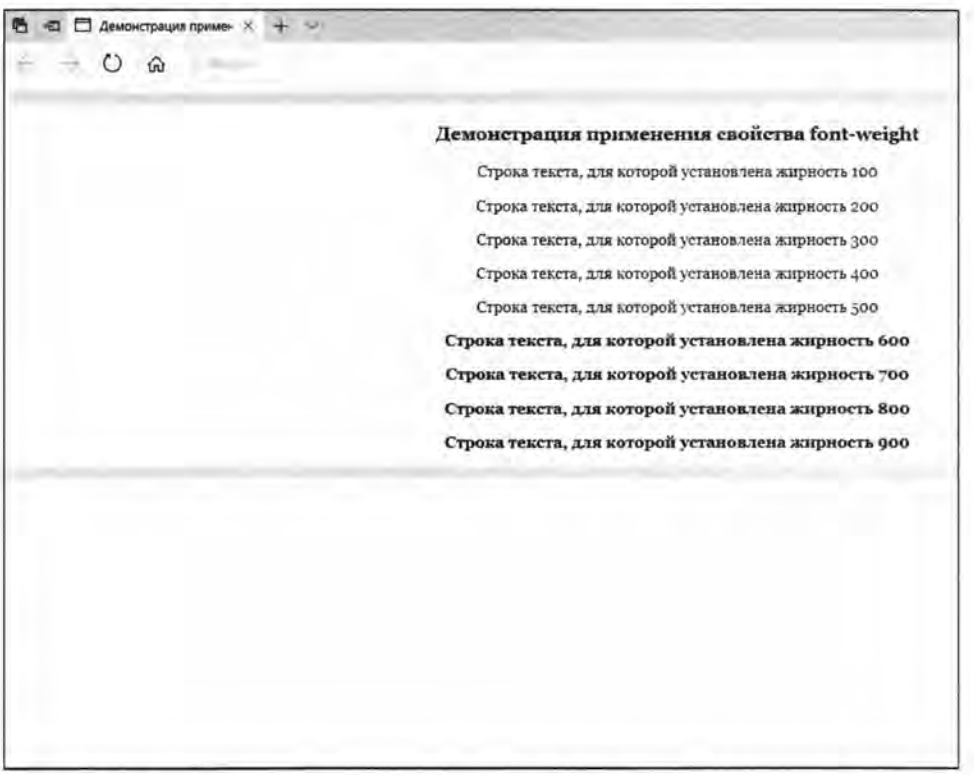


Рис. 6.1. Демонстрация применения свойства font-weight для шрифта Georgia

6.5. Универсальное свойство шрифта

Свойство **font** является стенографическим вариантом всех вышеописанных свойств шрифта. Все значения этих свойств в рамках свойства **font** перечисляются через запятую в следующем порядке: *font-style*, *font-variant*, *font-weight*, *font-size*, *line-height* и *font-family*.

Каждое из свойств может не задаваться. При этом оно будет использоваться в значении, установленном для него по умолчанию. Размер шрифта (свойство **font-size**) и размер межстрокового интервала (свойство **line-height**) задаются через прямой слэш. Названия используемых шрифтов перечисляются через запятую.

Пример задания:

```
body {font: italicbold 12pt/14pt Times New Roman serif}
```

Этим правилом для текстового содержимого тега BODY установлен полужирный курсивный шрифт Times New Roman размером 12 пунктов при высоте строки 14 пунктов. В том случае, если шрифт Times New Roman будет не найден браузером, задано использование любого шрифта из семейства Serif.

Другие примеры:

```
p{font: 900 12pt/14pt serif}
div {font: normal small-caps 120%/1.2em Helvetica}
pre {font: italic 600 12pt;
color: green}
```

6.6. Как подключить удаленные шрифты?

Эй-ти правило @font-face позволяет разработчику задавать свои, авторские шрифты. При этом, если эти шрифты отсутствуют на клиентском компьютере, то они подгружаются с сервера, URL-адрес которого задается одним из параметров **Эй-Ти** правила @font-face.

Пример использования:

```
@font-face {font-family: My Font;
src: url (http://myserver.da.ru/Font/MyFont.eot)}
```

Параметром **font-family** указывается имя шрифта, под которым он будет использоваться для вывода текста. Параметр **src** указывает браузеру URL-адрес, где его можно взять. После такого описания, шрифт MyFont готов к полноправному использованию. Например:

```
<STYLE>
@font-face { font-family: My Font;
   src: url (http://myserver.da.ru/Font/MyFont.eot)}
P {  color: yellow;
   font-family: "MyFont";
   font-size: 12pt}
</STYLE>
```

Ниже приведен примерный список принадлежности шрифтов к различным семействам:

Семейство Serif:

- Латинские шрифты: Times New Roman, Bodoni, Garamond, Minion Web, ITC Stone Serif, MS Georgia, Bitstream Cyberbit
- Греческие шрифты: Bitstream Cyberbit
- Кириллические шрифты: Adobe Minion Cyrillic, Excelcior Cyrillic Upright, Monotype Albion 70, Bitstream Cyberbit, ER Bukinst
- Еврейские шрифты: New Peninim, Raanana, Bitstream Cyberbit
- Японские шрифты: Ryumin Light-KL, Kyokasho ICA, Futo Min A101
- Арабские шрифты: Bitstream Cyberbit

Семейство SANS-Serif:

- Латинские шрифты: MS Trebuchet, ITC Avant Garde Gothic, MS Arial, MS Verdana, Univers, Futura, ITC Stone Sans, Gill Sans, Akzidenz Grotesk, Helvetica
- Греческие шрифты: Attika, Typiko New Era, MS Tahoma, Monotype Gill Sans 571, Helvetica Greek
- Кириллические шрифты Helvetica Cyrillic, ER Univers, Lucida Sans Unicode, Bastion
- Еврейские шрифты: Arial Hebrew, MS Tahoma
- Японские шрифты: Shin Go, Heisei Kaku Gothic W5

- Арабские шрифты: MS Tahoma

СЕМЕЙСТВО CURSIVE:

- Латинские шрифты: Caflisch Script, Adobe Poetica, Sanvito, Ex Ponto, Snell Roundhand, Zapf-Chancery
- Кириллические шрифты: ER Architekt
- Еврейские шрифты: Corsiva
- Арабские шрифты: DecoType Naskh, Monotype Urdu 507

СЕМЕЙСТВО FANTASY:

- Латинские шрифты: Alpha Geometrique, Critter, Cottonwood, FB Reactor, Studz

СЕМЕЙСТВО MONOSPACE:

- Латинские шрифты: Courier, MS Courier New, Prestige, Everson Mono
- Греческие шрифты: MS Courier New, Everson Mono
- Кириллические шрифты: ER Kurier, Everson Mono
- Японские шрифты: Osaka Monospaced

CSS3

Глава 7.

