Министерство общего и профессионального образования

Свердловской области.

Учебно-технический цент «Омега-1»

***Предмет:*** Информатика

**Разработка web сайта на основе HTML с использованием JavaScript.**

***Исполнитель:*** Неуймин Николай

ученик 11 «Б» класса

***Руководитель:*** Речкалова Н.В.

учитель ИВТ

**Екатеринбург 2005.Содержание:**

Введение 4

История HTML 5

Версии HTML 5

Гипертекст 6

Структура WEB-страницы 8

<HTML></html> 8

<HEAD></head> 9

<TITLE></title> 9

<STYLE></style> 9

<META> 9

<BODY></body> 9

<!-- Комментарий --!> 10

<H1><h1> 10

<HR> 10

<A></a> 11

Переход внутри одного документа 11

Переход к другому документу 12

<BASE> 14

Правила синтаксиса 15

Кодирование символов 16

Использование символов 16

Управление цветом 18

Конструктор документов 20

Приложение. Структура приложения 20

Цепочка 20

Применение цепочек 21

Иерархия 22

Уложим картинку в фоновый паркет 23

Графика 27

Формат GIF 27

Формат JPG 27

Как программировать картинки 28

Картинка в тексте 28

Картинка как ссылка 29

Построение таблиц. 30

Что такое таблица 30

Как задать таблицу 30

Атрибуты команды TABLE 31

Атрибуты команды TR 31

Атрибуты команды TD (TH) 32

Многоликие таблицы 33

Страничные отступы 33

Замечание 34

Многоколонная верстка 34

Наложение картинок 34

Замечание 35

Таблицы стилей 36

Уровни CSS 36

Способы определения таблиц стилей 36

Запись шаблона CSS 38

Группировка и наследование 38

Селекторы 39

Псевдоклассы 40

Применение таблиц стилей CSS 40

CSS в форматировании текста 40

Единицы измерения в таблицах стилей 41

Структурное форматирование 42

Введение в JavaScript 44

Использование JavaScript 44

Важность изучения JavaScript 44

JavaScript в действии 44

Ввод/вывод информации с помощью JavaScript 45

Объекты, методы и свойства 45

Управление данными с помощью переменных 46

Значения в языке JavaScript 46

Переменные в языке JavaScript 48

Массивы JavaScript и Escape-последовательности 49

Как пользоваться массивами 50

Выражения, условия, операции, строки и числа 52

Что такое выражения и условия 52

Знакомство с операциями 53

Создание сценариев с помощью функций и событий 56

Описание браузеров 59

Просмотр WEB-страниц 59

Microsoft Internet Explorer 59

Netscape Communicator 61

Терминология 64

Практическая часть 67

Список литературы 68

# Введение

Термин HTML (Hyper Text Markup Language) означает "язык маркировки гипертекстов". Это понятие более широкое, включает в себя Интернет и локальные сети, редакторы, браузеры, разнообразные програмные продукты, компакт-диски, обучающие курсы, дизайн и многое другое. HTML – своеобразная противоположность сложным языкам программирования, известным только специалистам.

HTML давно перестал быть просто языком программирования. Человек, изучавший этот язык, обретает возможность делать сложные вещи простыми способами и, главное, быстро, что в компьютерном мире не так уж и мало. Гипертекст подходит для включения элементов мультимедиа в традиционные документы. Практически именно благодаря развитию гипертекста, большинство пользователей получило возможность создавать собственные мультимедийные продукты и распространять их на компакт-дисках. Такие информационные системы, выполненные в виде набора HTML-страниц, не требует разработки специальных программных средств, так как все необходимые инструменты для работы с данными (WEB-браузеры) стали частью стандартного программного обеспечения большинства персональных компьютеров. От пользователя требуется выполнить только ту работу, которая относится к тематике разрабатываемого продукта: подготовить тексты, нарисовать рисунки, создать HTML-страницы и продумать связь между ними.

HTML, как основа создания WEB-страниц, имеет прямое отношение и к новому направлению изобразительного искусства – WEB-дизайн. Художнику в Интернете недостаточно просто нарисовать красивые картинки, оригинальный логотип, создать новый фирменный стиль. Он должен еще поместить все это в Сети, продумать связь между WEB-страницами, чтобы все двигалось, откликалась на действие пользователя, поражало воображение, вызывало желание создать что-нибудь свое, оригинальное в этой области.

***Цель работы***: изучить языки программирования HTML и JavaScript.

Исходя и выше поставленной цели при создании данного проекта я поставил следующие ***задачи***:

1. Изучить и проанализировать литературу по данной теме.
2. Изучить принцип работы браузеров.
3. Создать оригинальный сайт.

# История HTML

## Версии HTML

Первая версия HTML была разработана в 1989 году Тимом Бенерс-Ли для популярного в прошлом браузера Mosaic. Но в то время ни для языка, ни для браузера не нашлось достойного применения. В 1993 году появился HTML+ , и эта версия также осталась практически незамеченной. Начало широкого использования гипертекста дала версия 2.0, которая, появилась в июне 1994 года. Это был год роста популярности WWW по всему миру. Элементы, включенные в версию 2 , в большинстве своем используются по сей день.

В версии 3.0 HTML, которая появилась год спустя, была реализована возможность прорисовки математических символов (знаков интервала, бесконечности, дроби, скобок и т.д.) при помощи элементов языка. Под эту версию был разработан браузер Arena. Но этот проект оказался тупиковым и не получил дальнейшего распространения.

