

## тема 1.2

- типы адресов сети интернет
  - структура назначение классы ip
  - использование масок при адресации
  - Стек протоколов TCP/IP
- Система доменных имён DNS.
  - Ключевые характеристики DNS
  - Домены верхнего уровня.
  - Принципы функционирования доменной системы DNS.
  - Программная поддержка системы DNS.
- Система универсальных идентификаторов ресурсов URI и URL.
  - Принципы системы: расширяемость, полнота, читаемость.
  - Формат URL.
- Структура стеков TCP/IP.
  - Стек TCP/IP и модель OSI.
  - Инкапсуляция протоколов верхнего уровня в протоколы стека TCP/IP.
  - Структура пакетных TCP.

## тема 1.3 Сервисы глобальной сети интернет

- типы сервисов глобальной сети
  - программы клиенты
  - браузеры
- веб сервис
  - архитектура www
  - основные компоненты www(html url http cgi )
  - принципы работы клиент - сервис
  - назначение web технологий и роль www
    - технологии поддерживаемые сервером
    - технологии поддерживаемые клиентом

## 1.4

- Понятие гипертекста.
- Организационная структура информации Web-узла (иерархическая, линейная, в виде паутины).
- Модели проектирования Web-узла.

### Спиральная модель

- Microsoft Solutions Framework (MSF)
- Модель Уолта Диснея

Содержание этапов проектирования Web-узла: концептуальное, логическое, физическое проектирование.

### Концептуальное проектирование

- На этом этапе следует описать следующее:
  - Логическое проектирование
    - На этом этапе следует описать следующее:
  - Физическое проектирование
    - На этом этапе следует описать следующее:

Хостинг Web-узла.

Программное обеспечение рабочего места Webразработчика

- Необходимые программы:
- Вспомогательные программы:
- Необходимые программы Html-редактор
- Macromedia Home Site 5
- Графический редактор Adobe Photoshop
- Web-сервер

## 2.1разметка языков гипертекста

ПОНЯТИЕ О СТАНДАРТНОМ ОБОБЩЕННОМ ЯЗЫКЕ РАЗМЕТКИ SGML. КОНСОРЦИУМ W3C.

- История возникновения
- Структура консорциума
- Особенности внедрения рекомендаций
- Рабочий проект (WD — Working Draft)

Кандидат в рекомендации (CR — Candidate Recommendation)  
Предложение в рекомендации (PR — Proposed Recommendation)  
Рекомендация W3C (REC — W3C Recommendation)  
Последующие изменения

ВЕРСИИ ЯЗЫКА ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML. ПОНЯТИЕ О РАСШИРЯЕМОМ ЯЗЫКЕ РАЗМЕТКИ XML.

## Типы структур HTML и XHTML

### 2.2

Основные понятия

Первое правило

Общая структура HTML-файла

Заголовки, абзацы, разрывы строк

Заголовки

Это заголовок второго уровня

Это заголовок третьего уровня

Это заголовок четвертого уровня

Абзацы

Разрывы строк

Предварительно отформатированный текст

Выделенный текст

### 2.3

Форматирование текста

Пример:

Направление текста и таблиц

Элементы структурного форматирования текста

Пример неправильного использования:

Элементы физического форматирования документа

Шрифты

Элемент FONT

Элемент BASEFONT

Элементы Q и BLOCKQUOTE

Работа со строками и абзацами

Элемент P

### 2.4

Язык HTML обладает возможностями предоставления информации в виде списков. Список служит для добавления структуры в документ. Он позволяет в более удобном и понятном виде представить информацию, такую как, например: список покупок, пошаговое описание каких-либо действий, толковый словарь и т.п.

**Type** — задает информацию о виде используемых маркеров. Может принимать следующие значения: **\*Circle\*** — маркеры отображаются в виде незакрашенных кружков; **\*Disc\*** — маркеры отображаются в виде закрашенных кружков; **\*Square\*** — маркеры отображаются в виде закрашенных квадратов.

**Compact** — присутствие этого атрибута указывает браузеру на то, что данный список он должен отображать более компактно (например, уменьшив межстрочковый интервал).

**title** — заголовок элемента (выводится браузером в качестве комментария при наведении курсора на содержимое элемента);

Для элемента LI определены такие же стандартные атрибуты, что и для элемента UL, и еще два необязательных атрибута value и type. Эти атрибуты используются при работе с упорядоченными списками и будут рассмотрены позже. Визуально функции элемента LI сводятся к отображению маркера в неупорядоченных списках или нумерации в упорядоченных списках.

С помощью атрибута TYPE элемента UL можно непосредственно указывать вид пульки (маркера). Например (рис. 3.56):

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

**Type** — указывает вид нумерации элементов упорядоченного списка. Может принимать следующие значения:

**Type = a** — задает нумерацию строчными латинскими буквами; **Type = I** — задает нумерацию большими римскими цифрами; **Type = i** — задает нумерацию маленькими римскими цифрами.

**Start** — задает начальный номер первого элемента в упорядоченном списке. В качестве значения может

**lang, dir** — информация о языке и направленности текста;

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

В этом примере также проиллюстрировано использование атрибута value элемента 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. В качестве примера приведен отрывок кода документа, изображенного на рис. 3.62.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> <HTML> <HEAD>
```

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

```
<БЖультура Византии
```

#### **Элементы списков MENU и DIR. Использование этих элементов**

Элемент MENU позволяет задавать список как пункты меню. Элемент DIR представляет собой список-каталог. Оба этих списка отображаются как неупорядоченные, и использование обоих этих списков является нежелательным.

```
<БЖультура Византии
```

#### **Элементы списков MENU и DIR. Использование этих элементов**

Элемент MENU позволяет задавать список как пункты меню. Элемент DIR представляет собой список-каталог. Оба этих списка отображаются как неупорядоченные, и использование обоих этих списков является нежелательным.

```
<LI>Египетские пирамиды
```

## тема 1.2

### типы адресов сети интернет

Типы адресов:

1. Физический (MAC-адрес)
2. Сетевой (IP-адрес)
3. Символьный (DNS-имя)

Компьютер в сети TCP/IP может иметь адреса трех уровней (но не менее двух):

- Локальный адрес компьютера. Для узлов, входящих в локальные сети - это MAC-адрес сетевого адаптера. Эти адреса назначаются производителями оборудования и являются уникальными адресами.
- IP-адрес, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов.
- Символьный идентификатор-имя (DNS), например,

### *структура назначение классы ip*

**IP\*\*v4\*\*** - адрес является уникальным 32-битным идентификатором IP-интерфейса в Интернет.

**IP\*\*v6\*\*** - адрес является уникальным 128-битным идентификатором IP-интерфейса в Интернет, иногда называют **Internet-2**, адресного пространства IPv4 уже стало не хватать, поэтому постепенно вводят новый стандарт.

IP-адреса принято записывать разбивкой всего адреса по октетам (8), каждый октет записывается в виде десятичного числа, числа разделяются точками. Например, адрес

## *использование масок при адресации*

Маска — это число, которое используется в паре с IP-адресом; двоичная запись маски содержит единицы в тех разрядах, которые должны в IP-адресе интерпретироваться как номер сети. Поскольку номер сети является целой частью адреса, единицы в маске также должны представлять непрерывную последовательность.

Для стандартных классов сетей маски имеют следующие значения:

- класс A - 11111111. 00000000. 00000000. 00000000 (255.0.0.0);
- класс B - 11111111.11111111. 00000000. 00000000 (255.255.0.0);

## *Стек протоколов TCP/IP*

TCP/IP - собирательное название для набора (стека) сетевых протоколов разных уровней, используемых в Интернет. Особенности TCP/IP:

- Открытые стандарты протоколов, разрабатываемые независимо от программного и аппаратного обеспечения;
- Независимость от физической среды передачи;
- Система уникальной адресации;
- Стандартизированные протоколы высокого уровня для распространенных пользовательских сервисов.



## Система доменных имён DNS.

**DNS**(англ. *Domain Name System* — система доменных имён) — распределённая система преобразования имени хоста (компьютера или другого сетевого устройства) в IP адрес. DNS работает в сетях TCP/IP. Как частный случай, DNS может хранить и обрабатывать и обратные запросы, определения имени хоста по его IP (PTR-записи).

## *Ключевые характеристики DNS*

DNS обладает следующими характеристиками:

- *Распределённость хранения информации.* Каждый узел сети в обязательном порядке должен хранить только те данные, которые входят в его зону ответственности и (возможно) адреса корневых DNS-серверов.
- *Кеширование информации.* Узел может хранить некоторое количество данных не из своей зоны ответственности для уменьшения нагрузки на сеть.
- *Иерархическая структура,* в которой все узлы объединены в дерево, и каждый узел может или самостоятельно определять работу нижестоящих узлов, или делегировать (передавать) их другим узлам.
- *Резервирование.* За хранение и обслуживание своих узлов (зон) отвечают (обычно) несколько серверов, разделённые как физически, так и логически, что обеспечивает сохранность данных и продолжение работы даже в случае сбоя одного из узлов.

DNS важна для работы Интернета, ибо для соединения с узлом необходима информация о его IP-адресе, а для людей проще запоминать буквенные (обычно осмысленные) адреса, чем последовательность цифр IP-адреса. В некоторых случаях это позволяет использовать виртуальные серверы, например, HTTP-сервера, различая их по имени запроса. Первоначально преобразование между доменными и IP-адресами производилось с использованием специального

текстового файла HOSTS, который составлялся централизованно и обновлялся на каждой из машин сети вручную. С ростом Сети возникла необходимость в эффективном, автоматизированном механизме, которым и стала DNS.

## *Домены верхнего уровня.*

Первые домены верхнего уровня были рассчитаны на США:

- gov - государственные организации
- mil - военные учреждения
- edu - образовательные учреждения
- com - коммерческие организации
- net - сетевые организации

Позднее, когда сеть перешагнула национальные границы США появились национальные домены типа:

- uk - Объединенное королевство
- jp - Япония
- au - Австралия
- ch - Чехия
- su - СССР
- ru - Россия
- и т.п.

## *Принципы функционирования доменной системы DNS.*

Служба доменных имен работает как распределенная база, данные которой распределены по DNS-серверам.

Система доменных имен - это сервис прикладного уровня, значит, использует транспорт **TCP** и **UDP**.

## *Программная поддержка системы DNS.*

Сервис DNS строится по схеме "клиент-сервер". В качестве клиентской части выступает процедура разрешения имен - **resolver**, а в качестве сервера DNS-сервер (**BIND ...**)

Например, когда мы хотим обратиться к серверу ipm.kstu.ru, ваш браузер, используя **resolver**, поступает следующим образом:

1. ищет запись ipm.kstu.ru в файле hosts, если не находит, то,
2. посылает запрос на известный DNS-кэширующий сервер (как правило, локальный), если на этом сервере запись не найдена, то,
3. сервер DNS-кэширующий обращается к DNS-ROOT серверу с запросом адреса DNS сервера отвечающего за домен первого уровня ru, если получает адрес, то,
4. сервер DNS-кэширующий обращается к DNS серверу, отвечающего за домен первого уровня ru, с запросом адреса DNS сервера отвечающего за домен второго уровня kstu.ru, если получает адрес, то,
5. сервер DNS-кэширующий посылает запрос на DNS сервер, отвечающий за домен второго уровня kstu.ru, если получает адрес, то,
6. сервер DNS-кэширующий адрес кэширует и передает клиенту
7. клиент обращается по IP адресу - 195.208.44.20

# Система универсальных идентификаторов ресурсов URI и URL.

**URI** (Uniform Resource Identifier, Универсальный идентификатор ресурса) (RFC 2396, August 1998) - компактная строка символов для идентификации абстрактного или физического ресурса. Под ресурсом понимается любой объект, принадлежащий некоторому пространству. Включает и переопределяет определенные ранее URL (RFC 1738/RFC 1808) и URN (RFC 2141, RFC 2611).

URI предназначен для уникальной идентификации любого ресурса.

Некоторые подмножества URI:

**URL** (Uniform Resource Locator, Универсальный указатель ресурса), - подмножество схем URI, который идентифицирует ресурс по способу доступа к нему (например, его "местонахождению в сети") вместо того, чтобы идентифицировать его по названию или другим атрибутам этого ресурса.

Примеры URL:

- <http://www.ipm.kstu.ru/index.php>
- <ftp://www.ipm.kstu.ru/>

В HTML записывается так:

```
<a href="http://www.ipm.kstu.ru/index.php">` ` </a>
```

**URN** (Uniform Resource Name, Универсальное имя ресурса) - частная URI-схема "urn:" с подмножеством "пространства имен", который должен быть уникальным и неизменным даже в том случае, когда ресурс уже не существует или недоступен.

Предполагается что, например браузер, знает, где искать этот ресурс.

Синтаксис:

urn:namespace: data1.data2.more-data, где namespace (пространство имен) определяет, каким образом используются данные, указанные после второго ":".

Пример URN:

- urn:ISBN: 0-395-36341-6
- ISBN - тематический классификатор для издательств
- 0-395-36341-6 - конкретный номер тематики книги или журнала

При получении URN клиентская программа обращается к ISBN (каталогу "тематический классификатор для издательств" в Интернете). И получает расшифровку номера тематики "0-395-36341-6" (например: "квантовая химия").

URN массово используется в P2P сетях (подобных edonkey).

Пример URN указывающего на образ диска Adobe Photoshop v8.0 в сети edonkey:

```
urn:ed2k://file|Adobe Photoshop v8.0.iso|940769280|b34c101c90b6dedb4071094cb1b9f2d3|/
```

где:

1. ed2k - указывает на сеть
2. file - файл
3. Adobe Photoshop v8.0.iso - название файла
4. 940769280 - размер в байтах
5. b34c101c90b6dedb4071094cb1b9f2d3 - идентификатор файла (вычисляется с помощью хеш-функции)

## *Принципы системы: расширяемость, полнота, читаемость.*

URL - Uniform Resource Locators явно описывает, как добраться до объекта.

Синтаксис:

<scheme>:<scheme-specific-part>

где:

scheme = "http" | "ftp" | "gopher" | "mailto" | "news" | "telnet" | "file" | "man" | "info" | "whatis" | "ldap" | "wais" | ... - имя схемы

scheme-specific-part - зависит от схемы

Имя схемы - последовательность символов [a-z0-9+.-].

В scheme-specific-part можно использовать шестнадцатеричные значения в виде: %5f. Обязательно должны кодироваться непечатные октеты: 00-1F, 7F, 80-FF. Также всегда кодируются "небезопасные" символы: " ", "<", ">", "'", "#", "%", "{", "}", "|", ":", "\^", "~", "[", "]", "\`. Некоторые схемы резервируют и другие символы: ";", "/", "?", ":", "@", "=", "&". Их также необходимо кодировать, если хочется "обойти" их специальное трактование. Таким образом остаются [a-z0-9\$-\_.+!\*'(),] и резервированные символы в их специальном значении для данной схемы.

## *Формат URL.*

URL - Uniform Resource Locators явно описывает, как добраться до объекта.

Синтаксис:

<scheme>:<scheme-specific-part>

где:

scheme = "http" | "ftp" | "gopher" | "mailto" | "news" | "telnet" | "file" | "man" | "info" | "whatis" | "ldap" | "wais" | ... - имя схемы

scheme-specific-part - зависит от схемы

Имя схемы - последовательность символов [a-z0-9+.-].

В scheme-specific-part можно использовать шестнадцатеричные значения в виде: %5f. Обязательно должны кодироваться непечатные октеты: 00-1F, 7F, 80-FF. Также всегда кодируются "небезопасные" символы: " ", "<", ">", "\"", "#", "%", "{", "}", "|", ":", " ", "^", "~", "[", "]", "\`. Некоторые схемы резервируют и другие символы: ";", "/", "?", ":", "@", "=", "&". Их также необходимо кодировать, если хочется "обойти" их специальное трактование. Таким образом остаются [a-z0-9\$\_-+!\*'(),] и резервированные символы в их специальном значении для данной схемы.

## Структура стеков TCP/IP.

*Стек TCP/IP и модель OSI.*

*Инкапсуляция протоколов верхнего уровня в протоколы стека TCP/IP.*

*Структура пакетных TCP.*

# тема 1.3 Сервисы глобальной сети интернет

## типы сервисов глобальной сети

- World Wide Web;
- сервис имён доменов (DNS – Domain Name System);
- доступ к файловым архивам (FTP);
- электронная почта (e-mail);
- телеконференции (Usenet) и списки рассылки;
- сервисы общения ICQ, IRC и др.;
- сервис Telnet;
- поиск информации в Internet.



## *программы клиенты*

Некоторые программы-клиенты FTP (File Transfer Protocol) и WWW-браузеры поддерживают полезную функцию FTP-протокола `get`, позволяющую продолжить загрузку файла, передача которого была прервана по какой-либо причине, а не начинать процесс сначала. Естественно, FTP-сервер тоже должен поддерживать эту функцию.

## *браузеры*

Для доступа к информации в WWW на клиентских компьютерах используется специальное программное средство, называемое браузером (browser). Это приложение позволяет пользователю перемещаться по разнообразным данным Internet, предлагаемым в основном серверами WWW. В настоящее время наиболее популярными браузерами являются Microsoft Internet Explorer, Mozilla Firefox и Opera.

## веб сервис

### *архитектура www*

- язык гипертекстовой разметки документов HTML (HyperText Markup Language);
- универсальный способ адресации ресурсов в сети URL (Universal Resource Locator);
- протокол обмена гипертекстовой информацией HTTP (HyperText Transfer Protocol),

### *основные компоненты www(htm url http sgi )*

Позже команда NCSA добавила к этим трем компонентам четвертый: универсальный интерфейс Шлюзов CGI (Common Gateway Interlace)

Технология WWW состоит из четырех основных компонентов:

- язык гипертекстовой разметки документов HTML (HyperText Markup Language);
- универсальный способ адресации ресурсов в сети URL (Universal Resource Locator);
- протокол обмена гипертекстовой информацией HTTP (HyperText Transfer Protocol);
- универсальный интерфейс шлюзов CGI (Common Gateway Interface).

HTML (Hyper Text Markup Language) — язык разметки гипертекста возник на стыке нескольких направлений исследований и разработок. Язык HTML использует команды — теги, вводимые в текстовые документы, которые указывают, каким образом информация должна на выводиться на экран.

HTML-документ представляет собой текстовый файл, который может быть создан с помощью любого текстового редактора, но, в отличие от обычных текстовых файлов, он имеет расширение `.htm` (или `.html`).

## *принципы работы клиент - сервис*

От описания основных компонентов перейдем к архитектуре взаимодействия программного обеспечения в системе World Wide Web. WWW построена по хорошо известной схеме «клиент—сервер». Программа-клиент выполняет функции интерфейса пользователя и обеспечивает доступ практически ко всем информационным ресурсам Интернета. В этом смысле она выходит за обычные рамки работы клиента только с сервером определенного протокола.

## *назначение web технологий и роль www*

### **технологии поддерживаемые сервером**

Другую часть программного комплекса WWW составляет сервер протокола HTTP, базы данных документов в формате HTML, управляемые сервером, и программное обеспечение, разработанное в стандарте спецификации CGI. До самого последнего времени реально использовалось два HTTP-сервера: сервер CERN и сервер NCSA. Но в настоящее время число базовых серверов расширилось.

### **технологии поддерживаемые клиентом**

Клиент — это интерпретатор HTML. И, как типичный интерпретатор, клиент в зависимости от команд (разметки) выполняет различные функции. В круг этих функций входит не только размещение текста на экране, но и обмен информацией с сервером по мере анализа полученного HTML-текста, что наиболее наглядно происходит при отображении встроенных в текст графических изображений. При анализе URL-спецификации или по командам сервера клиент запускает дополнительные внешние программы для работы с документами в форматах, отличных от HTML, например GIF, JPEG, MPEG, Postscript и т.п.

## 1.4

### Понятие гипертекста.

гипертекст — это метод баз данных (БД), вводящего новую форму прямого доступа к данным.

### **Организационная структура информации Web-узла (иерархическая, линейная, в виде паутины).**

### **Модели проектирования Web-узла.**

Модель проектирования это метод разработки проекта, основанный на определённом представлении его разработчиков, как необходимо создавать свои работы. Ниже представлены некоторые распространённые модели, которые можно использовать для создания сайта и других проектов. Модель водопада



Рис.1. Модель водопада

Модель водопада предполагает выполнение нескольких, следующих друг за другом этапов, приведенных на (рис. 1) Вначале идет написание технического задания, затем анализ и проектирование сайта; следующие этапы — написание контента, дизайн и программирование. Заканчивается все генеральным тестированием и приёмкой проекта.

**\*Достоинства\*.** Это одна из самых простых и доступных моделей. Все этапы идут последовательно, каждый последующий не начинается, пока не закончится предыдущий.

**\*Недостатки\*.** Эта модель предполагает точное знание того, чего хочется реализовать на сайте. Обычно же, обстоит как раз наоборот, очень трудно сразу сформулировать цели, которые следует выполнить.

## Спиральная модель



Рис. 2. Спиральная модель

Данная модель использует противоположный подход, нежели модель водопада. Работа начинается с этапа «Планирование и анализ» и по часовой стрелке переходит к этапам выполнения, тестирования полученных результатов и оценки. На следующей итерации всё повторяется по новой, но уже с учётом выявленных недочётов проекта. Таким образом, пройдя несколько итераций и повторив все этапы несколько раз, проект избавляется от недостатков, обрстая дополнительными возможностями и преимуществами.

**\*Достоинства\*.** Возможность разрабатывать проект за несколько итераций позволяет постепенно его улучшать, реализовывая различные идеи.

**\*Недостатки\*.** «Лучшее — враг хорошего». На каком-то этапе всегда следует остановиться, чтобы представить проект. Но в данной модели чёткие критерии его выполнения отсутствуют. Это создает определенные сложности для расчета финансовых затрат на выполнение проекта.

# Microsoft Solutions Framework (MSF)



Рис.3. Модель Microsoft Solutions Framework

Данная модель сделана компанией Microsoft для своих собственных целей, но приобрела популярность и среди других разработчиков. Все программные продукты Microsoft создаются именно по этой методологии. Модель MSF вобрала в себя лучшее из двух моделей, описанных выше — спиральной и водопада. Состоит из четырех этапов: анализ, планирование, разработка и стабилизация (рис. 3). Каждый этап заканчивается достижением определённого результата, например, после анализа проекта идёт одобрение продуманной концепции. В итоге пишется определённый документ, в котором записывается результат выполнения данного этапа, для того, чтобы каждый из разработчиков чётко понимал своё место в проекте и задачи, которые ему предстоит решить. Модель итерационна и при прохождении всех этапов, проект можно доработать с учётом предыдущей итерации. Но, поскольку, окончание каждого этапа чётко указано, нет, как в спиральной модели, бесконечного повторения одного и того же процесса.

**\*Достоинства\*.** Пожалуй, MSF является одной из самых интересных моделей разработки и создания проектов, взявшей лучшее из других моделей и отказавшись от присущих им недостатков.

**\*Недостатки\*.** Практически их нет. Разве что можно отнести к ним большую, по сравнению с другими моделями, сложность. Поэтому для небольших сайтов рекомендуется использовать другие методы.

## Модель Уолта Диснея

Модель Уолта Диснея используется для проектирования сайта и состоит из трех этапов. В терминах модели эти этапы называются фазами мечтателя, реалиста и критика. Однако, мы будем придерживаться других названий, поскольку они более точно отражают суть процесса разработки сайта. Итак, процесс проектирования разбивается на следующие стадии:

- \1. Концептуальное проектирование.
- \2. Логическое проектирование.
- \3. Физическое проектирование.

Этапы следуют последовательно один за другим (рис. 4), но в некоторых случаях возможен переход к следующей стадии без окончания предыдущей. Это может происходить, например, когда разработчиков несколько и каждый работает со своей частью сайта. В любом случае, после окончания этапа физического проектирования следует вернуться к началу и внести соответствующие коррективы.



Рис.4. Этапы проектирования

Содержание этапов проектирования Web-узла: концептуальное, логическое, физическое проектирование.

## Концептуальное проектирование

Порой бывает сложно оценить эффективность сайта. Есть однако универсальный критерий, который довольно точно характеризует эффективность сайта. Это достижение разработчиками сайта поставленных перед ними целей. В этом случае сайт превращается в действенный инструмент, который выполняет возложенные на него функции.

Концептуальное проектирование служит для указания целей, задач сайта и определения аудитории, на которую он рассчитан.

*На этом этапе следует описать следующее:*

- \1. Основные и второстепенные цели.
- \2. Действия, которые необходимо предпринять для достижения поставленных целей.
- \3. Состав пользователей.
- \4. Интересы групп пользователей.
- \5. Разделы сайта.
- \6. Критерии достижения цели.

## *Логическое проектирование*

Разделы сайта, продуманные на предыдущем этапе, пока не упорядочены и не структурированы, поэтому их нужно привести к удобному и понятному виду. Логическое проектирование включает организацию информации на сайте, построение его структуры и навигации по разделам.

На данном этапе следует задаться вопросом, каким образом будет упорядочена информация. Варианты могут быть самыми разными и зависеть от типа данных и предпочтений создателей сайта: по времени, разделам, в алфавитном порядке, определенным группам или другим критериям. Так, для сайта музыкальной группы, поиск определенной песни можно сделать в виде алфавитного указателя, по названию альбома, первым строчкам песни, году выпуска и по ключевым словам. Одновременное использование различных способов охватывает большую аудиторию и позволяет быстрее найти нужную информацию на сайте.

### **На этом этапе следует описать следующее:**

- \1. Тип структуры сайта (линейная, иерархическая, контекстная, другая).
- \2. Названия разделов.
- \3. Что будет содержать в себе каждый раздел.
- \4. Организация и связь разделов между собой.
- \5. Какая информация будет размещена на определенных страницах сайта.

Конечный результат логического проектирования оформляется в виде блок-схем, структурных диаграмм или другими способами, показывающими взаимосвязь различных частей сайта.

## *Физическое проектирование*

Этап поиска проблем, а не их решений, связанных, по большей части, с технической реализацией сайта.

### *На этом этапе следует описать следующее:*

- \1. Технологии, которые будут применяться на сайте.
- \2. Используемое программное обеспечение.
- \3. Возможные проблемы и способы их устранения.

#### 4. Как будет обновляться информация.

После завершения данного этапа следует вернуться к концептуальному проектированию и проверить, не нужно ли внести изменения, в связи с переосмыслением проекта на других стадиях.

## Хостинг Web-узла.

WEB-хостинг – это услуга предоставления пространства на дисковых носителях сервера для размещения, хранения и поддержки веб-сайта клиента. Чтобы понять, для чего нужен хостинг, представьте, что Вы создали сайт. Пока он находится на Вашем компьютере, его никто, кроме Вас не увидит.

Большинство сайтов размещаются на удаленных серверах, которые всегда подключены к высокоскоростным Интернет-магистралям и обслуживаются специалистами. Это и есть хостинг. Компания, предоставляющая услуги хостинга называется провайдером хостинга или хостером.

В хостинг входит размещение всех файлов сайта клиента (графики, веб-страниц, скриптов, таблиц стилей и пр.), а также обработка скриптов и баз данных специальным программным обеспечением, находящимся на сервере хостера. Зачастую хостер предоставляет клиенту также адреса электронной почты, базы данных MySQL, PHP, SSI, Perl, Cron, FTP-доступ, доступ к файлу .htaccess, систему управления сайтом (CMS) и прочее. Все это зависит от выбранного вами тарифного плана. В услугу хостинга входит поддержка работоспособности сайта и обеспечение сохранности файлов. Для этого ежедневно осуществляется резервное копирование (back-up). Большой минус хостингу, который не выполняет регулярно резервное копированные данных клиентов.

Хостинг может быть:

§ платным

§ бесплатным

Бесплатный хостинг предоставляется без оплаты услуг, но, как правило, имеет существенные ограничения по предоставляемым функциям (например, отсутствует поддержка PHP или баз данных). Кроме этого, взамен за бесплатное размещение сайтов, хостер может вставлять на страницы клиента свою рекламу. Обычно сайты, пользующиеся услугами бесплатного хостинга – это любительские сайты. Чаще всего они имеют домены третьего уровня, предоставляемые самим хостером.

Платный хостинг надежнее и мощнее бесплатного. От суммы оплаты зависит количество дискового пространства и набор функций. В наиболее дешевые могут не входить поддержка баз данных MySQL, PHP, Perl и другое. Очень часто, при оплате хостинга на долгий срок, бесплатно предоставляется домен второго уровня, что позволяет существенно сократить затраты на содержание сайта. Но стоит обратить внимание на то, чтобы домен был зарегистрирован на ваше имя, а также вы имели доступ к управлению доменом. Это может решить ваши проблемы в будущем при переезде на другой хостинг.

Для того, чтобы разместить свой веб-сайт в сети, необходимо:

§ прежде всего, иметь собственный веб-сайт. Нужно иметь копию сайта на локальном компьютере (в html-файлах), или же готовые материалы + скрипт, который позволял бы создать веб-сайт непосредственно на сервере.

§ доменное имя. Нужно найти и приобрести доменное имя сайта. Желательно, чтобы имя говорило о тематике сайта, и было легко запоминающимся. Доменное имя может быть в любой, на ваш выбор, доменной зоне, например: .com .net .org



.ru и т.д.

§ заказать один из хостинг-планов в компании, осуществляющей услуги хостинга.

## Программное обеспечение рабочего места Webразработчика

### *Необходимые программы:*

 Htm1-редактор

 Графический редактор  Браузер(ы)

 Web-сервер

### *Вспомогательные программы:*

 FTP-клиент

 Программы для просмотра изображений  Программы для работы с CSS

 Notepad

### *Необходимые программы Htm1-редактор*

Что такое html-редактор? Это программа, помогающая Web-мастеру писать html-коды (html-теги) и создавать html-страницы. Такие программы позволяют генерировать код при нажатии на определенные кнопки или при выборе требуемых опций в меню, освобождая пользователя от огромной рутинной работы по написанию кодов, их многочисленных атрибутов и значений.



## *Macromedia Home Site 5*

Home Site 5 — это профессиональный кодовый редактор, и в процессе создания страниц вы можете и писать код «вручную», и одновременно использовать возможности редактора:

Иными словами, этот редактор создает «чистый» html-код. Он потребует от вас знания языка разметки.

## *Графический редактор Adobe Photoshop*

Лучший на сегодняшний день профессиональный графический редактор растровой графики — Adobe Photoshop

Adobe Photoshop 7 используется не только для создания, редактирования и подготовки изображений для размещения на Web-страницах, но и в полиграфии.

Вместе с Adobe Photoshop 7 поставляется программа Adobe ImageReady 7, в которой можно найти полезные функции для Web-дизайна, например для создания анимированных изображений, карт-ссылок, функции rollover с автоматической генерацией html- и JavaScript кода.

## *Web-сервер*

Для отладки страниц потребуется установить Web-сервер. Динамичная страница предварительно обрабатывается сервером — выполняется встроенный в нее код, и только после этого готовый результат высылается клиенту. В настоящий момент практически любая Web-страница — динамичная, т. е. создается с использованием

серверных технологий. Например, технология SSI (Server Side Include) — включения на стороне сервера — была первой, позволяющей осуществлять предварительную обработку кода страницы на сервере. Более подробно речь об этой технологии пойдет на шестом уроке.

Популярные Web-серверы — Apache и IIS. По статистике около 70% сайтов используют в качестве программного обеспечения Web-серверы Apache и примерно 25% — IIS.

# 2.1 разметка языков гипертекста

ПОНЯТИЕ О СТАНДАРТНОМ ОБОБЩЕННОМ ЯЗЫКЕ  
РАЗМЕТКИ SGML. КОНСОРЦИУМ W3C.

---

Любой документ имеет три составляющих:

- содержание;
- структуру;
- стиль.

**Содержание** документа на бумаге может быть сугубо текстовым, а также содержать изображения. Если документ представлен в электронном виде, он может содержать и мультимедийные данные, а также ссылки на другие документы.

Обычно содержание документа представляется не в произвольном порядке, а имеет определенную **структуру**. Структура – это состав и порядок следования частей (блоков) документа.

**Стиль** документа определяет форму вывода его содержания на то или иное устройство (например, принтер или дисплей). В понятие стиль входят характеристики шрифта (наименование, размер, цвет) всего выводимого документа или отдельных его блоков, порядок разбивки на страницы, расположение блоков на страницах и другие параметры.

**Языки разметки документов** являются искусственными языками, предназначенными для описания структуры документа и отношений между различными объектами структуры. Данные разметки называются также **метаданными**.

Первым языком разметки является **язык GML**. Его непосредственным наследником стал **язык SGML** – стандартный обобщенный язык разметки, определяющий правила записи элементов разметки документа.