Технологии визуального представления документа. Блоковая структура документа. Понятие о контейнере

В этой главе вы узнаете:

- *Что такое блоковая структура документа?*
- *Что такое нормальный поток?*
- *Что такое позиционирование в CSS3? Свойство position*
- *Что такое абсолютное позиционирование?*
- *Что такое относительное позиционирование?*
- *Что такое перемещаемые блоки? Свойство float*
- *Что такое многослойный вывод? Свойство Z-index*

7.1. Что такое блоковая структура документа?

Как было написано в одном из предыдущих уроков, под блокообразующие HTML-теги при выводе документа отводятся стилевые блоки, обладающие определенными визуальными свойствами (отступы, границы, поля). Под строковые HTML-теги такие блоки не выделяются. Все строковые элементы входят в состав блокообразующих элементов и отображаются в их информативной области. Блокообразующим элементом самого высокого уровня является тег BODY.

С точки зрения структуры использование блоков выглядит не совсем так. В CSS3 каждый HTML-тег, присутствующий в дереве документа (тег BR, например, не присутствует), помещается в структурный блок. Структурные блоки блокообразующих HTML-тегов называются главными. Таким образом, только главные структурные блоки могут иметь свое, индивидуальное стилевое оформление (отступы, границы, поля). Именно они, и их параметры, используются при форматировании (компоновке) документа. Главный блок может содержать либо структурную единицу текста (например, абзац) либо несколько структурных блоков, являющихся его потомками. Например, несколько абзацев. Вообще все структурные блоки, имеющие внутри себя хотя бы один дочерний структурный блок, ничего, кроме структурных блоков, содержать не могут. Рассмотрим это положение на примере:

```
<P>
Текст абзаца
Текст абзаца
<PRE>
   Отформатированный текст абзаца
   Отформатированный текст абзаца
   Отформатированный текст абзаца
</PRE>
</P>
```

На этом примере главный структурный блок содержит в себе некий текст абзаца, а также дочерний структурный блок (тоже главный), порожденный тегом PRE.

В соответствии с вышеприведенным положением CSS3 этого быть не должно. Выход из создавшегося положения заключается в том, что под текст абзаца выделяется безымянный структурный блок. И, таким образом, главный структурный блок, порожденный тегом **P**, становится обладателем двух дочерних структурных блоков и ничего более. Чем и достигается соблюдение требований языка CSS3.

Строковые теги, не образующие новых структурных единиц текста, заключаются в строковые структурные блоки.

Например,

```
<P>
Пример армейского юмора:
<STRONG> Приказ командира</STRONG>
<EM> Всем отсутствующим построиться в отдельную шеренгу </EM>
</P>
```

Тег **P** порождает главный структурный блок, который содержит несколько строковых структурных блоков:

- безымянный строковый блок ("Пример армейского юмора")
- строковый структурный блок, порожденный тегом STRONG ("Приказ командира")
- строковый структурный блок, порожденный тегом EM ("Всем отсутствующим построиться в отдельную шеренгу"

Такая блоковая структура позволяет наиболее полно и просто представлять структуру документа в рамках его блочной модели визуализации. Что касается практики, то на первый взгляд может показаться, что достаточно было

бы главных блоков, обладающих индивидуальным стилевым оформлением (поля, границы, отступы). Однако, помимо декоративного оформления, благодаря блоковой организации всего документа осуществляется задание месторасположения (позиционирование) отдельных фрагментов информации в пределах документа. Это означает, что, например, положение фрагмента текста на Web-странице задается положением структурного блока, которому он принадлежит. Причем, этим фрагментом текста может быть как структурное подразделение текста (абзац), так и отдельная его строка или даже слово.

В CSS3 положение и размеры блоков определяются относительно краев определенной области прямоугольной формы. Эта область называется контейнером. Контейнером для любого структурного блока является пространство, отведенное для информативной области его родительского структурного блока. При этом говорят, что родительский блок назначает контейнер своему потомку.

Стоит отметить, что относительно контейнера задается положение дочернего блока, но размеры последнего никоим образом не ограничиваются размерами контейнера, назначенного родительским блоком. Другими словами, структурный блок может выходить за рамки своего контейнера.

Ввиду всего вышесказанного становится понятным, что начальный контейнер назначается корневым HTML-тегом дерева документа (т.е. либо тегом BODY, либо тегом FRAMESET). Именно начальный контейнер является контейнером самого высокого уровня, его нельзя позиционировать (задавать его положение в контексте других блоков) и перемещать. Это значит, что применительно к нему игнорируются ответственные за эти действия CSS3-свойства **position** и **float**, описываемые далее. Можно задавать только геометрические размеры начального контейнера, указав атрибуты **width** и **height** корневого HTML-тега. Его дочерние блоки всех поколений не могут выходить за рамки начального контейнера.

Теперь рассмотрим, где, как и в какой последовательности могут отображаться структурные блоки в рамках визуального представления документа. Если язык HTML позволял осуществлять только последовательный вывод (содержимое элементов располагается на странице друг за другом в том порядке, в котором заданы сами HTML-элементы), то язык CSS3 позволяет реализовать более широкие возможности позиционирования. Их рассмотрение и будет произведено далее в этой главе.

7.2. Что такое нормальный поток?

В этом разделе будет произведено описание потока или, другими словами, последовательности вывода содержимого HTML-документа, являющейся обычной и используемой по умолчанию.

Именно нормальный поток вывода реализуется обычными средствами языка HTML. В этом случае информация отображается в том же порядке, в котором она задана в HTML-коде документа.