В 1996 году появился HTML 3.2. Это было новаторское решение, в спецификацию языка были введены фреймы, которые стали теперь весьма популярны у разработчиков WEB-страниц. Даже сейчас на основе этой спецификации можно реализовать интересные дизайнерские решения. Практически все современные браузеры поддерживают версию 3.2, поэтому авторы WEB-страниц уверенны в работоспособности всех элементов.

Наряду с официальными спецификациями языка, которые разрабатывались организацией W3C (W3 Консорциум ),компании-производители браузеров создавали собственные элементы (расширения). Впоследствии, некоторые из этих элементов, после получения всеобщего признания включились в спецификацию следующих версий языка. Но новаторское решение – фреймы, не были включены в спецификацию 3.2. Но браузеры поддерживали фреймы и многие книги, посвященные HTML, содержали описание фреймов без упоминания о том, что это нестандартные элементы. В последствии, фреймы стали стандартом де-факто. В версии 4 они уже были включены на полном основании.

И наоборот, элементы APPLET и SCRIPT, необходимые для расширения HTML другими программными кодами версии 3.2, не сыграли той роли, которую были призваны сыграть. Это объясняется тем, что браузеры различных версий по-разному интерпретировали программы на разных языках JAVA, JAVASCKRIPT, Visual Basic (VBScript). В результате не удалось получить достаточно надежный работающий код, и данные языки использовались любителями HTML в основном для экспериментов.

Официальная спецификация HTML 4 (Dynamic HTML) появилась в 1997 году. В это время уже было очевидно, что дальнейшее развитие гипертекста будет осуществляться за счет скрипт - программирования. Это оказалось немногим более эффективным, чем вводить в язык все новые элементы. Появившиеся в то время браузеры (Netscape Navigator 4, Microsoft Internet Explorer 4 и др.) уже достаточно надежно интерпретировали программный код (был установлен определенный уровень стандартизации). Однако проблемы у разработчиков еще остались. В качестве примера можно отметить, что многие скрипты начинаются с определения версии браузера, чтобы потом использовать тот или иной фрагмент кода. Очевидно, что на программиста ложится обязанность тестирования страниц на всех популярных в настоящие время браузерах.

В результате, использование всех возможностей Dynamic HTML стало уделом программистов достаточно крупных организаций, где есть условия для разработки сложных программ и всестороннего их тестирования. Создателям личных WEB-страниц подчас приходится искать компромисс между надежностью и новаторством, чтобы получить достаточно грамотный HTML-код.

# Гипертекст

Уже достаточно много времени затрачено на то, чтобы научиться при помощи HTML и браузера выводить на экран тексты.

Но это просто тексты. Используя мощный текстовый процессор, такой, как Microsoft Word, сверстать объявление, письмо брошюру и небольшую книгу можно гораздо быстрее, и результат будет лучшего качества.

При этом не надо писать программы, вникать во все тонкости многочисленных тегов и их атрибутов. Работая в Worde, практически не надо думать ни о чем, кроме содержания собственного произведения. Разнообразный оформительский инструментарий имеет интуитивно-понятный интерфейс и многочисленную библиотеку шрифтов, линий, значков, рамочек, орнаментов, фигур, картиночек и других полезных "штучек", которые делают текст на экране и бумаге приятным для глаза. При условии, конечно, что вы не лишены вкуса и чувства меры.

Так почему же во всем мире разразился настоящий бум HTML-программирования? Сбывается мечта А.П.Ершова: программирование становится второй грамотностью, народным увлечением.

Главных причин популярности HTML три. Вот они в порядке возрастания важности.

HTML-программирование очень простое. В нем нет традиционных алгоритмических структур, таких, как развилка, цикл, процедура. Оно линейно в своей основе.  
(Отступлением от этого тезиса являются ссылки. С их помощью можно организовать ветвление и процедурный вызов. Об этом будет рассказано ниже. Однако, эти средства слишком примитивны для серьезного сопоставления с аналогичными структурами языков типа Си или Паскаль.)

Многие авторы учебников даже стесняются называть HTML-тексты программами. Работу HTML-проектировщика они называют разметкой текста. На самом деле, в этом названии заключена методическая ошибка. Ведь оно (название) предполагает такой порядок работы:

берется обычный текст

и размечается, то есть, на него накладывается сетка из тегов так, чтобы браузер смог показать строчки текста на экране.

Однако, хорошие гипертекстовые документы при таком подходе получить гораздо сложнее, чем в случае когда, структура HTML-документа планируется еще до написания текстов. Ведь гипертекст устроен принципиально иначе (см. третью причину).

Переносимость.

Если, работая в Word'e, вы создаете документ для собственного потребления, у вас нет проблем. Нет проблем и тогда, когда вы посылаете свою работу в мир как печатную копию. Если же ваш партнер, издатель или коллега просит прислать электронную версию -- начинаются проблемы.

Для того чтобы документ появился на экране у абонента, необходимо, чтобы его компьютер и операционная система (платформа) позволяли запустить ваш любимый редактор.

Для того чтобы документ загрузился в Word, крайне желательно совпадение версий этого продукта у вас и вашего коллеги.

Для того чтобы текст на экране не выглядел как древненорвежский манускрипт, необходимо, чтобы ваш партнер имел на своем компьютере шрифты, которые были использованы при создании вашего текста.