SGML был разработан для совместной разработки машинных документов в больших правительственных и аэрокосмических проектах. Он широко применялся в печатном деле и издательской сфере, но его сложность затруднила повседневное использование. Основные наследники SGML — форматы HTML и XML.

требования к языку разметки документов:

1. Язык должен быть доступен для чтения человеком.
2. Размеченные файлы документов должны быть текстовыми и кодироваться с помощью символов кода ASCII
3. В языке могут использоваться ссылки как на внутренние ресурсы (в том же документе), так и на внешние ресурсы (в других документах).

В языке SGML и подобных ему языках используются специальные инструменты разметки документа:

- структуры документа;
- дескрипторы или элементы и сопутствующие им атрибуты;
- сущности (entities);
- комментарии.

Документы SGML имеют древовидную структуру.

**Дескрипторы** в SGML размещаются в начале (открывающий дескриптор) и в конце (закрывающий дескриптор) каждого *элемента* (item).

**Атрибуты** – это простые символьные конструкции (items), которые добавляются к элементам для придания им уточнения действия дескрипторов.

Языки обобщенной разметки, подобные SGML, допускают использование атрибутов, с которыми могут быть соотнесены до 15 различных типов значений, в том числе:

- Ссылки на любые ресурсы, находящиеся вне документа, на которые обычно ссылаются как на сущности (entities).
- Уникальный идентификатор (ID) элемента в документе.
- Указатели идентификаторов (ID Pointers), имеющие перекрестные ссылки для тех элементов, которые имеют ID, упомянутые в документе.
- Обозначения или атрибуты элементов, которые определяют обозначения в содержании элемента.
- Символьные данные (character data), или CDATA, представляющие собой любые допустимые символы, которые не могут выступать в качестве значений атрибутов.

**Комментарии** позволяют добавить информацию, которая не будет видна после обработки документа. Комментарии не влияют на скорость обработки документа, не рассматриваются и не обрабатываются как часть содержания SGML-документа. Они просто включаются в исходный текст.

Для проверки соответствия документа разметке заданного типа используются специальные программы – **анализаторы** (parsers). Анализаторы являются либо отдельными программами, либо частью программы обработки документа SGML. Чтобы анализатор мог выполнить проверку документа, создается специальный документ, называемый **определением типа документа**

Язык **HTML** является приложением языка SGML для использования в Internet с фиксированной структурой, фиксированным набором элементов (дескрипторов) и их атрибутов, а также фиксированным набором сущностей. расширенный язык разметки **XML** (Extensible Markup Language).

HTML соответствует международному стандарту ISO 8879. Текущий стандарт HTML 4.01 существует с 1999 г. В стоящее время опубликован проект пятого стандарта языка. Новая версия HTML обещает дополнить язык многочисленными расширениями и обеспечить более простую, логичную и удобную систему правил.

Dynamic HTML или DHTML — это способ создания интерактивного Web- сайта. DHTML возник как набор методов динамического создания и изменения Web- страниц путем вызова из HTML- документа сценариев. Однако развитие этих методов

привело к полному пересмотру концепции Web-документа и формированию понятия объектной модели документа DOM (Document Object Model).

DOM представляет собой платформенно — независимый программный интерфейс, позволяющий программам и скриптам управлять содержимым документов HTML и XML, а также изменять их структуру и оформление.

Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый из которых содержит некоторый объект. Узлы связаны между собой отношениями «родитель-потомок».

Изначально многие браузеры имели собственную модель DOM, не совместимую с остальными. Для того чтобы обеспечить совместимость, специалисты международного консорциума W3C классифицировали эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу, носящую название W3C DOM.

Язык XML является подмножеством языка SGML, полностью совместимым с ним.

Язык XML обеспечивает широкий спектр функциональных возможностей, которые отсутствуют в HTML

Что касается Интернета в целом, нет такого человека или группы людей, которые бы управляли Всемирной паутиной. Однако, Консорциум Всемирной паутины (W3C, World Wide Web Consortium) играет проактивную роль в разработке рекомендаций и прототипов технологий, относящихся к сети. Темы, которых касается W3C – веб-архитектура, стандарты веб-дизайна и обеспечение доступности сайтов для инвалидов. В попытке стандартизировать веб-технологии W3C создает спецификации, которые называются рекомендациями. В их

создании принимают участие множества крупных корпораций по разработке веб-технологий. Следование рекомендациям W3C – первый шаг на пути осуществления доступности вашего сайта

## *История возникновения*

Консорциум был создан в 1994 году как консультативный орган для лидеров компьютерной индустрии. Крупнейшие мировые компании и корпорации договаривались в W3C об обеспечении совместимости своих продуктов и внедрении новых

технологических стандартов. Первым крупным успехом консорциума стала стандартизация языка гипертекстовой разметки HTML (англ. HyperText Markup Language) в 1996 году.

Дело в том, что в середине 1990-х годов ряд крупнейших производителей программного обеспечения планировал выпустить каждый свою версию языка HTML со своими названиями тегов. Разумеется, это привело бы к хаосу в Интернете, и в результате веб-страница одной компании была бы размечена совершенно не так, как страница другой компании. Из-за этого веб-браузер одной компании не мог бы отображать страницы, созданные по правилам другой компании.

Именно W3C принадлежит заслуга в том, что HTML был выпущен с единым базовым набором тегов и атрибутов и веб-страницы стали такими, какими мы их знаем сейчас. Хотя полного совпадения тегов и атрибутов HTML достичь, к сожалению, не удалось до сих пор. Это, конечно, усложняет жизнь веб-разработчикам

## *Структура консорциума*

Общую администрацию консорциума Всемирной паутины осуществляют 3 организации:

- Массачусетский технологический институт (англ. Massachusetts Institute of Technology, MIT) в США;

- Европейский консорциум по исследованиям в области информатики и математики (англ. European Research Consortium for Informatics and Mathematics, ERCIM) во Франции;

- Университет Кейо (англ. Keio University) в Японии.

Членом консорциума может стать юридическое или частное лицо, занимающееся веб-технологиями и заинтересованное в развитии Интернета. Основным источником финансирования консорциума являются членские взносы. Членами консорциума уже являются более 350 организаций.

Международную координацию осуществляют так называемые «офисы W3C» (англ. W3C Offices), которые созданы уже в 14 странах мира. Время от времени консорциум Всемирной паутины также устраивает международные конференции.

Процесс выработки рекомендаций проходит в специальных группах. Рабочие и исследовательские группы консорциума включают штатный персонал, представителей организаций-членов и приглашённых экспертов. С предложениями в адрес рабочих групп

может выступить любое лицо, даже не члены W3C. 16 февраля 2012 W3C совместно с НИУ ВШЭ открыли представительство консорциума в России.

## *Особенности внедрения рекомендаций*

Рекомендации консорциума Всемирной паутины открыты, то есть не защищены патентами и могут внедряться любым человеком без всяких финансовых отчислений консорциуму. В отличие от других организаций, занимающихся разработкой стандартов для Интернета, консорциум Всемирной паутины не имеет программ сертификации (на соответствие рекомендациям консорциума) и не планирует их вводить, поэтому рекомендации W3C получили гораздо большее распространение, нежели стандарты любых других организаций.

В то же время, из-за отсутствия сертификации многие производители следуют рекомендациям лишь частично. Рекомендации консорциума построены таким образом, что частичное внедрение не нарушает общих стандартов. Некоторые популярные рекомендации имеют несколько степеней внедрения — кому как удобнее. Степени внедрения — это новое слово в сетевых стандартах, которое принесло консорциуму Всемирной паутины и его рекомендациям заслуженную популярность и признание Процесс стандартизации

Рекомендации W3C – это заключительный этап ратификации процесса рассмотрения вопросов стандартов рабочей группой Консорциума Всемирной паутины (W3C). Эта ратификация означает, что документ был подвергнут общественному рассмотрению и обзору организациями членами W3C. Рекомендации W3C направлены на стандартизацию веб-технологий.

В соответствии с описанием процесса разработки рекомендации W3C документ проходит через несколько этапов:

## *Рабочий проект (WD — Working Draft)*

На рабочем уровне проекта, стандарт публикуется для рассмотрения сообщества. WD документ является первой формой стандарта, которая выкладывается в публичный доступ. Обычно принимаются практически все комментарии, хотя это и не является обязательным, и нет гарантий что комментарий будет учтён при дальнейшей работе над документом.

На данном этапе описание стандарта, скорее всего, может иметь существенные отличия от его окончательной формы. Таким образом, любой, кто разрабатывает WD документ должен быть готов к значительному изменению их реализации стандарта.

## *Кандидат в рекомендации (CR — Candidate Recommendation)*

Кандидат в рекомендации версии стандарта (CR) это уже более стабильная версия документа чем WD. На этот момент группа разработки отвечает на вопросы тех, кто считает что нужно внести изменения в суть реализации стандарта. удовлетворен тем, что стандарт не то, что нужно его. Целью CR является получение помощи от сообщества разработчиков, которым предстоит реализовывать этот стандарт. Документ стандарта может меняться и дальше, но начиная с этого момента, возможность внесения существенных изменений с стандарт как правило закрыта. Дизайн небольших деталей все еще может измениться из-за обратной связи с реализаторами стандарта.

## *Предложение в рекомендации (PR — Proposed Recommendation)*

Вариантом стандарта на уровне «предложения в рекомендации» (PR) документ становится после прохождения предыдущих двух этапов. Пользователи уже внесли свои предложения в стандарт и реализаторы стандарта тоже внесли свои изменения. На данном этапе документ представляется на рассмотрение Консультативному Совету W3C для окончательного утверждения.

Хотя этот шаг является очень важным, он редко приводит к внесению каких-либо существенных изменений в стандарт.

## **Рекомендация W3C (REC — W3C Recommendation)**

Это заключительный этап развития стандарта. На данный момент стандарт уже прошёл широкий обзор и тестирование, как в теоретических, так и практических условиях. На этом этапе документ уже одобрен W3C в качестве стандарта для широкого развертывания в соответствующей области.

## **Последующие изменения**

Рекомендации могут обновляться позднее отдельно опубликованными исправлениям, которые аккумулируют достаточно существенные изменения, сделанные со времени последней публикации. W3C также публикует различные виды информационных заметок, которые не предназначены для рассмотрения в качестве стандарта.

Рекомендации могут время от времени обновляться. К рекомендациям публикуются сообщения о выявившихся ошибках и неточностях (англ. errata). Когда накапливается достаточный запас выявленных ошибок, выходит новая, исправленная и доработанная редакция (англ. edition) рекомендации (например, «редакция 1.1»). В исключительных случаях вся рекомендация может быть отозвана консорциумом для переработки.

Для удобства пользователей консорциумом созданы специальные программы-валидаторы (англ. Online Validation Service), которые доступны по Сети и могут за несколько секунд

проверить документы на соответствие популярным рекомендациям W3C. Консорциумом также созданы многие другие утилиты для облегчения работы веб-мастеров и программистов. Большинство утилит — это свободные программы, все они бесплатные. В последнее время, следуя мировым тенденциям, консорциум в целом гораздо больше внимания уделяет проектам с открытым исходным кодом.

# ВЕРСИИ ЯЗЫКА ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML.

## ПОНЯТИЕ О РАСШИРЯЕМОМ ЯЗЫКЕ РАЗМЕТКИ XML.

---

Первая версия **языка гипертекстовой разметки – HTML** (HyperText Markup Language), так же, как и сама технология Web, была разработана Тимом Бернерсом Ли в 1991 г. Язык HTML является реализацией правил языка SGML для типа документов, которые были названы *документами HTML*. Язык задает фиксированную структуру, фиксированный набор тегов и их атрибутов, а также фиксированный набор сущностей. Программы обработки документов HTML называются *Web-браузерами*. Результатом обработки документа является *Web-страница*, выводимая на экран дисплея.

В 1994 г. группа поддержки Интернет – IETF (Internet Engineering Task Force) разработала спецификацию HTML 2.0, с которой началось широкое распространение языка HTML в сети Internet. В том же году был создан консорциум W3C (World Wide Web Corporation), объединивший 165 коммерческих и академических организаций, разработчиков и пользователей (с момента создания и по настоящее время эту организацию возглавляет Т.Б. Ли). Последняя версия спецификации HTML – HTML 4.01 была принята консорциумом в декабре 1999 г.

· Язык XML обеспечивает широкий спектр функциональных возможностей, которые отсутствуют в HTML

Последняя версия спецификации языка XML – XML 1.1 была принята в апреле

2004 г.

На основе языка XML концерн W3C разработал дальнейшее развитие языка HTML

– **язык XHTML** (Extended HTML – расширенный HTML). Первая версия этого языка – XHTML 1.0 была принята в январе 2000 г. Эта версия фактически представляет собой переформулирование HTML 4 как приложения XML 1.0. Предполагается, что дальнейшее развитие языка HTML будет осуществляться в соответствии со спецификациями XHTML.

Новая версия XHTML – XHTML 1.1 была принята консорциумом W3C в мае 2001 г. Эта рекомендация определяют новый тип документа – XHTML на основе модулей. Каждый модуль XHTML 1.1 содержит один или несколько элементов и/или атрибутов языка HTML.

В соответствии со спецификацией, документы XHTML 1.1 состоит из следующих групп модулей XHTML:

Модули ядра – это модули, наличие которых необходимо в любом типе документа, соответствующего спецификации XHTML (в эту группу входят модули Structure, Text, Hypertext и List).

Модуль Applet, содержащий единственный элемент **<applet>** (этот элемент признан устаревшим и вместо него рекомендуется использовать элемент **<object>**).



Модули текстовых расширений, в которых определены различные дополнительные модули текстовой разметки (в эту группу входят модули Presentation, Edit и Bi-directional Text).

Модули форм (в эту группу входят модули Basic Forms и Forms). Модули таблиц (в эту группу входят модули Basic Tables и Tables).

Модуль Image, предоставляющий базовые возможности внедрения изображений (этот модуль также может независимо использоваться в некоторых реализациях клиентскими картами-изображениями).

Модуль Client-side Image Map, предоставляющий элементы для клиентских карт-изображений (для функционирования этого модуля необходимо включение модуля Image).

Модуль Object, предоставляющий поддержку включения объектов общего назначения.

Модуль Frames, предоставляющий элементы, относящиеся к фреймам.

URL (с помощью этого элемента вычисляются относительные URL документа).

Модуль Name Identification, используемый для идентификации определённых элементов в документах HTML.

Модуль Legacy, определяющий элементы и атрибуты, которые уже не рекомендовались в предыдущих версиях HTML и XHTML и не рекомендуются в дальнейшем.

## ***Типы структур HTML и XHTML***

Согласно спецификации HTML 4.01 для документов HTML определены **три структуры**, описываемые тремя DTD. Разработчики Web-страниц должны включать в свои документы одно из трех объявлений типов. Разница между DTD заключается в поддерживаемых ими элементах. Объявление DTD должно размещаться в самом начале документа.

**HTML 4.01 Strict DTD** (строгое определение) включает все элементы и атрибуты, не являющиеся отмененными (deprecated) и не использующиеся в документах с фреймами.

Определение **HTML 4.0 Transitional DTD** (переходное определение) включает все элементы, включенные в строгое DTD, а также отмененные элементы и атрибуты.

Определение **HTML 4.0 Frameset DTD** (определение для фреймов) включает, помимо элементов переходного DTD, фреймы.

Первая строка документа HTML, определенного в соответствии со спецификацией XHTML

Эта строка определяет используемую версию XML и кодировку символов документа. При кодировании символов в XML используется двухбайтовый код Unicode. В качестве значений параметра **encoding** наиболее часто используются кодировки **UTF-8**, в котором значения первых 128 символов представляются в однобайтовой кодировке, символы наиболее распространенных языков (в том числе русского и украинского) – двумя байтами, а остальные символы тремя байтами. В кодировке **UTF-16** все символы представляются двумя байтами (эти кодировку рекомендуется использовать для русских и украинских документов HTML).

## 2.2

### Основные понятия

---

**HTML** (*HyperText Markup Language*) - язык разметки гипертекста - предназначен для создания Web-страниц.

Под *гипертекстом* в этом случае понимается текст, связанный с другими текстами указателями-ссылками.

HTML представляет собой достаточно простой набор кодов, которые описывают структуру документа. HTML позволяет выделить в тексте отдельные логические части (заголовки, абзацы, списки и т.д.), поместить на Web-страницу подготовленную фотографию или картинку, организовать на странице ссылки для связи с другими документами.

HTML не задает конкретные и точные атрибуты форматирования документа. Конкретный вид документа окончательно определяет только *программа-браузер* на компьютере пользователя Интернета.

HTML также не является языком программирования, но web-страницы могут включать в себя встроенные программы-скрипты на языках *Javascript* и *Visual Basic Script* и программы-апплеты на языке *Java*.

Даже, если вы не собираетесь в дальнейшем редактировать "вручную" текст HTML (предполагая использовать графические редакторы), знание языка HTML даст вам возможность как лучше использовать эти средства, так и увеличит ваши шансы сделать HTML-документ более доступным и "читаемым" при просмотре браузерами разных фирм.

Основными компонентами HTML являются:

- **Тег (tag).** Тег HTML это компонент, который командует Web- браузеру выполнить определенную задачу типа создания абзаца или вставки изображения.
- **Атрибут (или аргумент).** Атрибут HTML изменяет тег. Например, можно выравнивать абзац или изображение внутри тега.
- **Значение.** Значения присваиваются атрибутам и определяют вносимые изменения. Например, если для тега используется атрибут выравнивания, то можно указать значение этого атрибута. Значения могут быть текстовыми, типа *left* или *right*, а также числовыми, как например ширина и высота изображения, где значения определяют размер изображения в пикселях.

Теги представляют собой зарезервированные последовательности символов, начинающиеся с < (знака меньше) и заканчивающиеся > (знаком больше).

Закрытие тега отличается от открытия только наличием символа '/'.

Предположим, у нас есть гипотетический атрибут форматирования текста, управляемый кодом <X>, и мы хотим применить его к словам "Это мой текст".

HTML-последовательность кодов и собственно текста будет выглядеть так:

```
<X>Это мой текст</X>
```

Теги могут вкладываться друг в друга иерархически, но без пересечений, то есть допустимо вложение вида <teg1><teg2>` `</teg2> </teg1>, но не <teg1><teg2>

```
</teg1></teg2>.
```

Действие вложенных тегов объединяется. Например, если внутри тега, создающего жирное начертание шрифта, вложен тег курсива, то в результате получится жирный курсив.

## Первое правило

Первое правило HTML: закрывайте все, что вы открыли!

НО! Из этого правила, как и из всех остальных, существуют исключения.

HTML- программа должна начинаться тегом <HTML> и заканчиваться тегом </HTML>



<HTML>

</HTML>

..... (здесь будут другие теги программы)



HTML- программы состоят из двух основных частей: заголовка и тела. Заголовок ограничивается парой тегов<HEAD> и </HEAD>, а тело - парой тегов<BODY> и

</BODY>.

В результате HTML- программа выглядит следующим образом:



<HTML>

<HEAD>

... (здесь будет заголовок)

</HTML>

</HEAD>

<BODY>

.... (здесь будут другие теги тела программы)

</BODY>



Кроме того, каждая HTML- программа имеет заголовок, который помещается в заголовок окна броузера. Заголовок окна броузера создается при помощи двух тегов <TITLE> и

</TITLE> и содержится между тегами <HEAD> и </HEAD>. Тогда программа принимает следующий вид:



<HTML>

</HTML>

<HEAD>

;

</HEAD>

<BODY>

.... (здесь будут другие теги тела программы)

</BODY>



Некоторые авторы, пишущие об языке HTML, советуют записывать теги прописными буквами, другие - используют строчные. Редактор HTML - Allaire HomeSite 4.5.1,

например использует по умолчанию нижний регистр для записи тегов. При создании моих страниц использовались оба варианта написания тегов. Как видите, допустимо и то и другое. Современные браузеры допускают запись тегов в любом регистре.

Уже позднее я узнала, что нельзя делать категоричные заявления по этому вопросу. Существуют теги и атрибуты "чувствительные" к написанию прописными или строчными буквами. Это регламентируется стандартами языка HTML, определенными [Консорциумом W3C](#).

Обращайтесь к [первоисточнику](#)!

Хорошее знание технического английского обязательно!



При написании HTML-программ возникает необходимость вставки *комментариев* - поясняющих текстов, которые невидны при загрузке документа в браузер. Для этой цели служит тег `<!--`. Все, что заключено между символами `<!--` и `-->` считается комментарием и не отображается в браузере.



Еще один тег, который очень важен в HTML-программе, но так же не предназначается для отображения какого-либо объекта в браузере - тег `<META>`. Этот тег служит специальным целям, а именно - указания языка, на котором написан документ, его кодовой страницы, ключевых слов, используемых поисковыми системами для классификации этого документа и т.п. Теги `<META>` обычно вставляются в HTML- программу на заключительном этапе создания Web-страницы - *публикации*.



Для вставки в HTML-программу фрагмента программ, написанных на языке JavaScript или Viual Basic Script сценариев используют теги `<SCRIPT>` и `</SCRIPT>`.



## Общая структура HTML-файла

Суммируя вышесказанное приведем общую структуру HTML-файл :

`<HTML>`

`<HEAD>`

`<Мета-теги>`

`<Функции скриптов>`

`</HEAD>`

`<BODY>`

Основная часть документа

`</BODY>`

`</HTML>`

## Заголовки, абзацы, разрывы строк

### *Заголовки*

Каждый пользователь компьютера, работающий в текстовом редакторе Microsoft Word знаком с понятием *стиля заголовка*. В HTML тоже применяется это понятие для структурирования документа и выделения важности заголовка. Всего существуют 6 стилей заголовка. Каждый из них обозначается в HTML-документе парными тегами `<Hi>`

и `</Hi>` Здесь *i* обозначает важность стиля. *H1* обозначает самый важный стиль заголовка,

*H2* - стиль заголовка второго уровня, а *H6* - стиль заголовка самого нижнего уровня.

В подавляющем большинстве случаев для заголовков Web-страниц используют три первых уровня заголовков `<H1>`, `<H2>` и `<H3>`. Объясняется это тем, что размеры шрифтов оставшихся заголовков (теги `<H4>` - `<H5>`) меньше размера обычного шрифта Web-страницы.

Вот как в документ можно добавить очень важный заголовок.

# An important heading

А вот результат.

**An important heading**

Посмотрим другие примеры:

**Это заголовок второго уровня**

**Это заголовок третьего уровня**

Это обычный текст

**Это заголовок четвертого уровня**

**Это заголовок пятого уровня**

**Это заголовок шестого уровня**



## Абзацы

Понятие абзаца в HTML-документе также аналогично понятию абзаца в Microsoft Word. Абзац обозначается в документе парными тегами `<P>` и `</P>`. Впрочем, применение закрывающего тега не является строго обязательным.

НО! Специфика тега `<P>` заключается в том, что после текста, который находится в его пределах, пустая строка добавляется *автоматически*.

Следует помнить и о другой особенности текстовых абзацев: когда текст достигает правой границы окна Web-браузера, переход на новую строку осуществляется автоматически, независимо от расположения тега `<P>`.

Для отдельного абзаца можно указать тип, размер и цвет шрифта отличным от стиля остального документа.

Например:

*My*

greetings to you!`</P>`

А вот и результат -

**\*My greetings to you!\***

## Разрывы строк

Если в середине строки появилась необходимость ее разорвать - используйте **одиночный** тег переноса строки `<BR>`. (Это соответствует нажатию клавишной комбинации [Shift]- [Enter] в текстовых процессорах Word). Код `<BR>` не означает конца логического абзаца, и за строкой с этим кодом дополнительная пустая строка не появится.

Вот и нарушено первое правило!

Кроме тега `<BR>` не требует закрытия тег добавления изображения `<IMG>`. Использование закрывающего тега абзаца `</p>` также не является строго обязательным.

HTML довольно "демократичен": неправильный тег или неправильное вложение тегов обычно не приводят к "зависаниям" браузера, а только вызывает сообщение об ошибке в строке состояния окна браузера вашего Интернет-читателя. Разумеется ошибки могут привести к неправильному форматированию HTML-документа.

Примером использования тега `<BR>` может служить написание почтового адреса или [стихотворения](#).

Броузеры показывают текст в своем окне, автоматически осуществляя перенос слов. Поэтому, если вы считаете необходимым запретить разрыв блока текста с пробелами между словами - воспользуйтесь специализированным символом \* \* - символом незазрывного пространства (non-breaking space). Например,

Освежающий и бодрящий напиток Coca Cola, приобрел широкую популярность в нашей стране.

А вот и результат:

Освежающий и бодрящий напиток Coca Cola, приобрел широкую популярность в нашей стране.



Не следует использовать цепочку символов \* \* для выравнивания текста в окне броузера. Для этой цели рекомендуется использовать таблицы стилей.



## Предварительно отформатированный текст

Достоинством браузеров является их способность самостоятельно распределять текст в окне броузера. Но иногда вы не нуждаетесь в этой услуге, хотите самостоятельно определить представление вашего текста в окне броузера. Например, вы хотите представить код программы в наиболее удобочитаемом виде. Такую возможность вам предоставляет тег `<pre>`. Посмотрим пример:

```
void Node::Remove()
{
if (prev)
prev->next = next; else if (parent)

parent->SetContent(null);

if (next)
next->prev = prev;

parent = null;
}
</pre>
```

Это выполнится так

```
void Node::Remove()
{
if (prev)
prev->next = next; else if (parent)
parent->SetContent(null);

if (next)
next->prev = prev;

parent = null;
}
```

Красиво, правда? Привлекает внимание, удобно читать программный код. Это сделано при помощи таблицы стилей.

Обратите внимание на то, что в случае использования тега **<pre>**, текст отображается браузером точно в таком виде, как он был создан в HTML-документе. Сохраняются все пробелы,табуляции и переводы строк. Исключением является только новая строка, следующая сразу же за тегом **<pre>**. Таким образом, эти два примера кода HTML на экране дисплея будут показаны одинаково:

предварительно отформатированный текст

предварительно отформатированный текст

</pre>

А именно:

предварительно отформатированный текст предварительно отформатированный текст

У этого тега существует необязательный атрибут, указывающий желаемый размер строки в символах, а именно:

Попробуем?

Предмет истории есть жизнь народов и человечества. Непосредственно уловить и обнять словом - описать жизнь не только человечества, но одного народа, представляется невозможным.

Обычно, для отображения предварительно отформатированного текста используются моноширинные шрифты, все символы которого имеют одинаковую ширину. Для того, чтобы браузер "не забыл" об этом следует применить таблицы стилей.

Например, это можно сделать так:

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

```
pre { color: green; background: white; font-family: monospace; }
```

```
</style>
```

Результат такого определения стиля для тега <pre> вы можете видеть на этой странице.



Если вы устанавливаете цвет текста, желательно также установить цвет фона. Это поможет вам избежать ситуации, когда буквы будут практически неразличимы на близком к ним по цвету фоне.



Совет № 2. Вместо того, чтобы устанавливать цвет фона для элемента `<pre>`, установите цвет фона для элемента `<body>`. Например, это можно сделать так:

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

```
body { color: black; background: white; }
```

```
pre { color: green; font-family: monospace; }
```

```
</style>
```

## Выделенный текст

В тех случаях, когда необходимо обратить особое внимание пользователя на тот или иной фрагмент текста, следует использовать стандартные средства форматирования.

Стандартные средства форматирования представлены специальными тегами, которые

*обязательно* являются *парными* (т.е. имеются открывающий и закрывающий теги). Чтобы увеличить размер шрифта на один пункт, используйте тег `<big>`.

Чтобы выделить текст **полужирным шрифтом**, воспользуйтесь **тегом `<b>`** или **тегом**

`<strong>`.

Чтобы выделить текст *курсивом*, воспользуйтесь *тегом `<i>`* или *тегом `<em>`*.

Вы заметили, что в данном случае выделение текста тегами `<i>` и `<em>` различаются? Объяснение простое - в HTML-коде этой страницы используется внедренная таблица стилей, в которой для тега `<em>` указан шрифт: полужирный курсив.

Шрифт пишущей машинки можно имитировать с помощью тега `<tt>`.

С помощью тега `<small>` размер шрифта можно уменьшить на один пункт. Существуют и другие теги, которые также предназначены для выделения текста.

Кроме того, в HTML включена поддержка математических символов и научных обозначений. Для построения простейших равенств и выражений вам могут пригодиться два тега `<sub>` (нижний индекс) и `<sup>` (верхний индекс).

Например:

A<sup>2</sup>+B<sup>2</sup>=C<sup>2</sup>

CO<sub>2</sub>=углекислый газ

В HTML-коде это записано так:

A<sup>2</sup>+B<sup>2</sup>=C<sup>2</sup>

CO<sub>2</sub>=углекислый газ

Если вы намерены придерживаться хорошего стиля программирования Web-страниц (с помощью HTML и XHTML), то вместо тегов форматирования используйте каскадные таблицы стилей.



На сегодняшний день многие старые теги HTML, которые предназначались специально для выделения текста, вытеснены новыми возможностями форматирования с помощью каскадных таблиц стилей.

## 2.3

### Форматирование текста

#### *Информация о языке*

Сведения о языке, на котором представлена информация в документе, могут быть заданы с помощью атрибута `lang`. Этот атрибут присутствует почти у всех элементов языка HTML и может быть задан в любом месте документа. Сведения о языке могут быть полезны в следующих случаях:

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

Атрибут `lang` указывает код содержимого элемента. 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-cockney..... кокни (диалект английского).*

Весь список кодов языков, поддерживаемых HTML 4.01, приведен в [RFC 1766] ("Тэги и идентификация языков" Х. Алвестранд, 1995). Двух буквенные первичные коды зарезервированы в стандарте [ISO 639], вот наиболее распространенные из них:

*Ru..... русский*

*En..... английский*

*Fr..... французский*

*Es..... испанский*

*Zh..... китайский*

*Ja..... японский*

*It..... итальянский*

*De..... немецкий*

*Nl..... голландский*

*He..... иврит*

*Ar..... арабский*

*El..... греческий*

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

- ü Атрибут lang, заданный непосредственно для самого элемента.
- ü Ближайший родительский элемент, у которого определен атрибут lang.
- ü Информация о языке, указанная (или сервером или элементом META) в поле "Content-Language" HTTP-заголовка.
- ü Значение, принимаемое браузером по умолчанию.

*Пример:*

```
<HTML lang="de" >

<HEAD>

< TITLE>Das ist multilanguage dokument</ TITLE>

</HEAD>

<BODY>

..... Текст на немецком языке .....

..... Текст на немецком языке .....

..... Текст на немецком языке .....

<P lang="es"> .... Текст на испанском..... </P>

<P>.....Текст на немецком языке..... </P>

<P>.....Текст на немецком языке..... </P>

<CITE lang="en">...цитата на английском. </CITE>
```

.....Текст	на	немецком	языке .....
.....Текст	на	немецком	языке .....
.....Текст	на	немецком	языке .....
</BODY>			
</HTML>			





**\*Примечание.\*** При использовании таблиц в HTML-документе, ячейки таблицы могут наследовать информацию о языке и направлении текста (атрибут *dir*) от первой ячейки объединения (подробности в главе, посвященной таблицам в HTML-документах).

## Направление текста и таблиц

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 элемента.

Принимаемые значения:

*LTR*..... слева направо (*dir=LTR*);

*RTL*..... справа налево (*dir=RTL*).

По умолчанию используется направленность текста слева направо.

Хотя в Unicode 2.0 определены специальные символы, отвечающие за направление текста, HTML предлагает для этих целей средства более высокого уровня. Ведь, например, чтобы в документе привести цитату на иврите, проще написать: `<CITE lang="he" dir="rtl">` цитата на иврите `</CITE>`, чем с использованием эквивалентных ссылок Unicode 2.0 *А." цитата на иврите"*

Браузеры не используют атрибут *lang* для определения направленности текста. Атрибут *dir* действует на протяжении всего элемента, включая все вложенные в него элементы.

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

*Пример*

```
<HTML dir="RTL">
```

```
<HEAD>
```

```
<TITLE>заголовок также будет выводиться слева направо</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
.... текст справа налево .
```

```
..... текст справа налево.....
```

```
<P dir="ltr"> .....текст слева направо.. </P>
```

```
<P> .... текст справа налево. </P>
```

```
..... текст справа налево.....
```

```
..... текст справа налево.....
```

```
</BODY>
```

```
</HTML>
```

## Неотображаемые символы

Набор символов документа может включать в себя множество различных неотображаемых символов. В основном, это типографские символы, используемые

некоторыми приложениями при отображении документа. В языке HTML определены только следующие неотображаемые символы:

v пробел набора символов ASCII ( )

v табуляция набора символов ASCII ( )

v **ASCII form feed** ( )

v пробел нулевой ширины (&#x200B;)

Символы &#x2028 и &#x2029, хотя и определены в спецификации [ISO 10646] как разделители строк и абзацев соответственно, в HTML не включены в категорию неотображаемых символов.

В спецификации HTML 4.01 не определена обработка символов пробелов, если они не заданы как неотображаемые символы. Поэтому

для достижения визуальных эффектов, основанных на применении неотображаемых символов вместо пробелов, рекомендуется применять таблицы стилей.

## Структурное представление текста документа

Для форматирования текста в HTML 4.01 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 (элемент CITE), фрагмент компьютерного кода (элемент CODE), выделенный текст (элемент EM), пример (элемент SAMP) и другие.

При непосредственном (физическом) форматировании визуальное свойство ставится в соответствие и характеризует непосредственно фрагмент текста. Например, элемент I выводит текст, заключенный между его начальным и конечным тегами, курсивом. Элемент 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 элементов физического форматирования, так как некоторые элементы структурного форматирования приводят к одинаковому визуальному эффекту. Например, для выделения текста курсивом предусмотрен один элемент физического форматирования I. Такого же видимого эффекта можно достичь при помощи четырех элементов структурного форматирования VAR, EM, DFN, CITE, просто они еще характеризуют смысловое содержание выделенного текста. Понимание структуры документа, а также необходимость ее четкого представления придет 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. В спецификации HTML4.01 элементы непосредственного форматирования указаны как нежелательные. Однако они поддерживаются всеми браузерами.

## Элементы структурного форматирования текста

Элемент **ABBR** — отмечает заключенный в себе текст как аббревиатуру. Этот элемент, хотя и задан в спецификации HTML 4.01, не поддерживается на настоящий момент ни одним браузером.

Элемент **Acronym** — отмечает свой текст как акроним, т.е. произносимое слово, состоящее из аббревиатур. Элемент **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.

*Например:*

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>
```

Пример использования элемента **ACRONYM** `</TITLE>`

```
</HEAD>
```

```
<BODY>
```

```
..... текст документа..... <BR>
```

```
..... текст документа..... <BR>
```

```
<BR>`` <BR>
```

```
<ACRONYM title="Санкт-Петербургский Государственный Университет Путей Сообщения"> ПГУПС  
</ACRONYM> — один из лучших технических ВУЗов страны. <BR>
```

```
<BR>`` <BR>
```

```
.....текст документа..... <BR>
```

```
.....текст документа..... <BR>
```

```
</BODY>
```



```
</HTML>
```

Элемент **CITE** — отмечает заключенный в себе текст как цитату. Браузерами такой текст отображается курсивом. Пример написания:

<CITE>Кто не работает — тот ест. Учись, студент </CITE>.

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

Элемент **DFN** — отмечает заключенный в себе текст как определение (англ. Definition) Поддерживается только браузером Internet Explorer и отображается им курсивом.

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

Элемент **KBD** — отмечает заключенный в себе текст как введенный пользователем с клавиатуры. Браузерами по умолчанию отображается моноширинным шрифтом. Является предпочтительной альтернативой элемента физического форматирования TT (выделяет текст как моноширинный).

Элемент **SAMP** — отмечает заключенный в себе текст как пример (пример отчета, выдаваемого программой, сценарием и т.п.). Браузерами по умолчанию отображается моноширинным шрифтом. Также является предпочтительной альтернативой элемента физического форматирования TT.

Элемент **STRONG** — отмечает заключенный в себе текст как сильно выделенный. Браузерами по умолчанию отображается жирным шрифтом. Является предпочтительной альтернативой элемента физического форматирования B (выделяет текст жирным шрифтом).

Элемент **VAR** — отмечает заключенный в себе текст как экземпляр переменной или

аргумента какой-либо программы. Браузерами по умолчанию отображается курсивом.

Из описания этих элементов видно, что некоторые из них приводят к одинаковому визуальному результату: выделение текста курсивом, например, можно осуществить элементами VAR, EM, DFN, CITE. Однако не только это является результатом их применения. Использование элементов структурного форматирования 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. Кроме того, с развитием Web-технологий и браузеров скоро станет возможно более интенсивное использование этих элементов (например, поиск в сети отчетов выполнения определенной программы).