Рассмотрим нормальный поток через призму языка CSS3 и его блочную организацию документа. При нормальном потоке структурные блоки располагаются друг за другом вертикально, начиная с верхнего края своего контейнера, назначенного родительским блоком, в порядке, в котором они указаны в HTML-документе. Вертикальное расстояние между двумя сестринскими блоками определяется размером их полей (свойство **margin**). Причем поля соседних блоков перекрываются и, поэтому, за расстояние между последними принимается размер наибольшего поля. Если по вертикали соприкасаются главный и строковый структурный блоки, то расстояние между ними определятся размером поля главного блока, т.к. строковый блок такового вообще не имеет. Левое поле каждого главного блока или левый край строкового блока соприкасается с левым краем своего контейнера.

Строковый блок выводится одной строкой. Шириной строкового блока является ширина назначенного ему контейнера. Несколько идущих подряд строковых структурных блоков также выводятся одной строкой.

7.3. Что такое позиционирование в CSS3? Свойство position

Средствами языка CSS3 структурный блок может быть вынут из нормального потока отображения документа. В этом случае его месторасположение на Web-странице перестает зависеть от его положения в очередности задания блоков, то есть не зависит от того, в каком месте HTML-кода документа задан элемент. Теперь его визуальное расположение на странице может быть задано независимо от других блоков. Причем он может перекрывать ранее отображенные блоки. Такая схема позиционирования называется абсолютной.

К тому же положение блока может быть задано относительно его обычного положения в нормальном потоке. В этом случае будет иметь место схема относительного позиционирования.

Схема, соответствующая нормальному потоку, называется схемой статического позиционирования. Задание схемы, согласно которой будет позиционироваться блок, осуществляется свойством **position**. В соответствие со своим назначением, оно может принимать следующие значения:

- *static* – указывает на то, что текущий блок является обычным, располагающимся в соответствие с нормальным потоком блоков. Другими словами, задает статическую схему позиционирования для данного блока;
- *relative*–указывает относительную схему позиционирования для данного блока;
- *absolute* – задает абсолютную схему позиционирования для данного блока;
- *fixed* – задает фиксированную схему позиционирования для данного блока;
- *sticky*– задает схему позиционирования, сочетающую в себе относительное и фиксированное позиционирование.

7.4. Что такое абсолютное позиционирование?

Структурный блок, для которого указана абсолютная схема позиционирования, изымается вместе со всеми своими потомками из нормального потока. Его положение задается относительно контейнера, назначенного ближайшим родителем с абсолютной или относительной схемой позиционирования. Если такого родителя не имеется, то положение структурного блока будет задаваться относительно начального контейнера.

Абсолютно позиционированные блоки не влияют на размещение последующих сестринских блоков. Они могут перекрывать ранее отображенные блоки, а также перекрываться блоками нормального потока, задаваемыми после них. Это значит, что при отображении Web-страницы абсолютно позиционируемый блок может быть перекрыт только другим абсолютно- или относительно- позиционируемым блоком, заданным после него.

Надо также отметить, что поля абсолютно позиционируемых главных блоков не перекрываются никакими другими полями.

Положение такого блока задается :

- по вертикали: расстоянием между верхним краем контейнера и верхним краем структурного блока или расстоянием между нижними краями;
- по горизонтали: расстоянием между левым краем контейнера и левым краем структурного блока или расстоянием между правыми краями.

Например,

```
IMG {position : absolute;
   top: 30px;
   left: 100px}
```

Вышеуказанные расстояния задаются с помощью CSS-свойств:

- *top* - своим значением указывает величину смещения верхнего края блока относительно верхнего края контейнера;
- *bottom* - устанавливает расстояние между нижними краями блока и контейнера;
- *left* - задает смещение левого края блока относительно левого края контейнера;
- *right* - задает смещение правого края блока относительно правого края контейнера.

Значения этих свойств задаются в стандартных единицах длины, определенных для языка CSS3. Также может быть использовано ключевое слово **auto**. Использование последнего в качестве значения для свойства *left* приведет к тому, что левый край блока будет совпадать с левым краем контейнера. Если свойство **top** выставлено в значении *auto*, то совпадать будут верхние края блока и контейнера. По умолчанию оба этих атрибута используются в значении *auto*.

В качестве примера приведен код документа, отображенного на рис. 7.1.

```
<html>
   <head>
      <title> Пример использования блоков с абсолютным
позиционированием </title>
      <style type ="text/css">
```

```
    P {
      position:absolute;
      top:60;
      left:100;
      background-color:lightgrey;
      border:double black;
      margin-left: 10 px;
      margin-right: 10 px;
      margin-top: 12 px;
      padding:50; }
    IMG {
      position:absolute;
      top:140;
      left:350
    }
    PRE{background-color:white;
      border:double black;
      padding:20;
      position:absolute;
      top:200;
      left:250
    }
    </style>
  </head>
  <body bgcolor=white>
    <h3 align=center> Пример использования блоков с
абсолютным позиционированием </H3>
    текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст
текст текст текст текст  текст текст текст текст текст текст
текст текст текст текст текст текст текст
    текст текст текст текст текст текст текст текст текст
текст текст
    текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст
текст текст текст текст текст текст текст текст текст текст
текст текст
    <p>
    текст абзаца текст абзаца текст абзаца текст абзаца текст
абзаца текст абзаца текст абзаца текст абзаца текст абзаца
текст абзаца текст абзаца текст абзаца текст абзаца текст
абзаца текст абзаца текст абзаца текст абзаца текст абзаца
```

```
    </p>
    <img src="face1.jpg" width=350>
    <pre>
      отформатированный текст
      отформатированный текст
        отформатированный текст
          отформатированный текст
    </pre>
  </body>
</html>
```

Допускается указание отрицательных значений для этих свойств. В этом случае блок будет выходить за рамки своего контейнера.

Рис. 7.1. Схематичная демонстрация абсолютно позиционируемого блока, выходящего за пределы своего контейнера

Если отрицательные координаты заданы относительно начального контейнера, то это приводит к утрате той части позиционируемого блока, которая выходит за пределы этого контейнера. Например, если в вышеприведенном примере для изображения указать координату top:-140, то часть изображения показана не будет:

Рис. 7.2. Пример документа, в котором для одного из блоков указаны отрицательные координаты относительно начального контейнера

7.5. Что такое относительное позиционирование?