Но даже, если все эти условия выполнены, все равно мало шансов на то, что ваш товарищ увидит текст неискаженным (картинки вылезают из рамок, границы страниц плывут по тексту,...). Это происходит потому, что ваш Word и Word вашего партнера настроены по-разному.

После того, как по телефону (или электронной почте) редакторы будут настроены одинаково, текст все равно может выглядеть по-разному уже по причинам, известным только фирме Microsoft.

Все, о чем говорилось, называется непереносимостью продукта. Не в том смысле, что продукт кого-то не любит, а в том, что разным пользователям невозможно, или крайне тяжело, отображать у себя то, что сделано в другом месте.

HTML-документ является переносимым продуктом. Это означает, что авторский труд будет легко доступен огромному числу пользователей, независимо от марки компьютера и типа операционной системы. Именно это свойство HTML-документов, а также их относительно малый размер, позволили с успехом использовать HTML-технологию для подготовки WWW-страниц в Интернете.

Переносимость HTML-документа достигается за счет того, что пересылается не экранное изображение, а программа. А изображение строит браузер, выполняя команды этой программы. Правда, и здесь не все гладко. Разные браузеры могут работать немного по-разному.

HTML-документ -- это гипертекст.

Обычный текст имеет линейную структуру. И хотя пользователь может читать любые его страницы и строчки, листая документ на экране или в бумажной книге, автор предполагает, что текст читается по порядку. Сначала первая страница, потом вторая и так далее.

Первый этюд к гипертексту

Некоторое нарушение линейности обычной книги вносят сноски, ссылки на другие страницы и ссылки на другую литературу.

Предполагается, что пользователь может прервать линейное чтение в месте ссылки, посмотреть другую часть читаемого текста или даже совсем другой текст, а затем продолжить чтение с места прерывания.

Второй этюд к гипертексту

Из массы линейной текстовой продукции выделяются словари, справочники, энциклопедии. Физически текст устроен линейно: за страницей 10 обязательно следует страница 11, а за сотой -- сто первая. Однако, предполагается, что читатель обращается к книге для справки, а не читает ее всю подряд от корки до корки.

Для упрощения навигации пользователя в таких книгах предусматриваются подробные оглавления, алфавитные и тематические указатели. Разделы словаря или справочника имеют систему развитых перекрестных ссылок. (Помните как у Лема:СЕПУЛЬКА - предмет для сепулькации, см. сепулькация.СЕПУЛЬКАЦИЯ -- действие, выполняемое при помощи СЕПУЛЬКИ, см. сепулька.)

Третий этюд к гипертексту

В век бурных компьютерных технологий как-то не хочется вручную копаться в большом словаре. Почему бы эту рутину не поручить "железному" другу с "мягкой" душой? Вгоним словарь в "железо" и настроим "душу". Щелкнул мышкой по нужному слову -- получил результат: нужный раздел книги на экране.

Последний этюд к гипертексту

Возможности компьютера рождают принципиально новую идею: почему бы изначально не проектировать текст в расчете на возможность чтения не по-порядку, а по контексту. Такой подход позволяет разным пользователям просматривать текст ориентируясь, на свои вкусы, уровень владения темой и поставленные задачи.

Структура компьютерной книги становится существенно нелинейной, она даже перестает быть иерархической, а скорее напоминает спутанную рыболовную сеть или порцию спагетти, опрокинутую на пол.

Для навигации по такой сети предусматривается простой способ: те фрагменты документа, которые имеют переходы на другие его части, каким-то образом выделены. Простое интерфейсное действие на такой ссылке (нажатие на Enter или мышиный щелчок) перебрасывают пользователя в другой информационный узел.

Посмотрел, вернулся обратно, или, не возвращаясь, продолжил путешествие по новой ссылке.

Синтаксис HTML.

## Структура WEB-страницы

Заготовка типичной WEB-страницы:

<HTML>

<HEAD>

<TITLE> Структура WEB-страницы </title>

<STYLE> H2 {font-family: Arbat;}CODE { font-family: Arial;} </style>

<META http-equiv. = ”Content-Type” content = “text/html; chariest = windows-1251">

<META name = “Author” content = “Pavel Trizno”>

<META name = “Keywords” content = “ HTML, document, element”>

</head>

<BODY bgcolor = #FFFFFF>

<!-- Комментарий к страницы --!>

<A name = “top></a>

Переход к <A href = “#bottom”> конец </a> документа <P>

Переход к <A href = “#S001”> <B> ссылке 1 </b></a><P>

<P>

<HR>

<H1> Заголовок 1 </h1>

<H2> Заголовок 2 </h2>

<H3> Заголовок 3 </h3>

<H3> Заголовок 4 </h4>

<H3> Заголовок 5 </h5>

<H3> Заголовок 6 </h6>

<HR>

Здесь расположена <B> ссылка 1 </b><A name = “S001”></a>

<HR>

<P> Здесь должен располагаться оригинальный текст WEB-страницы

<HR>

<A name = “bottom”></a><P>

Переход в <A href = “#top”>начало</a> документа

</body>

</html>

Если рассмотреть исходные тексты различных WEB-страниц, то можно легко увидеть схожесть их структуры. Это объясняется тем, что документы создаются по определенным правилам. В основу синтаксиса языка HTML лег стандарт ISO 8879:1986 «Information processing. Text and office systems. Standard Generalized Markup Language (SGML)». Правда, существует большое различие между стандартом официальном и стандартом фактическим. HTML постоянно развивается, дополняется новыми элементами.