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

*Пример использования:*

<HTML>

<HEAD>

<TITLE>

Использование элементов структурного форматирования текста

</TITLE>

</HEAD>

<BODY bgcolor=white> Обычная строка

<BR>` `<BR>

<CITE>Строка, отформатированная элементом CITE</CITE>

<BR>

<CODE>Строка, отформатированная элементом CODE</CODE>

<BR>

<DFN>СТроКа, отформатированная элементом DFN</DFN>

<BR>

<EM>Строка, отформатированная элементом EM </EM>

<BR>

<KBD>Строка, отформатированная элементом KBD </KBD>

<BR>

<SAMP>Строка, отформатированная элементом SAMP </SAMP>

<BR>

<STRONG> Строка, отформатированная элементом STRONG

</STRONG>

<BR>

<VAR> Строка, отформатированная элементом VAR </VAR>

<BR>

</BODY>

</HTML>

Все элементы структурного форматирования имеют следующие необязательные атрибуты, которые могут быть заданы или не заданы:

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.



Элементы INS и DEL используются для разметки вставленных и удаленных фрагментов текста документа, по отношению к предыдущей версии документа. То есть эти два элемента характеризуют свое содержимое либо как "вставленное" (INS), либо как "удаленное" (DEL). Элементы INS и DEL занимают обособленное положение среди всех остальных HTML-элементов. Связано это с тем, что они могут выступать и в качестве элементов уровня блока, и в качестве встроенных элементов, но не теми и другими сразу. То есть элементы INS и DEL не должны содержать элементы уровня блока, если они используются как встроенные.

*Пример правильного использования\*\*я:*

<P>

По последним сведениям состояние Билла Гейтса составляет <DEL> 97 </DEL>` `<INS> 90 </INS> миллиардов долларов

</P>

### *Пример неправильного использования:*

<P>

<INS>` `<DIV> текст ..</DIV>` `</INS>

</P>

Текст, отмеченный как "удаленный", отображается браузером перечеркнутым (рис. 3.41). Элемент DEL является предпочтительной альтернативой элементов физического форматирования STRIKE и S (обозначают текст как перечеркнутый).

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

#### **\*Пример (РИС. 3.41)\***

<HTML>

<HEAD>

<TITLE>Использование элементов INS и DEL</TITLE>

</HEAD>

<BODY bgcolor=white> Обычная строка


<BR>` `<BR>

<INS>Строка, отформатированная элементом INS</INS><XBR>

<DEL>Строка, отформатированная элементом DEL</DEL>` `<BR>

</BODY>

</HTML>

\*Примечание.\* Элементы *INS* и *DEL* пока распознаются только браузером *Microsoft Explorer*.

Эти элементы имеют следующие необязательные атрибуты:

*Cite=URL* — указывает месторасположение документа с информацией, объясняющей причину изменения документа;

*Datetime* — указывает дату изменения документа. Информация, задаваемая этими атрибутами, браузерами никак не отображается.

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:

<HTML>

<HEAD>

</HEAD>

<BODY>

.... текст документа .....

.... текст документа .....

<INS datetime= "1999-11-05T08:15:30-05:00" cite=

<http://www.pcmag.ru/prognoz2001.html> lang="ru">

Согласно прогнозам журнала "PC Magazine", после 2001 года высокоскоростная технология DSL найдет более широкое применение в корпорациях, нежели в среде домашних пользователей. </INS>

.... текст документа .....

.... текст документа .....

</BODY>

</HTML>

### Элемент *PRE*

Наличие этого элемента 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:*

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>Отрывок стихотворения В.В.Маяковского</ TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
Отрывок стихотворения В.В.Маяковского
```

---

Крошка сын

к отцу пришел, и спросила кроха: - Что

такое

```
</PRE>
```

```
</BODY>
```

```
</HTML>
```

такое

хорошо И что плохо?

На экране это будет выглядеть, как показано на рис. 3.42.





**\*Примечание.\*** Элемент *HR* прорисовывает прямую горизонтальную линию.

При использовании элемента *PRE* задание начального и конечного тегов является обязательным. *Атрибуты:*

**Width** — указывает браузеру желательную ширину блока форматированного текста. Ширина задается количеством символов.

**\*Примечание.\*** На данный момент этот атрибут браузерами не поддерживается.

*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.

Содержимое элемента *PRE* может содержать в себе встраиваемые элементы, кроме следующих: *IMG*, *APPLET*, *OBJECT*, *FONT*, *BASEFONT*, *BIG*, *SMALL*, *SUP*, *SUB*.

## Элементы физического форматирования документа

Элементы физического форматирования непосредственно определяют вид текста в окне браузера. По указанным выше причинам многие из них в спецификации HTML 4.01 объявлены как нежелательные, но все они поддерживаются браузерами, широко используются и долго еще будут использоваться, потому что удобны.

**Элемент B** — отображает заключенный в себе текст жирным шрифтом. Вместо него рекомендуется использовать элемент структурного форматирования *STRONG*.

**Элемент I** — отображает заключенный в себе текст курсивом. Является нежелательным. Вместо него рекомендуется использовать элементы структурного форматирования *VAR*, *EM*, *DFN*, *CITE*, которые, помимо выделения текста курсивом, указывают его смысловое значение.

**Элемент TT** — отображает заключенный в себе текст моноширинным шрифтом. Является нежелательным. Вместо него рекомендуется использовать элементы структурного форматирования *CODE*, *SAMP* и *KBD*, которые указывают также смысловое значение текста.

**Элементы STRIKE и S** — отображает заключенный в себе текст перечеркнутым. Определены в спецификации HTML 4.01 как нежелательные. Однако ввиду того, что предлагаемый вместо них элемент *DEL* поддерживается не всеми браузерами, можно порекомендовать одновременное использование этих элементов. Например:

`<DEL>``<STRIKE>..... перечеркнутый текст..... <STRIKE/>``</DEL>`

На практике обычно используется один элемент STRIKE (или S), безо всяких хитростей.

**Элемент BIG** — отображает заключенный в себе текст шрифтом, большим, чем шрифт окружающего текста. Браузеры поддерживают вложение не- нескольких элементов BIG друг в друга, но рекомендуется использо- вать для этих целей элементы заголовков **H1, H2, , H6**.

**Элемент SMALL** — отображает заключенный в себе текст шрифтом, меньшим, чем шрифт окружающего текста.

**Элемент SUB** — выводит заключенный в себе текст в нижнем индексе.

**Элемент SUP** — выводит заключенный в себе текст в верхнем индексе.

Элементы SUR и SUB обычно используются для вывода математи- ческих и химических формул.

*Пример использования (рис. 3.43)*

<HTML>

<HEAD>

<TITLE>

Использование элементов физического форматирования текста

</TITLE>

</HEAD>

<BODY bgcolor=white> Обычная строка

<BR>``<BR>

<B>Строка, отформатированная элементом B</BXBR>

<I>Строка, отформатированная элементом I</I>``<BR>

<TT>Строка, отформатированная элементом TT</TTXBR>

<STRIKE>СТроКа, отформатированная элементом STRIKE </STRIKE>``<BR>

<BIG>Строка, отформатированная элементом BIG</BIGXBR>

<SMALL>Строка, отформатированная элементом SMALL

</SMALL>``<BR>

Обычный текст <SUP>ТеКСТ в верхнем индексе </SUP>``<BR> Обычный текст <SUB>ТеКСТ в нижнем индексе </SUB>``<BR>

</BODY>

</HTML>

Все элементы физического форматирования имеют следующие 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. 3.43. Использование элементов физического форматирования текста

## Шрифты

### *Элемент FONT*

Для задания таких параметров шрифта, как тип шрифта, размер шрифта и цвет шрифта применяется элемент FONT. Он относится к элементам физического форматирования. В спецификации **HTML 4.01** указан как нежелательный, ввиду нарушения принципа разделения содержательной и описательной частей документа. Вместо него рекомендуется использовать таблицы стилей. Тем не менее, элемент FONT поддерживается всеми браузерами и его применение является довольно удобным. Начальный и конечный теги обязательны.

*Атрибуты (все необязательные)*

**Face** — указывает тип шрифта, которым браузер будет выводить текст документа. В качестве значения данного атрибута выступают полные названия шрифтов. Может быть задано несколько названий шрифтов через запятую. В этом случае браузер начинает с первого по списку шрифта и, если не удастся его подключить, переходит к следующему. Если браузер не смог подключить ни один из указанных атрибутом шрифтов, то он использует шрифт, установленный по умолчанию. Пример использования:

```
<FONT face="Complex", "Arial", "Times New Roman">
```

..... текст документа.....

..... текст документа.....

</FONT>

**Size** - - указывает размер шрифта в условных единицах: от 1 до 7. Шрифт используемый браузерами по умолчанию соответствует значению 3. Атрибут Size допускает относительное свое задание, например: size=+2. В этом случае размер шрифта увеличится на 2 условные единицы по сравнению с размером шрифта, принятого по умолчанию (рис. 3.44). Относительное задание обычно применяется в сочетании с использованием элемента BASEFONT, который задает базовый размер шрифта.

**Color** — задает цвет шрифта. В качестве значения принимается либо название цвета, либо его шестнадцатичный код ( RGB). Таблица цветов с их названиями и кодам RGB имеется на прилагаемом к книге CD.

Если при использовании элемента FONT не задан ни один из атрибутов, то его наличие игнорируется.

*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.



## Элемент BASEFONT

Элемент BASEFONT применяется для задания типа, размера и цвета шрифта, используемого браузером по умолчанию. Указанные им параметры шрифта используются по всему тексту документа, следующему после объявления самого элемента BASEFONT.

Значения параметров, установленные этим элементом, могут быть временно переопределены элементом FONT, и восстанавливают свое значения сразу после конечного тега последнего.

 **Примечание.\*** Установка базового размера шрифта не применяется к заголовкам (элементы H1, ..., H6), если они не изменены с помощью элемента FONT указанием относительного размера шрифта.

Элемент BASEFONT может использоваться несколько раз. Причем может задаваться и в заголовке HTML-документа и в его теле.

Начальный тег элемента BASEFONT обязателен, конечный тег запрещен. Атрибуты данного элемента и их задание абсолютно идентичны атрибутам элемента FONT.



**\*Примечание.\*** Браузер *Netscape Communicator* не поддерживает использование атрибута *face* для элемента *BASEFONT*.

*Пример использования:*

<HTML>

<HEAD>

<TITLE>

Пример использования элементов FONT и BASEFONT

</TITLE>

</HEAD>

<BODY>

.....текст документа, выводимый .....

..... черным шрифтом Times New Roman (по умолчанию) .....

<BASEFONT color=blue face="Arial">

.....текст документа.....

..... выводимый синим шрифтом Arial .....

<FONT color=black size=+2>

.....текст документа.....

. ...выводимый черным шрифтом Arial большего размера.. .

</FONT>

..... текст документа.....

..... выводимый синим шрифтом Arial нормального размера ....

</BODY>

</HTML>

## *Элементы Q и BLOCKQUOTE*

Элементы Q и BLOCKQUOTE задуманы как элементы, выделяющие в цитаты, приводимые в тексте документа:

**Q** — короткие цитаты (нет разбиения на абзацы);

**BLOCKQUOTE** - длинные цитаты.

Элемент BLOCKQUOTE, в отличие от элемента 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.

Элемент Q выводит заключенный в себе текст в кавычках. На данный момент поддерживается только браузером Internet Explorer 5.5 и выше.

Для них обоих начальный тег обязателен, конечный тег для BLOCKQUOTE не обязателен, для Q обязателен. При незаданном конечном теге BLOCKQUOTE цитатой считается весь следующий за начальным тегом текст.

*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 элемента BLOCKQUOTE, сверху и снизу отделяется пустыми строками и выводится с небольшим отступом влево при

направленности текста слева направо (рис. 3.43). Допускается вложение элементов BLOCKQUOTE друг в друга, чем достигаются дополнительные возможности форматирования документов (управление отступами). Пример использования (рис. 3.45):

<HTML>

<HEAD>

<TITLE>Использование элемента BLOCKQUOTE</TITLE>

</HEAD>


<BODY bgcolor=white>

<BR>

обычный текст...обычный текст... обычный текст<BR> обычный текст...обычный текст... обычный текст<BR>  
обычный текст...обычный текст... обычный текст<ER> обычный текст...обычный текст... обычный текст<BR>

текст, отформатированный элем. BLOCKQUOTE<BR> текст, отформатированный элем. BLOCKQUOTE<BR> текст,  
отформатированный элем. BLOCKQUOTE<BR> текст, отформатированный элем. BLOCKQUOTE<BR> текст,  
отформатированный элем. BLOCKQUOTE<BR>

</BLOCKQUOTE>

обычный текст...обычный текст... обычный TeKCT<BR> обычный текст...обычный текст ....обычный  
TeKCT<BR> обычный текст...обычный текст .... обычный TeKCT<BR> обычный текст...обычный текст .... обычный  
TeKCT<BR>

</BODY>

</HTML>

## **Работа со строками и абзацами**

Текст документа обычно разбивается авторами на абзацы, имеющие 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 и длина строк. Все это может привести к большим неудобствам. Чтобы их избежать в языке HTML предусмотрено разбиение текста на отдельные абзацы путем заключения их в содержимое элементов P.

## Элемент P

HTML располагает своими средствами для разбиения на абзацы. Для этой функции используется элемент уровня блока P. Текст, находящийся

между начальным и конечным тегами этого элемента, 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. В элементе P могут содержаться вложенные элементы уровня блока, включая и сам элемент P.

*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.

**\*Элемент BR\***



Место переноса строки в пределах самого 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 элемент 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 при- крепляемые объекты (таблицы, изображения и т.п.).

*Принимаемые значения:*

**\*None\*** — следующая строка текста отображается обычным образом, на ближайшем свободном снизу пространстве. Это значение ис- пользуется 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 соот- ветствующим заданием у них атрибута align. Тогда если объект при- креплен, например, к левому полю, то он обтекается текстом по правой своей стороне.

*Пример использования*

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. 3.47):

```
<BODY bgcolor=white> <H2 align=center>Мухаммед Али</H2> <IMG src="alil.jpg">
```

АЛИ МУХАММЕД (Ali Muhammad) (наст. имя и фам. Кассиус Клей, Clay) (р.17.01.1942) , американский спортсмен (бокс). Чемпион Олимпийских игр (1960) в полутяжелом весе. Неоднократный чемпион мира среди профессионалов (в 1964-74) в тяжелом весе. На XVII Олимпийских играх в Риме в 1/4 финала победил по очкам советского выдающегося боксера Геннадия Шаткова, а в финале победил поляка З.Петшиковского и стал Олимпийским чемпионом При весе 97 кг и росте 192 см

Мухаммед Али на ринге был необычайно легок и подвижен. Ему принадлежит фраза: "Пор хаю, как бабочка и жалю, как пчела". В течении 20 лет его профессиональной карьеры он был законодателем мод на ринге. Его редкие поражения, а их было всего 5, считались случайностью. В целом, Мухаммед Али провел 25 титульных или отборочных к ним боев, и это стало бы рекордом, не будь Джо Луиса, у которого таковых на один больше. А первое поражение он испытал в марте 1971 года, в Нью-Йорке от Джо Фрэзера, которого за ярко выраженный силовой стиль звали "черным Марчиано".