При относительном позиционировании блок сначала размещается на Web-странице в соответствии с нормальным потоком. Затем он может быть смещен относительно своего исходного положения в нормальном потоке. При этом все остальные сестринские структурные блоки располагаются на странице так, как-будто относительно позиционированный блок не был перемещен. Т.е. на месте последнего в нормальном потоке отображается пустое пространство, соответствующее его геометрическим размерам.

HTML-тег порождает относительно позиционированный блок, если для его свойства **position** указано значение *relative*. Относительно позиционируемые блоки, также, как и абсолютно позиционируемые, перекрывают любые блоки нормального потока, независимо от их порядка задания в HTML-коде документа (до или после относительно позиционируемого блока). Блоки, для которых применена относительная схема позиционирования, могут быть перекрыты только такими же блоками (вместе с их потомками), заданными позднее, или абсолютно позиционируемыми блоками (вместе с их потомками), также заданными позднее.

Потомки, относительно позиционируемого блока, смещаются вместе с ним, так как вместе с родительским блоком смещается их контейнер Смещение блока относительно своего нормального положения задается с помощью свойств **top** и **left**, которые характеризуют вертикальную и горизонтальную составляющие смещения соответственно.

Задание значений свойствам **top** и **left** для относительно позиционированного блока аналогично заданию этих свойств для абсолютно позиционируемых блоков. Допускается использование отрицательных значений (в этом случае блок выходит за рамки своего контейнера).

7.6. Что такое перемещаемые блоки? Свойство float

Абсолютно позиционируемый блок располагается поверх блоков нормального потока, а также поверх ранее от позиционированных блоков.

Обычно это воспринимается как их положительное качество. Однако могут быть случаи, когда мало просто расположить определенным образом какой-либо блок на Web-странице, а надо чтобы и окружающие блоки учли его положение и разместились таким образом, чтобы им не перекрываться. Средства языка CSS3 позволяют реализовать некоторые возможности в данном направлении, указав блок как перемещаемый.

Блок является перемещаемым, если для него определено свойство **float**. С помощью этого свойства можно сместить блок либо к левой, либо к правой границе его контейнера. Если, например, блок задан как перемещаемый влево (float: left), то он смещается к левой границе контейнера, а все блоки, которые до этого находились левее его, теперь будут располагаться правее. Другими словами, нормальный поток будет обтекать его справа

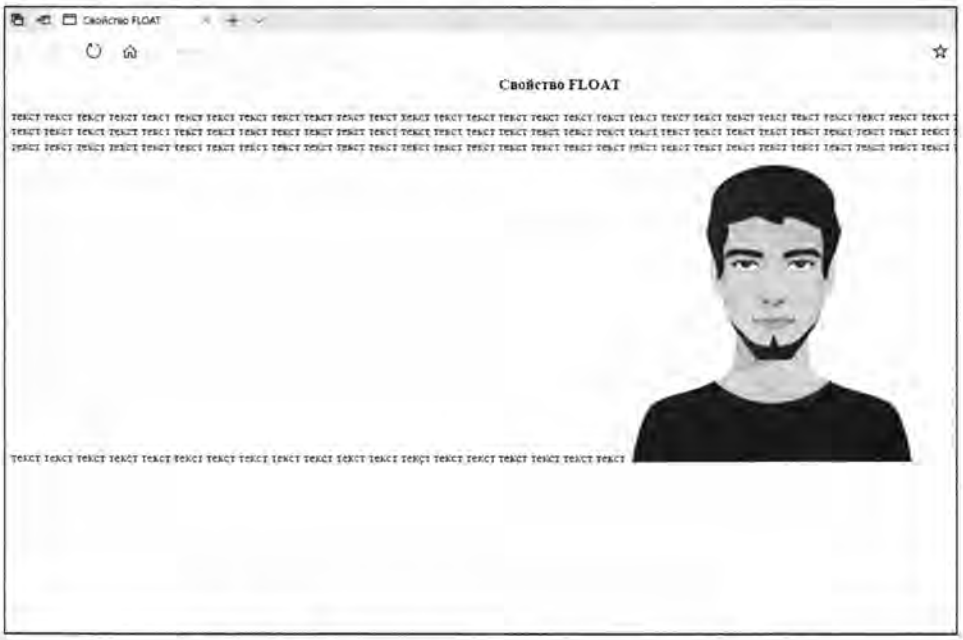


Рис. 7.3. Пример блока, перемещенного влево

Близкой аналогией свойства **float** является прикрепление таблиц и изображения к краям документа с помощью задания атрибута **align**. Благодаря чему достигается обтекание их текстом.

В соответствие со своим назначением, свойство **float** может принимать следующие значения:

- *left* - структурный блок задается как перемещаемый влево;
- *right* - структурный блок задается как перемещаемый вправо;
- *none* - структурный блок никуда не перемещается.

Свойство **float** можно использовать для относительно позиционируемых структурных блоков. При этом в качестве перемещаемого блока будет выступать то исходное положение, относительно которого задается смещение блока, позиционируемого по относительной схеме. Сам этот блок в соответствии со своей схемой, при смещении перестает влиять на положение окружающих сестринских блоков и перекрывает их, как это было описано выше.

Свойство **float**, описанное для абсолютно позиционируемого блока, игнорируется.

7.7. Что такое многослойный вывод? Свойство Z-index

В предыдущих разделах особое внимание было уделено перекрываемости структурных блоков при их отображении. В общем случае для языка CSS3 каждый структурный блок располагается в трехмерном пространстве. При этом, помимо горизонтальных и вертикальных координат, характеризующих положение в плоскости, блоки имеют координату вдоль оси Z, располагаясь один поверх другого. Указание Z-координаты позволяет разработчику явно указывать, какой структурный блок должен располагаться поверх другого. В языке CSS3 эта возможность реализуется с помощью свойства **Z-index**. Значением этого свойства является целое число, показывающее позиционный уровень блока. Структурный блок с более высоким позиционным уровнем перекрывает блок с более низким позиционным уровнем. Допускается использование отрицательных значений. Каждый родительский блок назначает позиционный контекст своим потомкам, на котором они отображаются.