Рассмотрим основные элементы HTML.

### <HTML></html>

Означает документ на языке HTML. Я уже упоминал о том, что одним из принципов языка является многоуровневое вложение элементов. Данный элемент является самым внешним, так как между его начальным и коечным тегами должна находится вся WEB – страница. В принципе этот элемент. Он имеет атрибуты version, long и dir, которыми в данном случае редко кто пользуется, и допускает вложение элементов HEAD, BODY и других, определяющую общую структуру WEB-страницы. Конечным тегом </html> оканчиваются все документы в формате HTML.

### <HEAD></head>

Область заголовка Web – страницы, её первая часть. HEAD служит только для формирования общей структуры документа. Этот элемент может иметь атрибуты IANG, DIR, должен включать элемент TITLE и допускает вложение элементов BASE, META, LINK, OBJECT, SCARIPT, STULE.

### <TITLE></title>

Элемент для размещения заголовка Web – страницы. Строка текста , расположенного внутри этого элемента, отображается не в документе, а в заголовке окна браузера. Эта строка часто используется при организации поиска в WWW. Эта строка должна достаточно точно отражать назначение документа.

### <STYLE></style>

Элемент для описания стиля некоторых элементов Web – страницы. В файле Strukt. Htm назначены шрифты для элементов h2, code. Для каждого элемента существует стилевое оформление по умолчанию, поэтому употребление элемента style не обязательно, но желательно. Этот элемент введён сравнительно недавно, без него немыслима работа в приложениях World, Excel.

### <META>

Элемент содержит служебную информацию, которая не отражается при просмотре Web- страницы. Внутри него нет текста в обычном понимании, поэтому нет и конечного тега. Каждый элемент Meta содержит два основных атрибута, первый из которых определяет тип данных, а второй – содержание. Вот несколько примеров meta – данных:

Дата, обозначающая срок годности документа:

*Name – “Expires” content = “Дата”*

Адрес электронной почты:

*Name = “Reply – to” content = “ Имя @ Адрес “*

Имя автора Web-страницы:

*Name = “Author” content = “ Имя автора”*

Набор ключевых слов для поиска:

*Name = “Keywords” content = “слово 1, слово 2, слово 3…”*

Краткое описание содержания Web- страницы:

*Name = “Description” content = “Содержание страницы”*

Описание типа и характеристик Web – страницы:

*Name = “Content – Type” content = “ Описание страницы”*

Указание приложения, в котором была создана Web – страница:

*Name = “Generator” content = “Название HTML – редактора”*

Атрибут Name используется приложением – клиентом для получения дополнительной информации о Web – страницах и их упорядочения. Его часто заменяют атрибутом http – equiv. Он используется сервером для создания дополнительных полей при выполнении запроса.

Элемент META может содержать URL. Шаблон атрибута таков:

URL = “http: // адрес”

### <BODY></body>

Этот элемент заключает в себе гипертекст, который определяет собственно Web – страницу. Конечный тег этого элемента надо искать в конце HTML-файла. Внутри элемента BODY можно использовать все элементы, предназначенные для дизайна Web-страницы. Внутри начального тега элемента BODY можно расположить ряд атрибутов, обеспечивающих установки для всей страницы целиком.

Атрибут, определяющий фон страницы, является одним из самых важных для дизайна.

*background = “Путь к файлу”*

Более простое оформление фона сводится к заданию его цвета:

bgcolor = “#RRGGBB”

Цвета фона задается тремя двухразрядными шестнадцатеричными числами, которые определяют интенсивность красного, зеленого и синего. Оба приведенных выше атрибута не являются альтернативными и часто используются совместно: если по каким-либо причинам не может быть найден рисунок фона, используется цвет.

Поскольку фон станицы может изменятся, необходимо иметь возможность подбирать соответственный цвет текста. Для этого имеются следующей атрибут:

*text = “#RRGGBB”*

Для задания цвета текста гиперссылок используются следующие атрибуты:

*link = “#RRGGBB”*

Точно так же можно задать цвет для просмотренных гиперссылок:

*vlink = “#RRGGBB”*

Можно также указать изменение цвета для последней выбранный пользователем гиперссылки:

*Alink = “#RRGGBB”*

### <!-- Комментарий --!>

В любом языке программирования есть конструкции, позволяющие создавать произвольные ремарки. HTML в этом смысле – не исключение. Текст, введений внутри этого элемента, игнорируется браузером. Эти элементы могут располагаться в любом месте WEB-страницы. Признаком комментария служит восклицательный знак, а текст комментария должен обрамляться двойным дефисом. Например:

*<!-- Комментарий --!>*

### <H1><h1>

Элемент заголовка. Существует шесть уровней заголовка, которые обозначаются H1…H6. Заголовок уровня 1 самый крупный, а уровень 6 обеспечивает самый маленький заголовок. Для заголовков могут использовать атрибуты, задающие выравнивание влево, по центру или вправо:

*align = “left”*

*align = “center”*

*align = “right*”

### <HR>

Горизонтальная линия – очень часто используемый элемент. Во-первых, потому что с его помощью очень удобно делить страницу на части.