```
</BODY>
```

```
</HTML>
```



Теперь организуем обтекание изображением текстом, прикрепив его к левому краю. Для этого изображению укажем атрибут align=left, то есть в HTML-коде документа строку <IMG src="alil.jpg"> (вставка картинки alil.jpg в документ) заменим на строку <IMG src="alil.jpg" align=left>. В результате чего получим текст, обтекающий изображение (рис. 3.48).



Применение элемента 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 элемента BR без указания атрибута clear (что равносильно указанию clear=none) не прерывает обтекания текстом, а только осуществляет перенос на следующую строку. Вставим элемент <BR> после предложения "Неоднократный чемпион мира среди профессионалов (в 1964-74) в тяжелом весе" (рис. 3.49).



Использование элемента BR с атрибутом clear=left прервет обтекание текстом и продолжит вывод текста сразу под прикрепленным объектом (рис. 3.50).



### *Запрет перехода на новую строку*

Иногда разработчикам требуется избежать перехода на новую строку между двумя определенными словами. Комбинация символов ( или &#xA0;) действует как неразрывный пробел.



Чтобы запретить переходы на следующую строку в рамках какого-либо фрагмента текста, используется HTML-элемент NOBR. Текст, заключенный в его содержимом, гарантировано будет отображен одной строкой.

*ейки т*

**\*Примечание.\*** Для запрета переноса строки в рамках ячейки таблицы предусмотрен логический атрибут NOWRAP элемента ячейки таблицы TD.

В языке HTML предусмотрены элементы, определяющие содержащуюся 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 элементы H1, H2, H3, H4, H5, H6. Все заголовки отображаются жирным шрифтом (рис. 3.51).

Элементы 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 этот заголовок. Весь остальной текст располагается выше и ниже его.

Заголовки сверху и снизу выделяются пустыми строками.

При использовании элементов 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 задания (рис. 3.51)*

<HTML>

<HEAD>

</HEAD>


<BODY bgcolor=white> 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>		
		

#### Вставка горизонтальных линий


В HTML реализована возможность рисования 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 элемента HR. Визуальные параметры этой линии задаются с помощью атрибутов элемента 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** — этим атрибутом указывается цвет линии в виде ключевого слова, обозначающего цвет, или в виде RGB-кода.

**\*Примечание.\*** Браузер *Netscape Communicator* не поддерживает использование атрибута *color* для HTML-элемента *HR*.

Пример (рис. 3.52)

```
<HTML>
<HEAD>
<TITLE>Горизонтальные линии</TITLE>
</HEAD>
```

<BODY>

## Примеры горизонтальных линий HR

HR noshade

HR size=20

HR size=20 noshade

HR size=45

HR width=50%

HR width=50% color=black

HR width=50% color=black size=10

HR width=50% color=black size=10 align=left

</BODY>

</HTML>



## 2.4

Язык HTML обладает возможностями предоставления информации в виде списков. Список служит для добавления структуры в документ. Он позволяет в более удобном и понятном виде представить информацию, такую как, например: список покупок, пошаговое описание каких-либо действий, толковый словарь и т.п.

В HTML различают: \* неупорядоченные списки (список покупок);

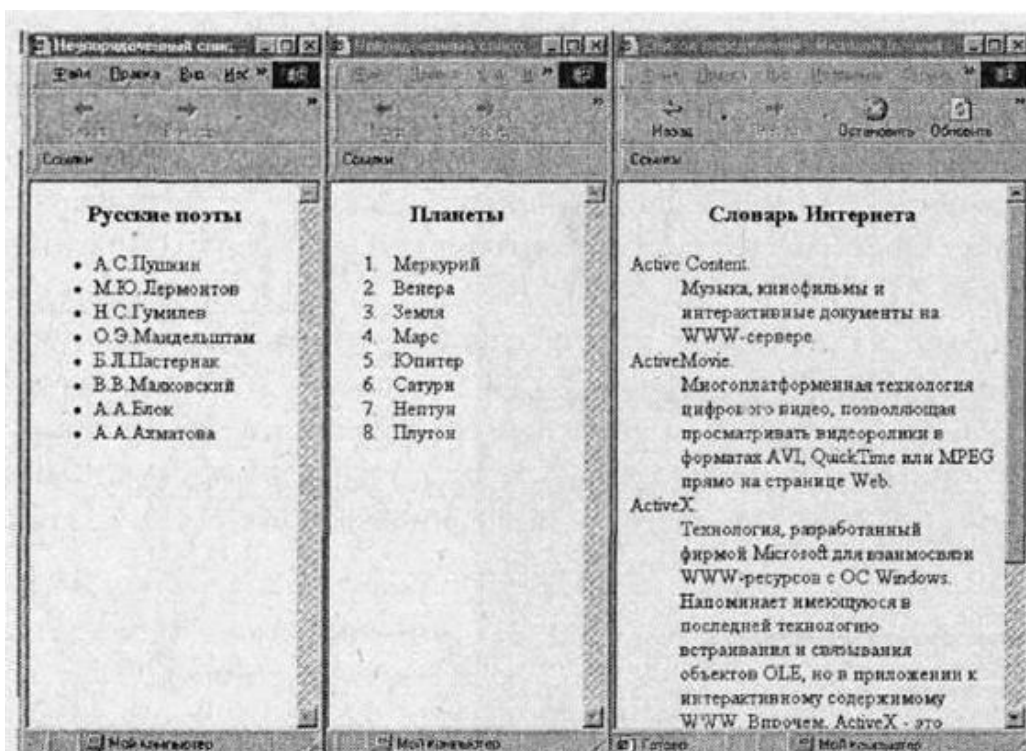
\* упорядоченные списки (пошаговое описание, в котором каждый шаг пронумерован);

\* список определений (толковый словарь).

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

*Неупорядоченные (UL) и упорядоченные (OL) списки и их элементы (LI)*

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




**Рис. 3.53.** Примеры списков всех трех видов. Слева направо неупорядоченный список, упорядоченный и список определений

*Неупорядоченные списки. Элемент UL*

Задание начального и конечного тегов элемента UL является обязательным.  
(все необязательные):

Атрибуты

**Type** — задает информацию о виде используемых маркеров. Может принимать следующие значения: **\*Circle\*** — маркеры отображаются в виде незакрашенных 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).

**Compact** — присутствие этого атрибута указывает браузеру на то, что данный список он должен отображать более компактно (например, уменьшив межстрочковый интервал).

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

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

открывающего тега LI или до закрывающего тега UL (OL).

Пример (рис. 3.54)

```

<TITLE>Пример задания неупорядоченного списка</TITLE> </HEAD> <BODY><P>

<STRONG>Николай Степанович Гумилёв</STRONG> — один из величайших поэтов

<EM>"Серебрянного века"</EM>. Им было создано новое поэтическое направление — акмеизм.

</P>

<UL id="poem_list">

<H4>Стихотворения из цикла "Романтические цветы"</H4> <LI>Мечты <LI>Перчатка

<LI>Заклинание <LI>Озеро Чад <LI>` ` <EM>Сады души *</EM> <LI>Основатели

<H4>Стихотворения из цикла "Жемчуга"</H4> <LI>Потомки Каина <LI>Жамень

<LI>Старый конкистадор <LI>Варвары <UL>

<P>

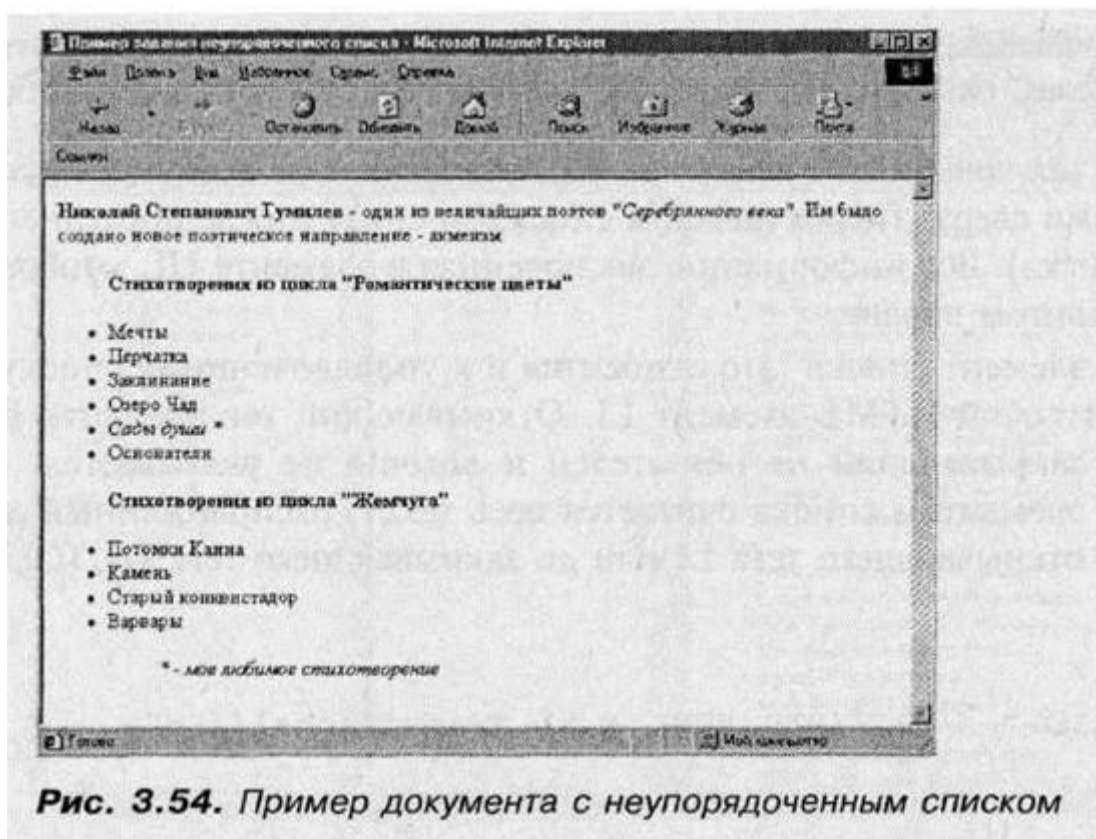
<EM>*</EM> — мое любимое стихотворение</EM>

</p>

</BODY>

</HTML>

```





*Для элемента LI определены такие же стандартные атрибуты, что и для элемента UL, и еще два необязательных атрибута value и type. Эти атрибуты используются при работе с упорядоченными списками и будут рассмотрены позже. Визуально функции элемента LI сводятся к отображению маркера в неупорядоченных списках или нумерации в упорядоченных списках.*

Допускается использование вложенных неупорядоченных списков (рис. 3.55). При отображении вложенных неупорядоченных списков браузеры по умолчанию используют различную маркировку списков разного уровня.

Например (рис. 3.55)

```
<HTML>
<HEAD>
</HEAD>
<BODY bgcolor=white>
```

## Российские футбольные клубы и их лучшие игроки

```
<LI>Санкт-Петербург
```

```
<LI>"Зенит":
```

```
<LI>А.Панов
```

```
<LI>А.Куртиян
```

```
<LI>Р.Березовский
```

```
<LI>С.Оганисян
```

```
</UL>
```

```
</UL>
```

```
<LI>Москва
```

```
<LI>"Спартак" :
```

```
<LI>Е.Титов
```

```
<LI>В.Ананко
```

```
<LI>Робсон
```

```
<LI>А.Тихонов
```

```
</UL>
```

<LI>Локомотив:

<LI>А.Чугайнов

<LI>З.Джанашия

<LI>А.Булыкин

< LI>Е.Харлачев

</UL>

<LI>"ЦСКА":

<LI>В.Кулик

<LI>В.Хомуха

<LI>С.Семак

<LI>Е.Варламов

</UL>

</UL>

<LI>Волгоград:

<LI>"Ротор"

<LI>И. Веретенников

<LI>В.Краснов

<LI>В.Есипов

<LI>Д .Зубко

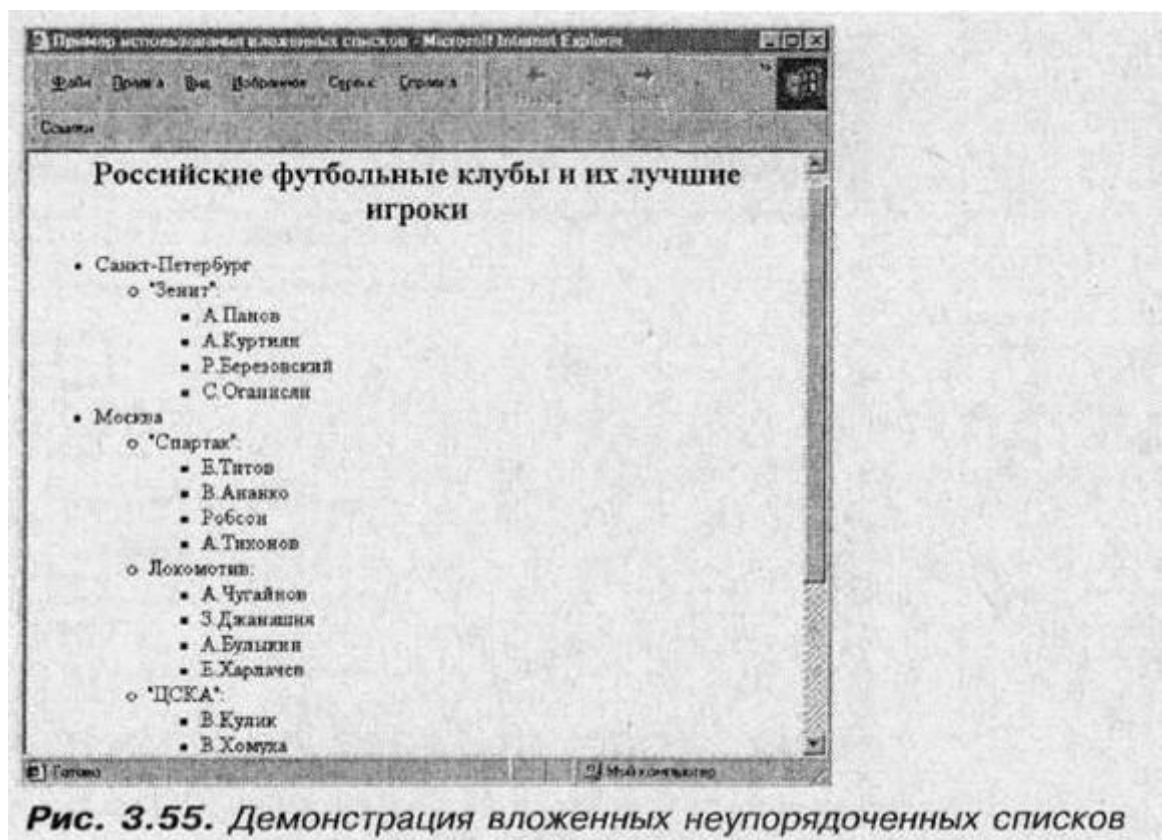
</UL>

</UL>

</UL>

</BODY>

</HTML>



**Рис. 3.55.** Демонстрация вложенных неупорядоченных списков

С помощью атрибута TYPE элемента UL можно непосредственно указывать вид пульки (маркера). Например (рис. 3.56):