В качестве значения свойству **Z-index** можно указывать ключевое слово *auto*. При этом любой структурный блок с числовым значением **Z-index** всегда будет перекрывать блок, значением свойства **Z-index** которого является ключевое слово *auto*. Для всех элементов, для которых не указано свойство Z-index, по умолчанию используется значение *auto*.

CSS3

Глава 8.

Переполнение и видимость

В этой главе вы узнаете:

- *Что такое переполнение? Свойство overflow*
- *Как управлять видимостью блока?*

С помощью особых свойств языка CSS3 могут быть сработаны некоторые визуальные эффекты, такие как: обработка переполнения и управление видимостью.

8.1. Что такое переполнение? Свойство overflow

Бывают случаи, когда содержимое структурного блока не помещается в его информативной области. При этом возникает ситуация, называемая переполнением. Такое явление может возникнуть, когда:

- явно заданные геометрические размеры информативной области (значения **width** и **height**) не позволяют вместить в нее требуемую информацию в том виде, в каком она задана. Например: для абзаца отведена область 4x5 см, а на нее надо поместить страницу печатного текста;
- структурный блок является абсолютно позиционируемым и выходит за рамки своего контейнера.

В первом случае, по умолчанию, размеры блока будут увеличены до значений, необходимых для полного вывода всей содержащейся в нем информации. Второй случай зависит от браузера. Обычно ничто не мешает абсолютно позиционируемым блокам отображаться вне своих контейнеров, назначенных их родителями. Это верно до тех пор, пока абсолютно пози-

ционируемый блок находится в рамках начального контейнера. Если такой блок выходит за пределы начального контейнера, то он усекается.

Свойством CSS3, отвечающим за описание обработки ситуации переполнения, является свойство **overflow**. Через него задается, что будет происходить с непомещающейся в структурный блок информацией. Свойство **overflow** может принимать следующие значения:

- visible;
- hidden;
- scroll;
- auto.

Visible - это значение говорит о том, что вся информация (и дочерние блоки в том числе), заключенная в структурном блоке должна быть отображена. Причем отображена в пределах структурного блока. Это приводит к увеличению размеров блока до необходимого значения. Это значение используется по умолчанию.

Пример использования:

```
<html>
   <head>
      <title>
      Демонстрация использования CSS-свойства overflow</title>
      <style type ="text/css">
      P {border:double black;
         overflow:visible;
         width:50}
      body {text-align:center}
      </style>
   </head>
   <body>
      <h3> Первоначальная ширина блока равна 50.
      Свойство overflow: visible </h3>
      <P>
      <img src="face1.jpg">
      </P>
   </body>
</html>
```

Рис. 8.1. Результат применения свойства overflow в значении visible

Hidden – это значение указывает на то, что вся информация (и дочерние блоки в том числе), не уместившаяся в структурном блоке заданных размеров, будет усекаться и отображена не будет.

Пример использования:

```
<html>
   <head>
      <title>
      Демонстрация использования CSS-свойства overflow</title>
      <style type ="text/css">
      P {border:double black;
         overflow:hidden;
         width:100}
      body {text-align:center}
      </style>
   </head>
   <body>
      <h3> Первоначальная ширина блока равна 100.
      Свойство overflow: hidden</h3>
```

```
    <P>
    <img src="face1.jpg">
    </P>
  </body>
</html>
```

Рис. 8.2. Результат применения свойства overflow в значении hidden

Scroll - использование этого ключевого слова в качестве значения свойства **overflow**, позволяет установить полосы прокрутки, чем достигается возможность доступа к информации, не изменяя размеры пространства, занимаемого структурным блоком на странице.

Auto - обработка возникшей ситуации переполнения определяется браузером. Им может быть использовано одно из вышеописанных значений в зависимости от контекста ситуации и своих индивидуальных настроек. Обычно при значении auto в случаях переполнения браузер добавляет полосу прокрутки. Причем он может добавить только одну полосу прокрутки, тогда как при значении scroll всегда добавляется обе полосы, даже если одна из них не нужна.

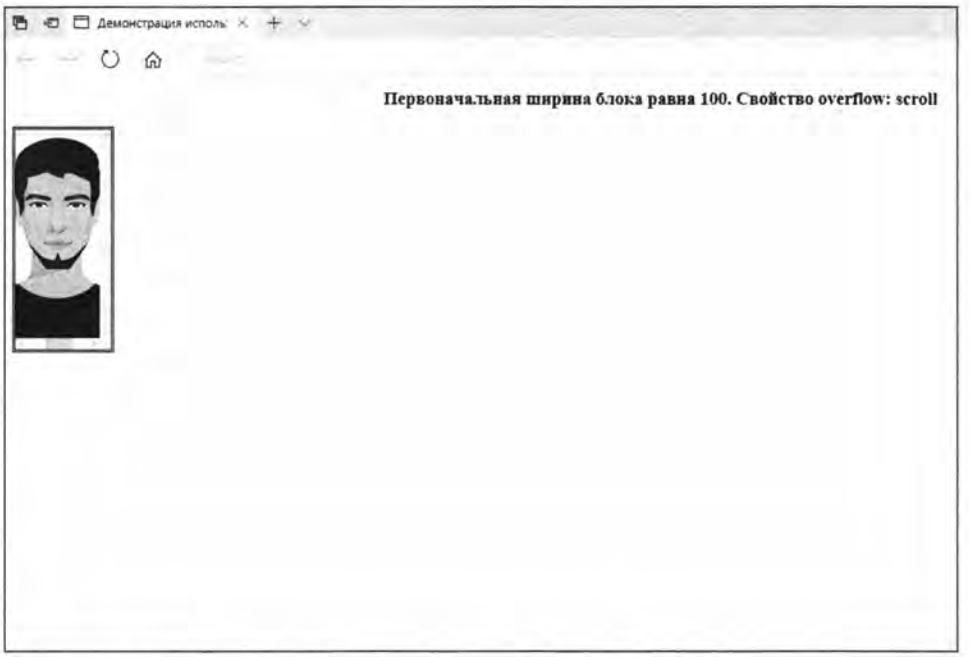


Рис. 8.3. Результат применения свойства overflow в значении scroll

Рис. 8.4. Результат применения свойства overflow в значении scroll

8.2. Как управлять видимостью блока?