Во-вторых, потому что выбор подобных элементов оформления у автора страницы очень небольшой. Этот элемент не имеет конечного тега, но допускает ряд атрибутов для выравнивания влево, по центру, вправо, по ширине:

*align = “left”*

*align = “center”*

*align = “right*”

*align = “justify*”

Можно задать толщину линии:

*Size = толщина в пикселях*

Можно управлять длиной линии:

*width = длина в пикселях*

*width = длина в процентах*

Можно выбрать цвет:

color = “цвет”

### <A></a>

HTML-документ может быть очень большим, и в этом случае нужно иметь возможность быстрого перемещения к нужному разделу документа. Для этого можно использовать механизм гиперссылок. Необходимо также в нужных местах текста расставить соответствующие метки. Шаблон для создания меток таков:

<A name = “>Произвольный текст</a>

В данном случае данной строке документа присваивается имя, и, следовательно, в другой части документа или даже на другом документе может быть создана гиперссылка, приводящая в эту точку. Для перехода внутри документа можно использовать следующую конструкцию:

<P>Переход к <A href = “#метка”>метке</a></p>

Несколько подобных строк могут образовать своеобразное оглавление Web-страницы, которое можно разместить в начале и в конце документа.

Читая эту работу, вы, конечно, уже достаточно много поработали со ссылками и понимаете, что это такое на практике.

Вы поняли, что:

гипертекстовые ссылки среди других элементов текста выделяются цветом и подчеркиванием;

мышиный курсор на ссылке меняет свою форму и превращается в указующий перст;

для перехода по ссылке необходимо щелкнуть по ней мышкой;

для возврата из ссылки необходимо использовать навигационную кнопку браузера "Назад" ("Back").

Если ссылки образуют вложенную цепочку, то кнопки "Назад" ("Back") и "Вперед"("Forward") можно использовать для движения по пройденному ссылочному пути в обе стороны. Они работают как традиционные операции "откатка" и "накатка" в большинстве прикладных программ.

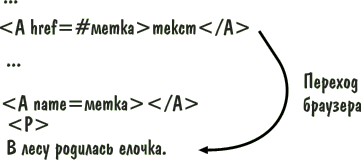
В нижеприведенном примере используется цепочка вложенных ссылок. Совершите небольшое путешествие. Сначала, щелкая ссылки, доберитесь до текста, в котором ссылок уже нет. Затем, пользуясь навигационными кнопками браузера, "прогуляйтесь" по пройденному пути вперед и назад.

Ссылки могут уводить не только в разные документы, но загружать на экран разные фрагменты одного и того же текста. В последнем случае навигационные средства браузера работают также.

#### Переход внутри одного документа

Для задания гипертекстового перехода внутри документа используют две команды <A>. Первая команда с атрибутом **href** является источником перехода, вторая с атрибутом **name** -- приемником.

|  |  |
| --- | --- |
| **Общий вид программы** |  |
| ...  <**A** href=#метка>текст</**A**>  ...  ...  <**A** name=метка></**A**>  ... | Задание перехода по метке  На экран выводится ссылка: текст  Метка. Сюда браузер приходит по ссылке.  На экране ничего не отображается. |



Для организации перехода внутри документа нужно:

Выбрать имя для метки.

Имя должно быть уникальным в HTML-программе. Это означает, что других меток с таким именем в программе быть не должно.  
А что будет, если не послушаться этого совета?  
Когда одинаковых меток несколько, браузер выполняет переход на самую первую метку в программе.

Имя может заключаться в кавычки.

Имя может состоять из любых символов. В том случае, когда имя содержит пробелы или ключевые символы HTML, кавычки обязательны.

Запрограммировать переход по метке.  
В месте перехода нужно написать команду <**A** href=#метка>текст</**A**>.  
Здесь:

**#** -- ключевой символ;

метка -- выбранное имя для метки;

текст -- запись, которая будет выглядеть на экране браузера как ссылка.

Установить тег с меткой в нужном месте HTML-программы.  
Перед фрагментом HTML-программы, на который должен выполниться переход записать команду-метку <**A** name=метка></**A**>.  
При этом:

имя метки должно быть то же самое, что и в команде, задающей переход;

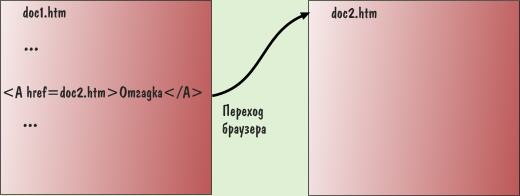
на экране тег-метка <**A** name=метка> никаких изображений не строит.

#### Переход к другому документу

Для того, чтобы браузер загрузил в свое окно новый HTML-документ, нужно записать в программе ссылку при помощи команды <**A**> с атрибутом href=имя\_файла.

|  |  |
| --- | --- |
| **Общий вид программы** |  |
| ...  <**A** href=имя\_файла>текст</**A**>  ... | Выполнить файл "имя\_файла".  На экран выводится ссылка: текст  При щелчке пользователя на ссылке  браузер строит на экране документ  по программе, заданной в файле  "имя\_файла". |

Документ  
Программа (файл doc1.htm)  
Программа, на которую настроена ссылка в документе (файл doc2.htm)



Переход к метке другого документа

Мы научились решать такие задачи:

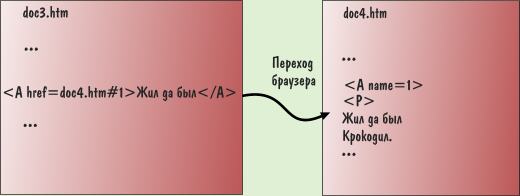
передавать управление в другую часть документа;

передавать управление в другую программу (HTML-файл).