```
<HTML>
```

```
<HEAD>
```

```
</HEAD>
```

```
<BODY>
```

```
<LI>Маркер — незакрашенный кружок
```

```
<LI>Маркер — незакрашенный кружок
```

```
<LI>Маркер — незакрашенный кружок
```

```
<LI>Маркер — незакрашенный кружок
```

```
</UL>
```

```
<LI>МарКер — закрашенный квадратик
```

```
<LI>МарКер — закрашенный квадратик
```

```
<LI>МарКер — закрашенный квадратик
```

```
<LI>МарКер — закрашенный квадратик
```

</UL>

<LI>МарКер — закрашенный кружок

<LI>МарКер — закрашенный кружок

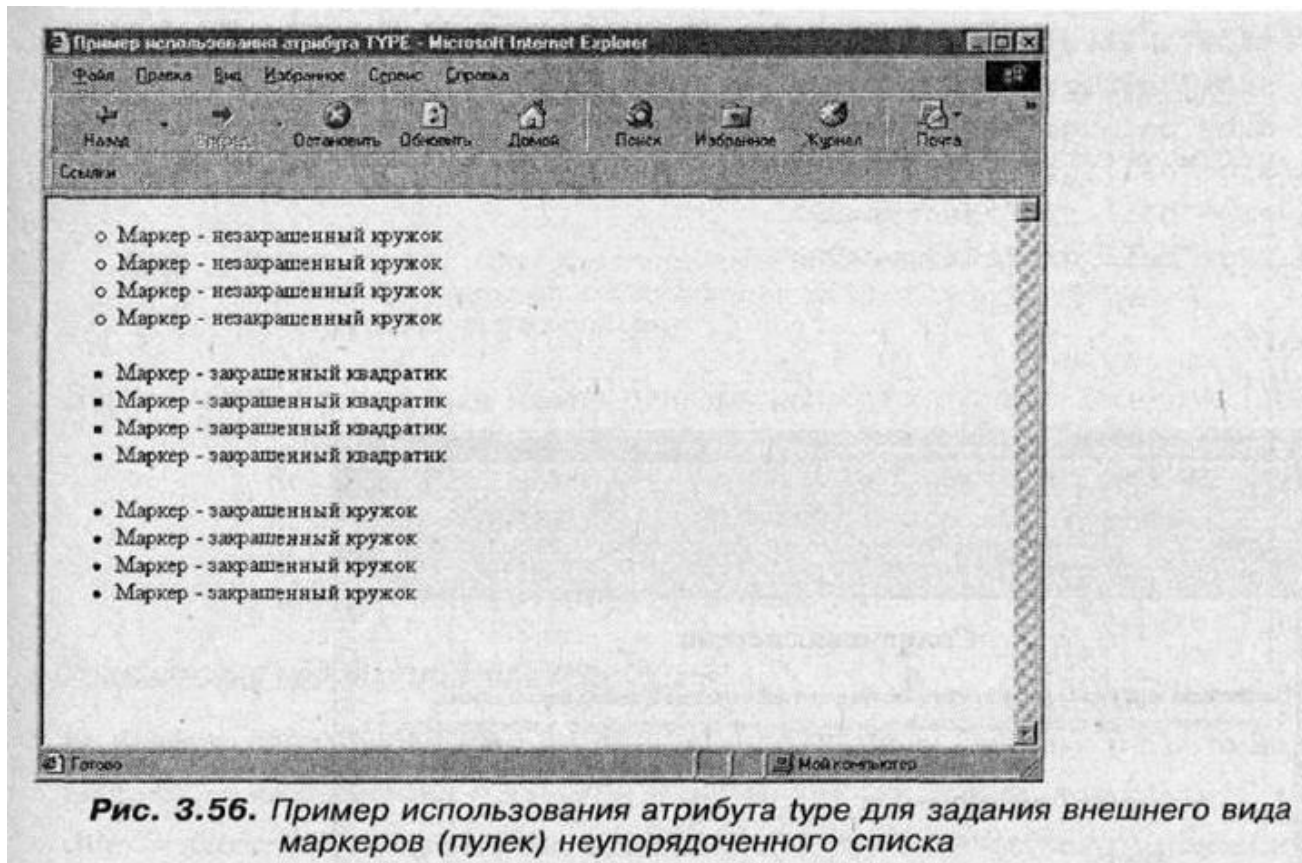
<LI>МарКер - закрашенный кружок

<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 элемента LI для каждого элемента списка используются элемент IMG — вставки изображения маркера и элемент BR — перехода на новую строку.

Пример (РИС. 3.57)

<HTML>

<HEAD>

<TITLE>Пример списка с нестандартными маркерами

</TITLE>

</HEAD>

<BODY bgcolor=white>

## Солнечная система

<STRONG>Солнечная система</STRONG> состоит из центрального светила — Солнца и девяти планет, обращающихся вокруг него, их спутников, множества малых планет, комет и межпланетной пыли.

</P>

<H4>Список планет Солнечной системы</H4>

<IMG src="ball.gif">Меркурий<BR>

<IMG src="ball.gif">Венера<BR>

<IMG src="ball.gif">Земля<BR>

<IMG src="ball.gif">Марс<BR> <IMG src="ball.gif">Юпитер<BR> <IMG src="ball.gif">Сатурн<BR> <IMG src="ball.gif">Уран<BR> <IMG src="ball.gif">Нептун<BR>

<IMG src="ball.gif">Плутон<BR> </UL> </BODY> </HTML>

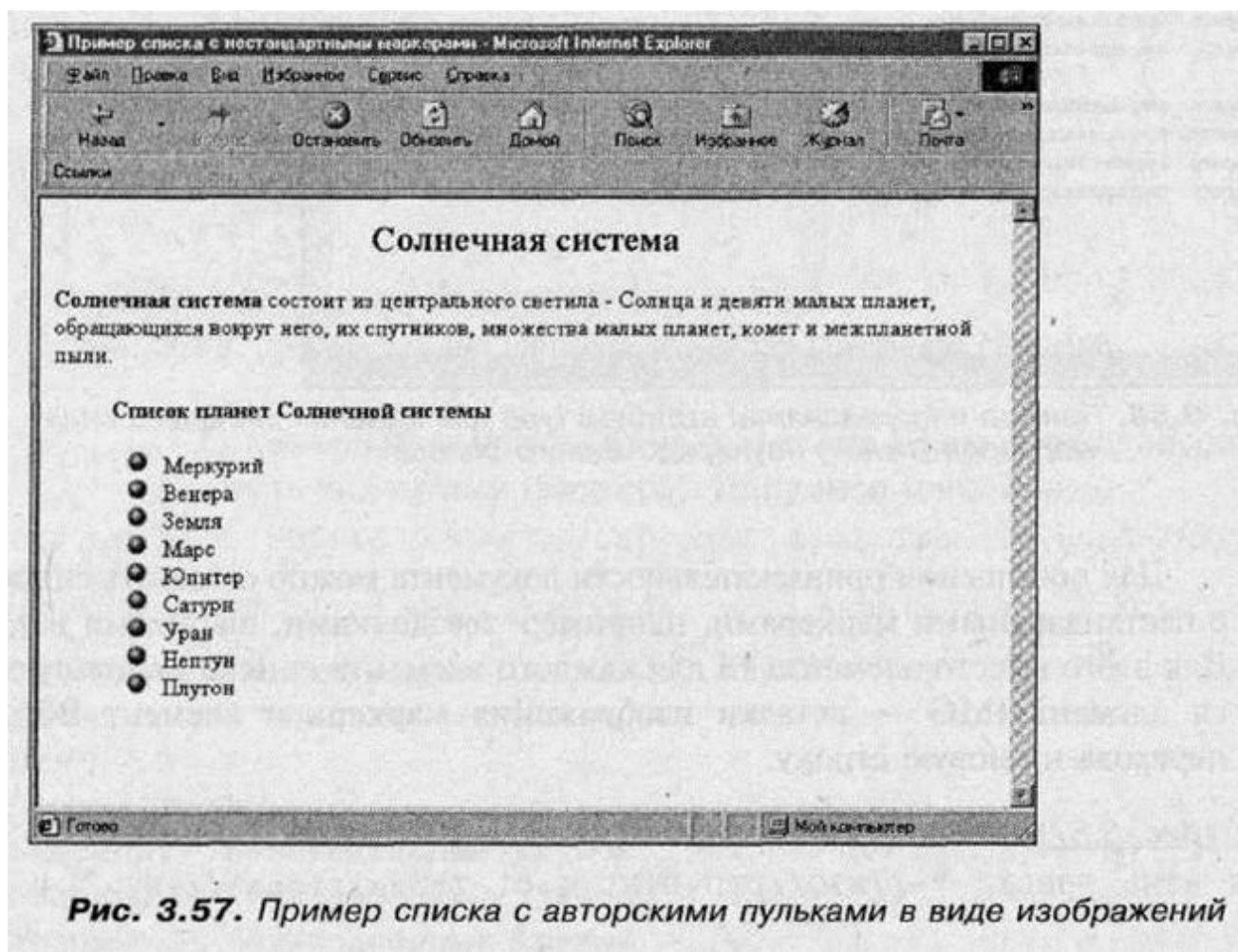


Рис. 3.57. Пример списка с авторскими пулями в виде изображений

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

Указание начального и конечного тегов элементов OL является обязательным.

*Атрибуты (все необязательные):*

**Type** — указывает вид нумерации элементов упорядоченного списка. Может принимать следующие значения:

**Type** = *I* — 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 римскими цифрами.*

**\*Примечание.\*** При использовании вложенных упорядоченных списков браузеры по умолчанию не делают разной

нумерации



списков различного уровня.

**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 элемента можно менять непосредственно при задании элемента списка LI. В этом случае используется вышеописанный атрибут Type и атрибут value, задаваемые для элемента 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 атрибут start элемента **OL**.

*Пример (РИС. 3.58)*

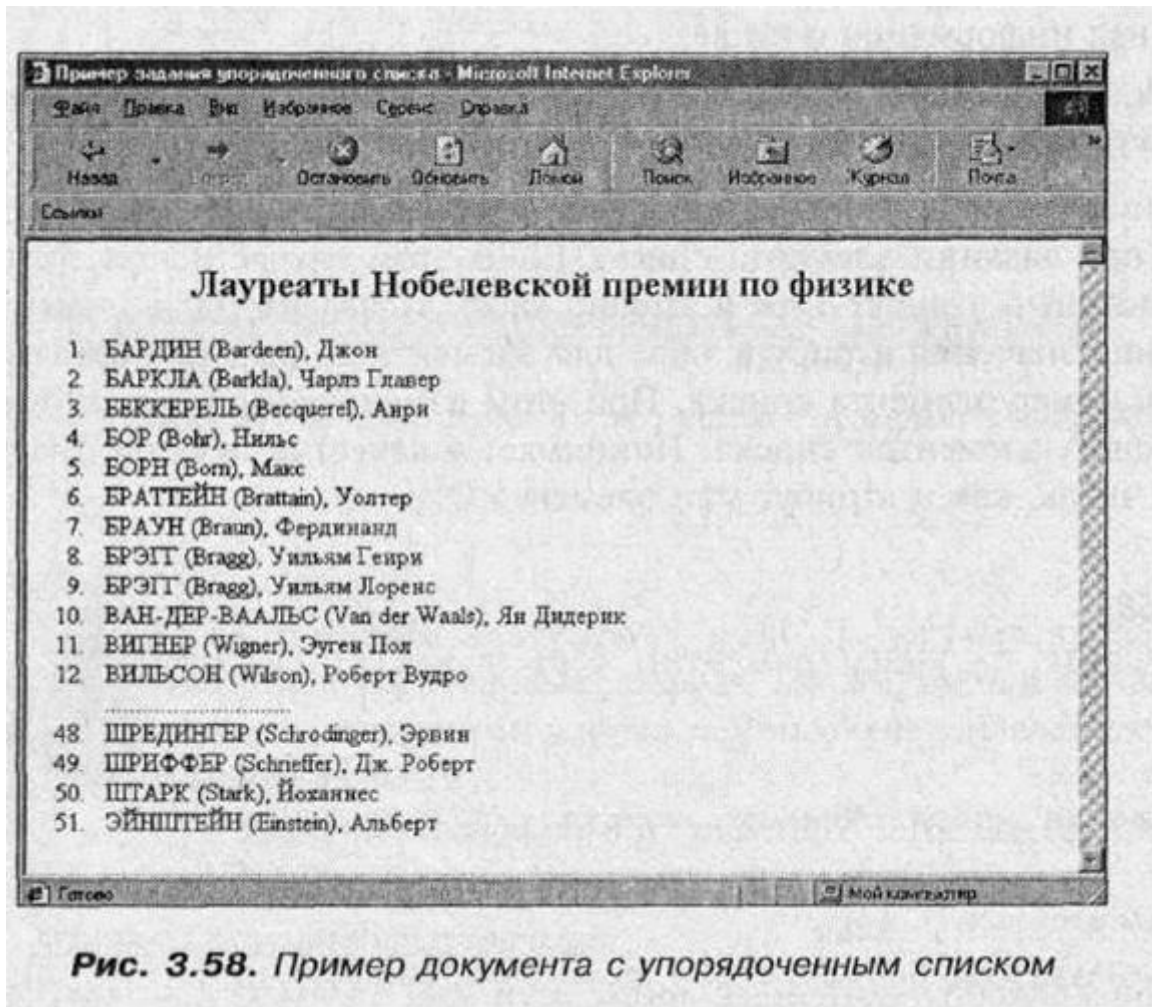
```
'  
<HTML>  
  
<HEAD>  
  
</HEAD>  
  
<BODY bgcolor=white>
```

## Лауреаты Нобелевской премии по физике

```
<LI>БАРДИН (Bardeen), Джон  
<LI>БАРКЛА (Barkla) , Чарлз Главер  
<Ы>БЕККЕРЕЛЬ (Becquerel), Анри  
<LI>БОР (Bohr) , Нильс  
<LI>БОРН (Born), Макс  
<LI>БРАТТЕЙН (Brattain) , Уолтер  
<LI>БРАУН (Braun) , Фердинанд  
<LI>БРЭГГ (Bragg) , Уильям Генри  
<LI>БРЭГГ (Bragg) , Уильям Лоренс  
<LI>ВАН-ДЕР-ВААЛЬС (Van der Waals), Ян Дидерик  
<LI>ВИГНЕР (Wigner), Эуген Пол  
<LI>ВИЛЬСОН (Wilson) , Роберт Вудро  
<BR> .....  
<LI value = 48>ШРЕДИНГЕР (Schrodinger) , Эрвин  
<LI>ШРИФФЕР (Schrieffer) , Дж. Роберт  
<LI>ШТАРК (Stark) , Йоханнес  
<LI>ЭЙНШТЕЙН (Einstein) , Альберт  
</OL>
```

</BODY>

</HTML>



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

Например (рис. 3.59)

<HTML>

<HEAD>

</HEAD>

<BODY bgcolor=white>

## Нобелевские лауреаты

<LI>` ` <B>Премия по химии</B>

<LI>АРРЕНИУС (Arrhenius) , Сванте



<LI>АСТОН (Aston) , Фрэнсис Уильям

<LI>БЕРГИУС (Bergius), Фридрих

<BR> .....

<LI value=30>ФИШЕР (Fischer) , Эмиль

<LI>ЭЙЛЕР-ХЕЛЬПИН (Euler-Chelpin) , Ханс фон

<LI>ЮРИ (Urey) , Гарольд К.

</OL>

<LI>` ` <B>Премия по литературе</B>

<LI>ВЕНАВЕНТЕ-И-МАРТИНЕС (Benavente y Martinez), Хасинто

<LI>БЕРГСОН (Bergson), Анри

<LI>БЬЕРНСОН (Bjernson), Бьеристерне

<LI>ГАМСУН (Hamsun), Кнут

<BR> .....

<LI value=2 9>ШОУ (Shaw), Джордж Бернард

<LI>ШПИТТЕЛЕР (Spitteler), Карл

<LI>ЭЙКЕН (Eucken) , Рудольф

<LI>ЭЧЕГАРАЙ (Echegaray) , Хосе

</OL>

<LI>` ` <B>Премия мира</B>

<LI>АССЕР (Asser), Тобиас

<LI>АРНОЛЬДСОН (Arnoldson) , Клас

<LI>БАЙЕР (Bajer), Фредрик

<LI>БЕЕРНАР (Beernaert), Огюст

<BR> .....

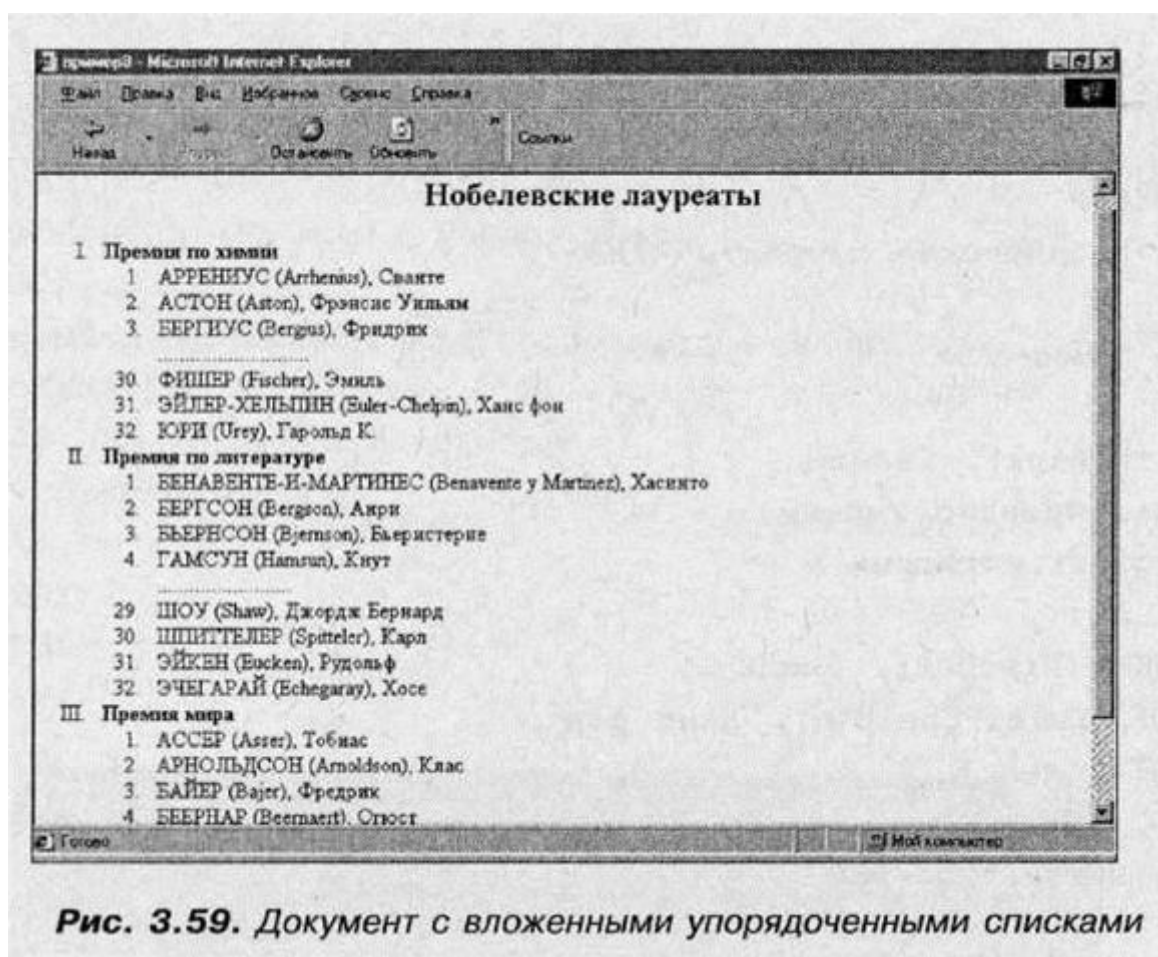
<LI value=25>САХАРОВ, Андрей <LI>ФРИД (Fried), Альфред

</OL>

</OL>

</BODY>

</HTML>



**Рис. 3.59.** Документ с вложенными упорядоченными списками

В этом примере также проиллюстрировано использование атрибута value элемента 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.

В качестве примера приведен отрывок кода документа, изображенного на рис. 3.62.

Например:

```
<OL type=A>
```

```
<LI>Краткая
```

```
биография
```

```
Карла
```

```
Великого
```

<LI>Карта Франкского королевства в X веке

<BR>.....

<LI value=5> Хронология развития Византийского государства

<LI>Карта Византийского государства в X веке

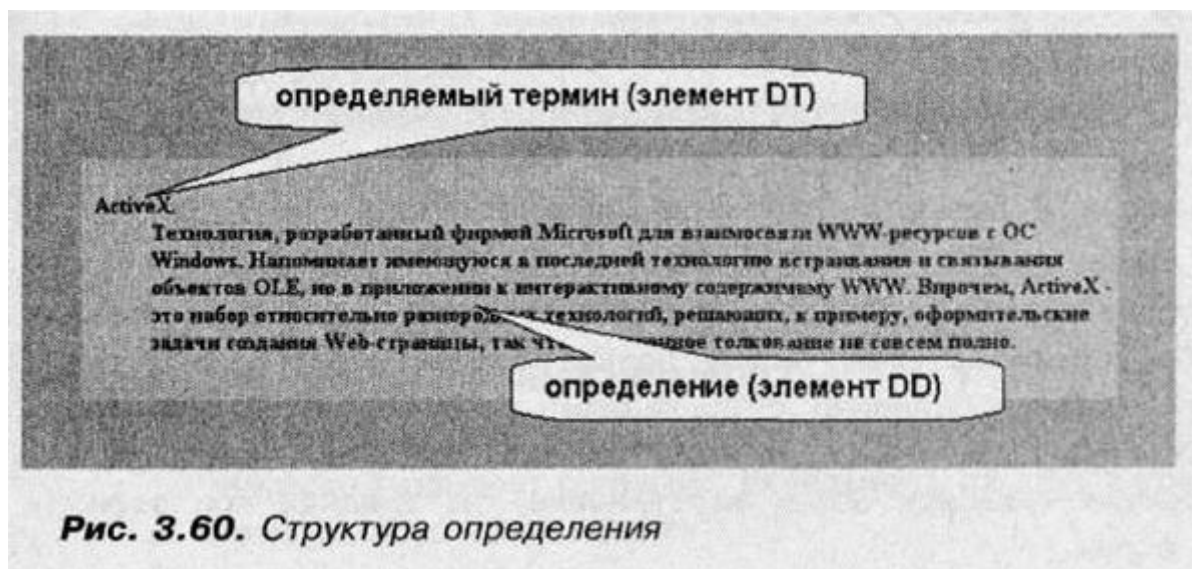
</OL>

### Списки определений

Списки определений являются особым видом списков, которые представляют содержащуюся в себе информацию в виде словарных статей. Сначала указывается определяемый термин, а затем приводится его определение, отображенное с отступом. Таким образом, каждый элемент такого списка состоит из двух частей (рис. 3.60):

\* определяемого термина, задаваемого с помощью элемента DT (Definition Term);

\* текста с его определением, задаваемого с помощью элемента DD (Definition Description). Сам список определений заключается в содержимом элемента-контейнера DL (Definition List).



(

Элементы 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.

Для элемента DL допустимо задание атрибута **compact**, дающего указание браузеру отображать список более компактно.

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

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

*Пример использования списка определений (рис. 3.61)*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML> <HEAD>
```

```
<BODY bgcolor=white>
```