С помощью CSS-свойства может быть задана видимость структурного блока, то есть, виден или не виден будет блок пользователю. Таким CSS – свойством является свойство **visibility.** В соответствие со своим назначением оно может принимать следующие значения:

- *visible* - при этом значении структурный блок является видимым;
- *hidden* - при таком значении весь структурный блок становится невидимым и не отображается. Все остальные блоки ведут себя так, как будто его нет.

CSS3

Глава 9.

Отображение списков средствами языка CSS3

В этой главе вы узнаете:

- *Как создать список?*
- *Как задать форму курсора? Свойство cursor*

9.1. Как создать список?

Возможности языка CSS3 позволяют и для списков задавать стилевые настройки. При этом для HTML-тегов UL и OL могут быть определены следующие свойства:

- **list-style-type** –задает тип маркера или нумерацию элементов списка. Этот тип выбирается из стандартных, предусмотренных в CSS3 маркеров;
- **list-style–image** – позволяет использовать изображения, располагающиеся во внешних графических файлах, в качестве маркеров;
- **list-style-position** – указывает положение маркера в списке, т.е. будет он отображен в составе элемента списка, или выдвинутым влево от него.

Подробное рассмотрение этих свойств, собственно, и является предметом данного раздела.

Свойство **list-style-type** определяет тип маркера или нумерацию элементов списка в том случае, если для этих целей не используются изображения (list-style-image: none), или они, по каким-либо причинам, недоступны (например, неправильно указан их URL-адрес). В качестве значений для свойства **list-style-type** могут применяться следующие ключевые слова:

- *none* - вообще никакой маркер не отображается;

- disk - в качестве маркера используется закрашенный кружок;
- *circle* - в качестве маркера используется незакрашенный кружок (окружность);
- *square* - маркер имеет вид закрашенного квадратика;
- *decima* - задает нумерацию списка арабскими цифрами;
- *lower-roman* - задает нумерацию списка строчными римскими цифрами;
- *upper-roman* - нумерация будет осуществляться прописными римскими цифрами;
- *lower-alpha* - нумерация будет осуществляться латинскими строчными буквами;
- *upper-alpha* - для нумерации будут использоваться прописные латинские буквы.

Подробное описание самих списков, а также маркеров и видов нумераций, было произведено ранее, в главе, посвященной HTML.

Пример использования:

```
UL { color: green;
list-style-type: disk
}
```

Благодаря свойству **list-style-image** в качестве маркера для неупорядоченных списков (HTML-тег UL) может использоваться графическое изображение. Значением этого свойства может быть либо ключевое слово **none** (и при этом никакое изображение не используется, а применяется один из стандартных маркеров), либо URL-адрес файла, в котором изображение хранится.

Пример задания:

```
UL {list-style-image: url(http://techrussia.org/assets/img/
Robots/VR.png)}
```

В свою очередь, свойство **list-style-position** задает положение маркера относительно элемента списка. При этом возможны два варианта:

- *inside* - маркер как бы встраивается в элемент списка и является его первым символом;

- *outside* - маркер отображается несколько смещенным влево относительно элемента списка.

Наглядно разницу между ними можно увидеть на следующем примере:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title> Демонстрация использования CSS-свойства
неупорядоченных списков list-style-position </title>
    <style type ="text/css">
      UL { padding:30;
      border: double gray;
      font-family:"Arial";
      background-color:white}
      BODY {text-align:center;
      background-color:lightgrey}
      TH { font-size:14pt}
    </style>
  </head>
  <body>
  <table>
    <TR>
    <TH> Список с inside-маркером
    <TH> Список с outside-маркером
    <TR>
    <TD>
      <UL style="list-style-position:inside ">
    <h5 align=center> Направления Всероссийского
технологического марафона TechRussia: </h5>
    <li> Виртуальная и дополненная реальность </li>
    <li> Интернет-вещей </li>
    <li> Беспилотные летательные аппараты </li>
    <li> Космические технологии </li>
    <li> Робототехника </li>
    <li> Автоматизированные системы управления </li>
    <li> Промышленный дизайн </li>
    <li> Финансовые технологии </li>
    <li> Нейротехнологии </li>
    <TD>
      <UL style="list-style-position:outside ">
    <h5 align=center> Направления Всероссийского
технологического марафона TechRussia: </h5>
    <li> Виртуальная и дополненная реальность </li>
    <li> Интернет-вещей </li>
```

```
      <li> Беспилотные летательные аппараты </li>
      <li> Космические технологии </li>
      <li> Робототехника </li>
      <li> Автоматизированные системы управления </li>
      <li> Промышленный дизайн </li>
      <li> Финансовые технологии </li>
      <li> Нейротехнологии </li>
   </table>
   </body>
</html>
```

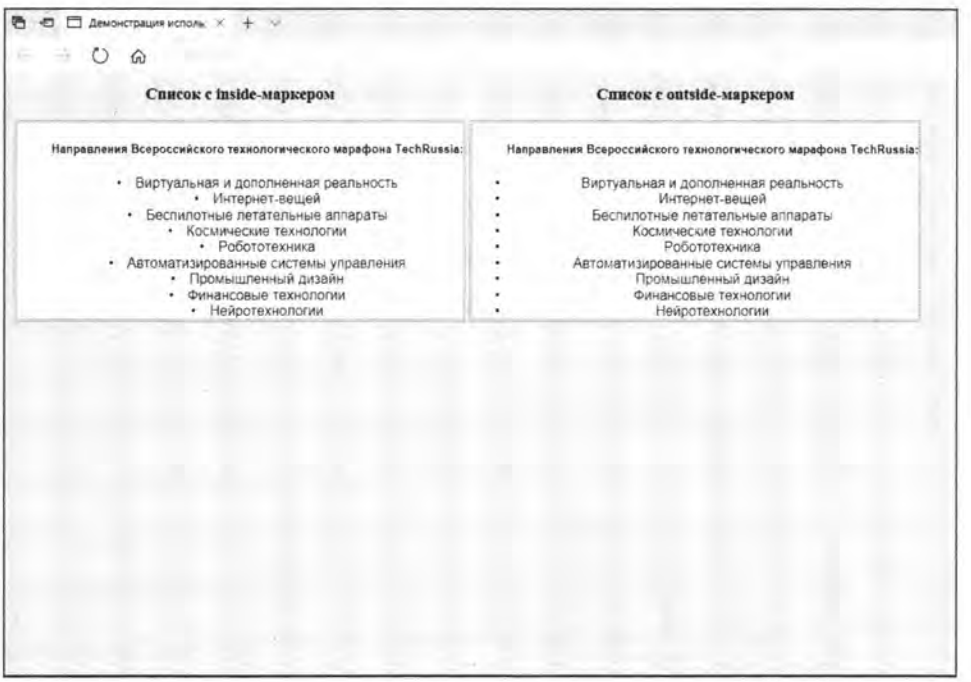


Рис. 9.1. Демонстрация использования свойства list-style-position

9.2. Как задать форму курсора? Свойство cursor

CSS-свойство cursor позволяет разработчику задавать для курсора свой авторский внешний вид. В качестве значения этого свойства указывается URL-адрес файла подключаемого курсора. Причем задаваться может целый список URL-адресов, каждый из которых будет опрашиваться в по-

рядке очередности, если с предыдущего URL-адреса подгрузить курсор не удастся.

Пример использования:

```
BODY {cursor: url ("myfirst.cur"), url ("mysecond.cur")}
```

В этом примере сначала будет попытка подключить курсор из файла *myfirst.cur* и если она окажется безуспешной, то будет подключаться второй курсор. Примечательно, что форму курсора можно изменять в области каждого HTML-элемента индивидуально. Например, можно изменять его форму курсора, если он находится под абзацем: P {cursor: url ("coolcur.cur")}

В качестве ключевых слов могут быть использованы следующие:

- *default* - устанавливается курсор, используемый по умолчанию. Обычно в виде стрелки;
- *crosshair* - устанавливается курсор в виде крестика (пересечения двух отрезков);
- *text* - такой вид курсор принимает, когда указывает на текст, который можно выделить;
- *wait* - курсор принимает вид песочных часов;
- *auto* - вид курсора задается браузером автоматически от контекста.

CSS3

Глава 10.

Визуальные функции в CSS3

В этой главе вы узнаете:

- *Как использовать функцию BLUR?*
- *Как использовать функцию OPACITY?*
- *Как используют функцию DROP-SHADOW?*
- *Как использовать функцию GRAYSCALE?*
- *Как использовать функцию INVERT?*

Функции накладывают определенные эффектные характеристики на информационное содержимое блоков при выводе их на экран. Всего имеется восемнадцать эффектов, которые могут быть применены с помощью одноименных функций:

- attr
- blur
- brightness
- calc
- contrast
- drop-shadow
- grayscale
- hue-rotate
- invert
- linear-gradient
- opacity
- perspective
- radial-gradient
- repeating-linear-gradient
- repeating-radial-gradient

- saturate
- sepia
- var

Среди функций выделяются фильтры, которые позволяют изменять вид изображения, применяя к нему различные визуальные эффекты.

Применение фильтров осуществляется в рамках задания таблицы стилей аналогично обычным CSS-правилам.

Селектор {filter: Имя_фильтра (параметр1=значение, параметр2=значение, ... , параметрN=значение) }

Например:

```
<IMG src="vvv.gif" style="Filter: shadow (Direction=225,
color=gray)">
```

Фильтры могут быть применены как к графическим изображениям, так и к текстовым блокам. В случае с изображениями использование фильтра аналогично тому, как это делается в программных продуктах по обработке фотоизображений, например, в Adobe Photoshop.

Внимание! Для того, чтобы фильтр мог быть применен к текстовому блоку необходимо, чтобы для него были явно заданы ширина (width) и высота (height) или указано абсолютное позиционирование.

Параметры могут задаваться в произвольном порядке. Если описание каких-либо параметров отсутствует, то они устанавливаются в значении, используемом по умолчанию.

Итак, рассмотрим подробно каждый из фильтров:

10.1. Как использовать функцию BLUR?

Применение функции BLUR приводит к размытию отображенной информации в определенном направлении и с определенной интенсивностью.

Параметр функции BLUR характеризует интенсивность размывки. Чем больше значения, тем сильнее будет размыто изображение.

Пример задания:

```
<html>
   <head>
      <title> Применение фильтра BLUR </title>
      <style type ="text/css">
      P {  margin:10;
      padding:20;
      width:400;
      font-weight:700;
      border: double black;
      background:white}
      img { FILTER: BLUR(50)}
      </style>
   </head>
   <body>
      .....текст документа.....
      .....текст документа.....
      <P>     .....Т Е К С Т          А Б З А Ц А .....
      </P>
      .....текст документа.....
      <img src="face1.jpg">
      .....текст документа.....
   </body>
</html>
```

Рис. 10.1. Демонстрация наложения функции BLUR на изображения

10.2. Как использовать функцию OPACITY?

С помощь функции OPACITY устанавливается прозрачность объекта. Причем, прозрачность может быть задана одинаковой для всего объекта, а может изменяющейся в виде градиента.

Значение функции OPACITY задает степень прозрачности. Значение устанавливается в процентах. Отрицательное значение не допускается.

Пример использования:

```
<html>
   <head>
      <title> Применение функции OPACITY </title>
      <style type ="text/css">
      P {  margin:10;
      padding:20;
      width:400;
      font-weight:700;
      border: double black;
      background:white}
      img {FILTER: opacity(20%)}
      </style>
   </head>
   <body>
      .....текст документа.....
      .....текст документа.....
      <P>
      .....ТЕКСТ АБЗАЦА .....
      </P>
      .....текст документа.....
      <img src="face1.jpg">
      .....текст документа.....
   </body>
</html>
```

Рис. 10.2. Демонстрация наложения функции OPACITY на изображения

10.3. Как используют функцию DROP-SHADOW?

Результатом применения функции DROP-SHADOW, отображается копия объекта, несколько сдвинутая относительно исходного объекта и помещённая на задний план. Проще говоря, создается тень-копия данного объекта.

Функция DROP-SHADOW имеет следующие параметры:

- *сдвиг по x* – горизонтальное смещение копии относительно исходного объекта;
- *сдвиг по y* – вертикальное смещение копии относительно исходного объекта;

- *размытие* – этот параметр задает радиус размытия тени;
- *цвет* – указывается цвет копии. По умолчанию тень черная.

Пример задания:

```
<html>
   <head>
      <title>Применение функции DROP-SHADOW</title>
      <style type ="text/css">
      P {margin:10;
      padding:20;
      width:400;
      font-weight:700;
      border: double black;
      background:white}
      img {width:400; filter:drop-shadow(120 120 120px yellow)}
      </style>
   </head>
   <body>
      <img src="Drone.png">
   </body>
</html>
```

Рис. 10.3. Демонстрация использования функции DROP-SHADOW

10.4. Как использовать функцию GRAYSCALE?

Наложение функции GRAYSCALE преобразует цветной в объект в черно-белый.

Пример использования:

```
<html>
  <head>
    <title> Применение функции GRAYSCALE </title>
    <style type ="text/css">
    p{margin:10;
      padding:20;
      width:400;
      font-weight:700;
      border: double black;
      background:white}
    img {width:400; filter:grayscale(100%)}
    </style>
  </head>
```

Рис. 10.4. Применение функции GRAYSCALE

```
  <body>
    <img src="Drone.png">
  </body>
</html>
```

10.5. Как использовать функцию INVERT?

Применение этого фильтра инвертирует пикселы объекта, как это производится в графических редакторах. Значение функции выражается в процентах, где 100% - полная инверсия цветов.

Пример использования:

```
<html>
  <head>
    <title> Применение фильтра INVERT </title>
    <style type ="text/css">
    p{margin:10;
      padding:20;
      width:400;
      font-weight:700;
      border: double black;
      background:white}
    img {width:400; filter:invert(100%)}
    </style>
  </head>
  <body>
    <img src="Drone.png">
  </body>
</html>
```

Функции SATURATE, SEPIA, BRIGTHNESS и CONTRAST позволяют управлять насыщенность, яркостью, контрастом цветов и конвертировать изображение в сепию. Их использование аналогично примерам, показанным выше.

Рис. 10.5. Пример использования функции INVERT


HTML
5
CSS
3

Для заметок

Издательство "Наука и Техника"

Книги по компьютерным технологиям, медицине, радиоэлектронике

Уважаемые читатели!

Книги издательства "Наука и Техника" вы можете:

- **заказать в нашем интернет-магазине**

  www.nit.com.ru (более 100 пунктов выдачи на территории РФ)

- **приобрести в магазине издательства по адресу:**

  Санкт-Петербург, пр. Обуховской обороны, д.107 (ежедневно с 10.00 до 18.30), тел. **(812) 412-70-26**

- **приобрести в Москве:**

"Новый книжный" Сеть магазинов	тел. (495) 937-85-81, (499) 177-22-11
ТД "БИБЛИО-ГЛОБУС"	ул. Мясницкая, д. 6/3, стр. 1, ст. М "Лубянка", тел. (495) 781-19-00, 624-46-80
Московский Дом Книги, "ДК на Новом Арбате"	ул.Новый Арбат, 8, ст. М "Арбатская", тел. (495) 789-35-91
Московский Дом Книги, "Дом технической книги"	Ленинский пр., д.40, ст. М "Ленинский пр.", тел. (499) 137-60-19
Московский Дом Книги, "Дом медицинской книги"	Комсомольский пр.,д. 25, ст. М"Фрунзенская" тел. (499) 245-39-27
Дом книги "Молодая гвардия"	ул. Б. Полянка, д. 28, стр. 1, ст. М "Полянка", тел. (499) 238-50-01

- **приобрести в Санкт-Петербурге:**

Санкт-Петербургский Дом Книги	Невский пр. 28, тел. (812) 448-23-57
Буквоед. Сеть магазинов	тел. (812) 601-0-601

- **приобрести в регионах России:**

г. Воронеж, "Амиталь" Сеть магазинов	тел. (473) 224-24-90
г. Екатеринбург, "Дом книги" Сеть магазинов	тел. (343) 289-40-45
г. Нижний Новгород, "Дом книги" Сеть магазинов	тел. (831) 246-22-92
г. Владивосток, "Дом книги" Сеть магазинов	тел. (423) 263-10-54
г. Иркутск, "Продалить" Сеть магазинов	тел. (395) 298-88-82
г. Омск, "Техническая книга" ул. Пушкина, д.101	тел. (381) 230-13-64

Мы рады сотрудничеству с Вами!

Группа подготовки издания:

Зав. редакцией компьютерной литературы: *М. В. Финков*

Редактор: *Е. В. Финков*

Корректор: *А. В. Громова*

ООО "Наука и Техника"

Лицензия №000350 от 23 декабря 1999 года.

192029, г. Санкт-Петербург, пр. Обуховской обороны, д. 107.

Подписано в печать 21.11.2017. Формат 70x100 1/16.

Бумага газетная. Печать офсетная. Объем 22 п. л.

Тираж 1500. Заказ 11275.

Отпечатано с готовых файлов заказчика

в АО "Первая Образцовая типография"

филиал "УЛЬЯНОВСКИЙ ДОМ ПЕЧАТИ"

432980, г. Ульяновск, ул. Гончарова, 14.

Издательство «Наука и Техника»

Эта книга является превосходным учебным пособием по созданию красивых, функциональных сайтов с использованием популярного бесплатного фреймворка (набора инструментов и шаблонов) Bootstrap. Изложение ведется последовательно: от настроек Bootstrap до создания сайтов разной степени сложности. По ходу даются все необходимые пояснения и комментарии.

Эта книга является превосходным учебным пособием для изучения языка программирования PHP. Вы будете изучать PHP постепенно: от написания первой программы до создания полноценных проектов. Вы научитесь создавать интерактивные элементы, динамические сайты с использованием шаблонизаторов и так далее. Значительную часть книги занимают практические минипроекты.

ISBN 978-5-94387-750-6
978-5-94387-7506

Издательство "Наука и Техника"
г. Санкт-Петербург

Для заказа книг:
(812) 412-70-26
e-mail: nitmail@nit.com.ru
www.nit.com.ru