Эти два метода можно объединить и передать управление из одного документа к метке внутри другого.

|  |  |
| --- | --- |
| **Общий вид программы** |  |
| Первый HTML-файл  ...  <**A** href=имя\_файла#метка>текст</**A**>  ...  Второй HTML-файл  ...  <**A** name=метка></**A**>  ... | Приступить к показу фрагмента  с меткой "метка" в файле  "имя\_файла". На экран выводится  ссылка: текст.  При щелчке пользователя на  ссылке браузер строит на экране  документ по программе, заданной  в файле "имя\_файла", начиная с  фрагмента с меткой "метка". |

Документ  
Программа (файл doc3.htm)  
Программа, на которую настроена ссылка в документе (файл doc4.htm)



Имена файлов и ссылки на них

Имена файлов с HTML-программами имеют расширение htm или html.

Без необходимости обозначение html лучше не использовать, так как не все операционные системы понимают 4 символа в расширении имени файла.

Имена для файлов всегда стараются придумать осмысленные. Когда файлов много, порядок в именах имеет особое значение.

Если, например, ваш многофайловый гипертекст имеет иерархическую структуру, старайтесь в обозначениях учитывать иерархию. Так, основной файл с оглавлением можно назвать 00.htm (или index.htm). Файл с первой главой -- 01.htm и так далее (быть может до 99.htm). Когда главы имеют файлы-параграфы, их имена получаются из имени главы добавлением двузначного номера параграфа. Так же поступают и с файлами-пунктами параграфа.

Предлагаемая нотация позволяет по имени файла определить его принадлежность. Например, файл 0205.htm -- описывает построение пятого параграфа второй главы, а файл 111299.htm -- 99-ый пункт 12-ого параграфа 11-ой главы.

Если в теге <**A**> указано имя файла, браузер ищет файл в том же каталоге, в котором расположен файл со ссылкой. Когда необходимо сослаться на файл из другого каталога можно:

указывать в ссылке полный путь к файлу, начиная с имени дисковода;

указывать в ссылке путь по отношению к файлу со ссылкой.

Второй способ гораздо лучше, так как при правильном использовании позволяет переносить ваш многофайловый гипертекст из одного каталога в другой и с одного компьютера на другой без дополнительных настроек.

|  |  |
| --- | --- |
| Пусть, например, ваш гипертекст расположен в каталоге BOOK корневого каталога диска C. В самом каталоге BOOK расположен файл с оглавлением 00.htm. Остальные файлы помещены в подкаталоги 01, 02, 03, 04 в соответствии с принадлежностью к той или иной главе документа. |  |

Как можно сослаться из файла 00.htm на файл 02.htm?

**C:/BOOK/02/02.htm** (абсолютная адресация)

**./02/02.htm** (относительная адресация)

Первый способ приводит к непереносимости вашего продукта. Стоит вам перезаписать, например, каталог BOOK на диск D, как ссылки перестают работать.

Вторая запись предлагает браузеру ориентироваться на текущий каталог -- об этом говорит указание "./". Можно перемещать каталог BOOK в любое место и даже переименовывать его - ссылки работать будут.

Относительная ссылка из файла 02.htm в файл 00.htm запишется так: **../00.htm**. Символы "../" указывают на родительский каталог.

Замечание

При записи пути к файлу используйте косую черту с наклоном вправо, а не влево, как принято в DOS, даже если ваш браузер находит ссылки и с левонаклонной чертой. Наклон черты становится существенным при размещении гипертекста на сервере для доступа к нему из Интернета. Серверы, как правило, работают под операционной системой UNIX, а в ней черточки -- правонаклонные.

### <BASE>

Элемент для задания базового адреса (URL) для ссылок. Это позволяет опускать начальную часть адреса в ссылках документа. Для использования этого элемента необходимо использовать следующую конструкцию:

<BASE href =”http:// компьютер/путь 1”>

Фрагмент адреса *путь1* не является обязательным. При формировании полного адреса он будет отброшен. Если в тексте документа встретится относительная ссылка

<A href =”*путь 2/имя документа. Htm “ >Видимый текст ссылки</a>,*

то она будет соответствовать URL

http: // компьютер/путь 2/имя документа. Htm

В том случае, когда надо задать базовый адрес для локального диска (например, D), должна быть использована такая конструкция:

<BASE href =file://D:\путь\>

Тогда при указании относительной ссылки можно будет задавать не только имя файла, но и имена папок, в которых он находится.

В выражении абсолютной ссылки можно также опустить указание на схему доступа (file://) В этом случае будет учитываться только левая часть абсолютной ссылки до первого левого символа “\”, то есть имя локального диска.

## Правила синтаксиса

При использовании каждого элемента важно знать, какие элементы могут располагаться *внутри него,* и *внутри каких* элементов может находиться он сам. Так, взаимное расположение элементов HTML, HEAD, TITLE, BODY должно быть стандартным на любой странице, в тех случаях, когда не используются фреймы. Если страница представляет собой документ планировки фреймов, то вместо элемента BODY используется элемент FRAMESET.

Существуют группы элементов, которые используются совместно. К ним относятся элементы для создания таблиц, списков, фреймов. В этом случае порядок вложения элементов определяется логикой создания того или иного объекта на странице. Таблицы и фреймы часто используются для того, чтобы разместить детали страницы (рисунки, текст и др.) в определённом порядке. Например, располагая рисунок *внутри* ячейки таблицы, можно добиться определённого его положения.