## Математический справочник

---

```
<DT>` ` <B>Делитель нуля.</B>
```

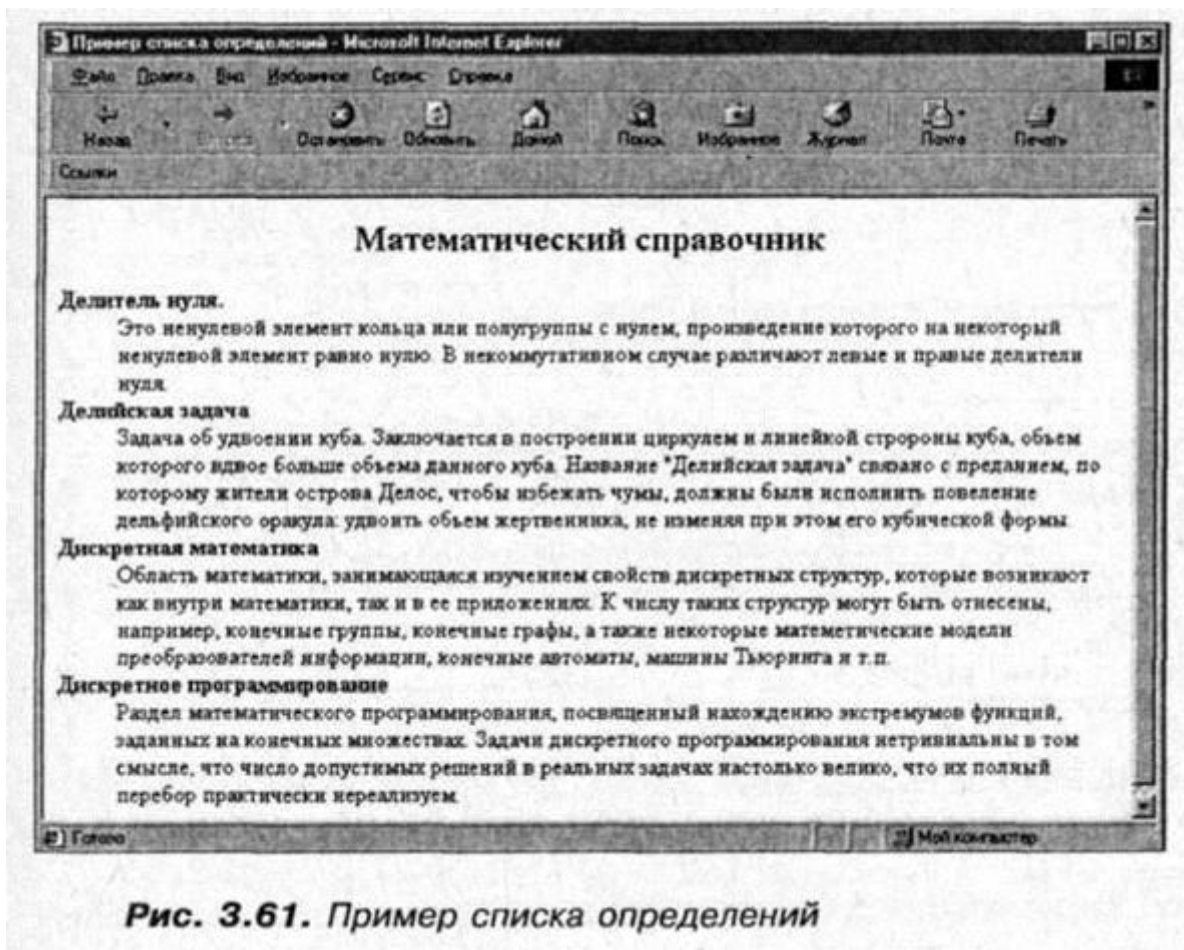
<DD>ЭТО ненулевой элемент кольца или полугруппы с нулем, произведение которого на некоторый ненулевой элемент равно нулю. В некоммутативном случае различают левые и правые делители нуля. <DT>` ` <B>Делийская задача</B>

<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 оркула: 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. 3.61. Пример списка определений**

\*Комбинирование различных списков\*

Возможности языка HTML позволяют комбинировать списки различного типа друг с другом, вкладывать их друг в друга.

Рассмотрим эту возможность на примере:

<HTML>

<HEAD>

<TITLE>

Пример вложения друг в друга списков различного типа.

</TITLE>

</HEAD>

<BODY bgcolor=white>

<H2>Содержание учебника по истории раннего средневековья</H2>

<LI>Глава 1. Западная и центральная Европа в VI-XI веках

<LI>Древние Германцы и Римская империя

<LI>Франкское королевство

<DT>

<DD>` `<B>

<DT>Франкское королевство

<DD>ОНО ВОЗНИКЛО В конце V века на части территории бывшей Римской империи при завоевании франками во главе с Хлодвигом Галлии.

</DL>

</B>

</DL>

<BR> .....

<LI value=7>Культура Западной и Центральной Европы.

<B>Образование славянских государств

</OL>

<LI>Глава 2. Византийская империя в VI-XI веках

<OL>

<LI>Византия при Юстиниане

<DT>

<DD>` `<B>

<DT>Юстиниан I.

<DD>Завоевал Северную Африку, Сицилию, Италию, часть Испании. Провел кодификацию римского права, стимулировал большое строительство.

</DL>

</B>

</DL>

## <B>Жультура Византии

</OL>

<B>Приложения

<OL type=A>

<LI>Краткая биография Карла Великого

<LI>Карта Франкского королевства в X веке

<BR> .....

<LI value=5> Хронология развития Византийского государства

<B>Жарта Византийского государства в X веке

</OL>

</UL>

</BODY>

</HTML>

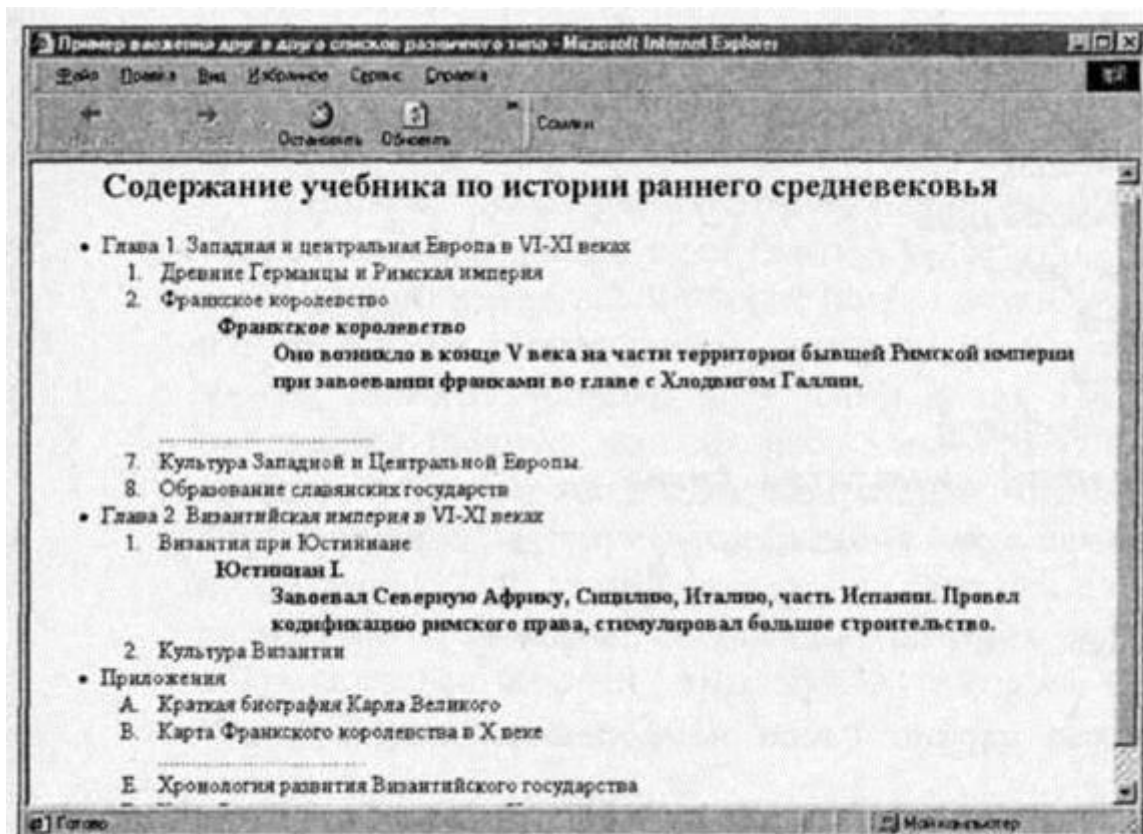


Рис. 3.62. Пример вложения друг в друга списков различного типа

## Элементы списков MENU и DIR.

### Использование этих элементов

Элемент MENU позволяет задавать список как пункты меню.

Элемент DIR представляет собой список-каталог. Оба этих списка отображаются как неупорядоченные, и использование обоих этих списков является нежелательным.

Первоначально они были введены в HTML с перспективами дальнейшего их развития. Однако эти перспективы реализованы не были и сейчас они представляют собой упрощенный и ненужный вариант неупорядоченного списка, создаваемого элементом UL.

В спецификации HTML 4.01 элементы MENU и DIR определены как отмененные, однако они еще поддерживаются всеми браузерами. Вместо списков, задаваемых этими элементами, рекомендуется использовать неупорядоченные списки.

При задании списков MENU и DIR недопустимо использовать в их содержимом элементы уровня блока, что говорит о невозможности вложенных списков такого типа.

*Пример использования (рис. 3.63)*

```
<HTML>

<HEAD>

<TITLE>Пример использования элементов MENU и DIR

</TITLE>

</HEAD>

<BODY>

<DT>Франкское королевство

<DD>ОНО ВОЗНИКЛО В конце V века на части территории бывшей Римской империи при завоевании франками во главе с Хлодвигом Галлии.

</DL>

</B>

</DL>

<BR> .....

<LI value=7>Культура Западной и Центральной Европы.

<LI>Образование славянских государств

</OL>

<LI>Глава 2. Византийская империя в VI-XI веках

<OL>

<LI>Византия при Юстиниане

<DT>

<DDXB>

<DT>Юстиниан I.

<DD>Завоевал Северную Африку, Сицилию, Италию, часть Испании. Провел кодификацию римского права, стимулировал большое строительство.

</DL>

</B>
```



</DL>

## <I>Жультура Византии

</OL>

<LI>Приложения

<OL type=A>

<LI>Краткая биография Карла Великого

<LI>Карта Франкского королевства в X веке

<BR> .....

<LI value=5> Хронология развития Византийского государства

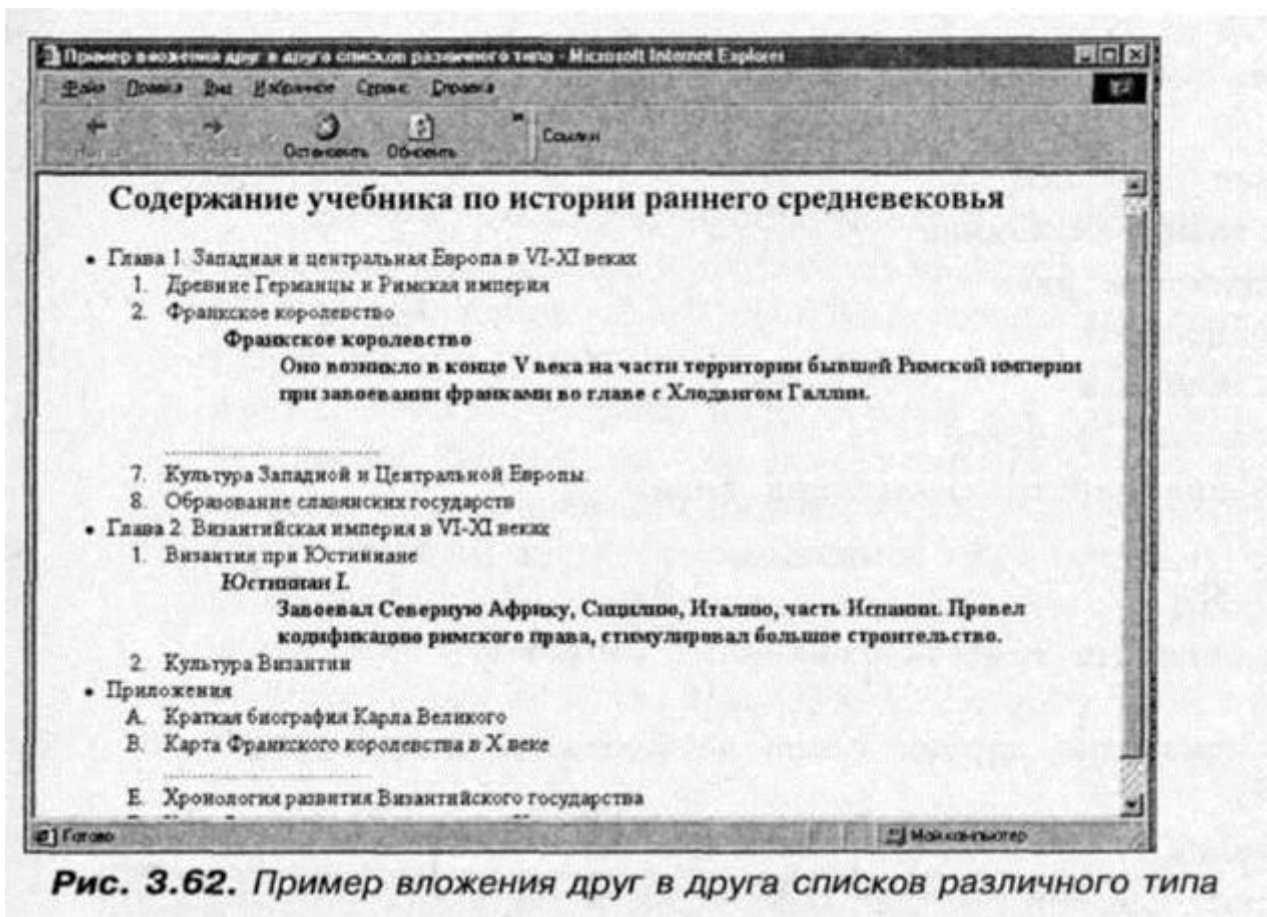
<LI>Карта Византийского государства в X веке

</OL>

</UL>

</BODY>

</HTML>



# Элементы списков MENU и DIR.

## Использование этих элементов

Элемент MENU позволяет задавать список как пункты меню.

Элемент DIR представляет собой список-каталог. Оба этих списка отображаются как неупорядоченные, и использование обоих этих списков является нежелательным.

Первоначально они были введены в HTML с перспективами дальнейшего их развития. Однако эти перспективы реализованы не были и сейчас они представляют собой упрощенный и ненужный вариант неупорядоченного списка, создаваемого элементом UL.

В спецификации HTML 4.01 элементы MENU и DIR определены как отмененные, однако они еще поддерживаются всеми браузерами. Вместо списков, задаваемых этими элементами, рекомендуется использовать неупорядоченные списки.

При задании списков MENU и DIR недопустимо использовать в их содержимом элементы уровня блока, что говорит о невозможности вложенных списков такого типа.

*Пример использования (РИС. 3.63)*

<HTML>

<HEAD>

<TITLE>Пример использования элементов MENU и DIR

</TITLE>

</HEAD>

<BODY>

<STRONG> Семь чудес света </STRONG>

<BR>` ` <BR>

*<LI>Египетские пирамиды*

<LI>Висячие сады Семирамиды

<LI>Александрийский маяк

<LI>Колосс Родосский

<LI>Гробница Мавзола

<LI>Храм Артемиды Эфесской

<LI>Статуя Зевса работы скульптора Фидия

</MENU>

<BR>` `<BR>` `<BR>` `<BR>

<DIR>

<STRONG>

Основные политические партии России на рубеже XIX-XX веков

</STRONG>` `<BR>

<LI>Черносотенцы

<LI>Октябристы

< LI > К а д е т ы

<LI>Социал-революционеры

<LI>Анархисты

<LI>Большевики

<LI>Меньшевики

</DIR>

</BODY>

</HTML>