Большое количество элементов, которые используются для форматирования текста, допускают самые разнообразные варианты вложения. И сами они обязательно должны располагаться внутри определённых элементов.

Например, есть два абзаца (первый в зелёной рамке) и таблица:

<P style = “border: 3px solid green”> Текст абзаца 1</p>

<TABLE>…</ table>

<P> Текст абзаца 2 </p>

Таблица в данном случае – независимый элемент. Её можно, например, выравнивать независимо от остального текста.

Можно использовать другой код:

<P style = “border: 3px solid green “>Текст абзаца 1

<TABLE>…</table>

<P> текст абзаца 2</p>

Исчез конечный тег первого абзаца. Теперь таблица является частью первого абзаца, и зелёная рамка будет охватывать таблицу и текст.

И наоборот, элемент P может находиться внутри таблицы: например, один элемент ячейки TD может содержать несколько абзацев P.

Правила синтаксиса распространяются и на использование стартового и конечного тегов, атрибутов и содержимого элемента. Нельзя путать понятия «элемент» и «тег». *Элемент –* это контейнер, содержащий атрибуты внутри стартового тега и полезную информацию между стартовым и конечным тегами. *Тег –* это конструкция, заключённая в угловые скобки и используемая для обозначения области действия элемента.

Некоторые элементы не имеют конечного тега. Элементу BR, обозначающему конец строки, не нужен конечный тег. Некоторые элементы могут использоваться с конечным тегом и без него. Элемент абзаца P может иметь конечный тег , но если тег не задан, то признаком окончания действия элемента служит следующий элемент, который может логически определить конец текущего абзаца: другой элемент P, элемент рисунка IMG, элемент списка UL, элемент таблицы TABLE и др.

Важным правилом, которое не имеет исключений, является размещение атрибутов элемента *внутри начального тега.*

## 

## Кодирование символов

Основным языком Интернета является английский язык. Но для гипертекстовых документов предусмотрено использование национальных алфавитов. Стандартным режимом отображения является кодировка ISO Latin 1 (ISO 8859-1). Она подходит как для MS-DOS (набор символов ASCCII), так и для Windows. В тоже время браузеры поддерживают набор символов Unicode 2.0 (ISO10646), что позволяет использовать национальные алфавиты. Это означает, что символы могут кодироваться однобайтовым числом (коды в пределах 0-255) или двухбайтовым (0-65 535). В первом случае для использования национального алфавита необходим атрибут charset, так как одни и те же коды символов могут быть интерпретированы по-разному для различных кодовых страниц.

Кириллица (Windows) и/или Кириллица (Кои8-Р) – это две наиболее популярные в нашей стране кодировки.

Если в документе есть *указание* на определённую кодовую страницу, то выбор шрифта будет предопределён. Коды 32-127, то есть знаки препинания, цифры и буквы латинского алфавита, отображаются правильно, а коды 128-255 могут отображаться по-разному.

Если кодовая страница указана неправильно, то, например, при создании гипертекстовых документов в MS FrontPage Express в текст страницы автоматически добавляется конструкция типа charset =xxxxx, которая не позволяет использовать кириллицу. В этом случае необходимо правильно указать значение атрибута: charset =windows-1251.

Если документ использует кодировку Unicode, то для работы с ним подойдут такие средства, как Блокнот, Norton Commander или WordPad, и придётся остановиться на гипертекстовом редакторе. В этом случае русская буква А будет выглядеть в режиме «источника» так &#1040 (в десятичной кодировке). Такую страницу будет сложно читать и редактировать. Поэтому каждый новый редактор надо тестировать на возможность использования русских букв: набрать небольшой текст, сохранить документ, а затем просмотреть его в режиме источника.

## Использование символов

В HTML и в браузерах реализована возможность прорисовки символов по их кодам. Символы могут быть общепринятыми, нестандартными и используемыми в HTML в качестве служебных символов. Их называют *спецсимволами.* Первый способ ввода спецсимвола заключается в указании его кода. Например, латинскую букву А можно задать так: &#65. Для некоторых символов предусмотрена *мнемоническая* кодировка.

Часто используемые спецсимволы (по стандарту ISO 8859 – 1):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Код символа** | **Числовой код** | **Мнемонический код** | **Название** | **Символ** |
| 34 | &#34 | &quot | Прямая кавычка | “ |
| 38 | &#38 | &amp | Амперсанд | & |
| 60 | &#60 | &lt | знак «меньше» | < |
| 62 | &#62 | &gt | Знак «больше» | > |
| 153 | &#153 | &trade | Троговая марка | ™ |
| 160 | &#160 | &nbsp | Неразрывный пробел |  |
| 162 | &#162 | &cent | Цент | ¢ |
| 163 | &#163 | &pound | Фунт | £ |
| 164 | &#164 | &curren | Знак валюты | ¤ |
| 165 | &#165 | &yen | Йена | ¥ |
| 166 | &#166 | &brvbar | Вертикальная строка | ¦ |
| 167 | &#167 | &sect | Знак параграфа | § |
| 169 | &#169 | &copy | Знак копирайта | © |
| 171 | &#171 | &laquo | Левая типографская кавычка | « |
| 172 | &#172 | &not | Знак отрицания | ¬ |
| 174 | &#174 | &reg | Знак “зарегестрировано” | ® |
| 176 | &#176 | &deg | Знак градуса | ° |
| 177 | &#177 | &plusmn | Знак “плюс минус» | ± |
| 178 | &#178 | &sup2 | Степень 2 | ² |
| 179 | &#179 | &sup3 | Степень 3 | ³ |
| 181 | &#181 | &mikro | Знак «микро» | µ |
| 182 | &#182 | &para | Знак абзаца | ¶ |
| 183 | &#183 | &middot | Точка-маркер | · |
| 185 | &#185 | &sup1 | Степень 1 | ¹ |
| 187 | &#187 | &raquo | Правая типографская кавычка | » |
| 188 | &#188 | &frac14 | Одна четвёртая | ¼ |
| 189 | &#189 | &frac12 | Одна вторая | ½ |
| 215 | &#215 | &times | Знак умножения / кавычки/ | \* |
| 247 | &#247 | &divide | Знак деления | / |

Для отделения символа от последующего текста надо вводить точку с запятой.

Мнемонические символы бывают очень полезны, особенно для тех символов, которые нельзя ввести с клавиатуры. Например, для указания разрешения экрана можно использовать такую конструкцию:

640&times; 480

Двойная кавычка, угловые скобки и амперсанд (<, >, &) являются служебными в HTML. Если их надо использовать в обычном тексте на Web-странице, то они должны быть указаны только при помощи кодов.

Кириллица тоже может быть реализована в виде набора спецсимволов. Русские буквы расположены (при кодировании Windows-1251) в том месте кодовой таблицы, где первоначально располагались латинские буквы с тильдами, умляутами. Поэтому англоязычные редакторы HTML при переключении клавиатурного регистра часто записывают кириллицу в виде кодов.

Типы данных

Основным типом данных Web-страницы является текстовой тип. Существует много элементов, которые являются контейнерами для текста, или используются для форматирования текста. Числовые и мнемонические коды, обозначающие символы, предваряются знаком амперсанта &. Таким образом, амперсанд и угловые скобки могут быть введены в текст не на прямую, а только при помощи кодов. Текстовые данные, являющиеся значениями атрибутов, заключаются в прямые двойные кавычки.

Числовые данные требуются только для указания значений атрибутов и записываются без кавычек.

Гиперссылки обычно состоят из двух частей: текста подсказки и адреса, который определяет координату перехода. Текст подсказки или заменяющий его рисунок располагается внутри элемента А, а адрес задаётся при помощи атрибута href. Гиперссылка обычно выделяется на фоне текста цветом или подчёркиванием. Вид указателя мыши обязательно изменяется, когда он помещается над гиперссылкой.

Существуют правила для определения размеров элементов. Размер, заданный обычным числом, выражается в пикселях. Например, задаётся длина горизонтальной линии в 100 пикселов:

<HR width = 100>

Горизонтальный размер может быть задан в процентном отношении к ширине окна браузера:

<HR width = 50%>

Если размер окна будет изменён, то изменится и длина горизонтальной линии.

Символ \* может использоваться для разделения пространства страницы в определённых пропорциях. Два способа определения фреймов ( атрибут cols создаёт фреймы с вертикальным делением), причём левый фрейм в данном случае всегда будет в три раза уже, чем правый.

<FRAMESET cols = ” 25%, 75% “>

<FRAMESET cols = “1\*, 3\* “>

## Управление цветом

Кодирование цвета используется для раскрашивания шрифтов, горизонтальных линий и фона, других элементов страницы. Цвета обозначаются английскими названиями или числовыми шестнадцатеричными кодами. Существует несколько атрибутов, значениями которых являются параметры цвета. Самый простой способ определить цвет – написать название цвета на английском языке. Так, например, задаётся красный цвет шрифта в элементе FONT:

Color = “red”

Допустимые названия цветов:

|  |  |  |
| --- | --- | --- |
| **Русское название** | **Английское название** | **RGB – код** |
| Аквамарин | Aqua | #00FFFF |
| Белый | White | #FFFFFF |
| Жёлтый | Yellow | #FFFF00 |
| Зелёный | Green | #008000 |
| Золотой | Gold | #FF0700 |
| Индиго | Indigo | #4B0080 |
| Каштановый | Maroon | #800000 |
| Красный | Red | #FF0000 |
| Оливковый | Olive | #808000 |
| Оранжевый | Orange | #FF0000 |
| Пурпурный | Purple | #800080 |
| Светло-зелёный | Lime | #00FF00 |
| Серебристый | Silver | #C0C0C0 |
| Серый | Gray | #808080 |
| Сизый | Teal | #008080 |
| Синий | Blue | #0000FF |
| Ультрамарин | Navy | #000080 |
| Фиолетовый | Violet | #EE80EE |
| Фуксиновый | Fuchsia | #FF00FF |
| Чёрный | Black | #000000 |

Но в общем случае цвет определяется RGB – кодом. Любой цвет в этом случае представляется как комбинация красного (R), зелёного (G), синего (B) цветов, взятых в определённых пропорциях. Доля каждой цветовой составляющей определяется интенсивностью цвета и выражается двуразрядным шестнадцатеричным числом. В десятичном исчислении эти числа соответствуют диапазону от 0 до 255. В Windows–95 это соответствует цветовому режиму монитора True Color (24-разрядное двоичное кодирование цвета).

В выше приведённой таблице видно, что для формирования стандартных цветов часто используются или крайние значения интенсивности базового цвета 00 и FF, или среднее значение 80. Выбрав цвет, можно увидеть его численные характеристики, и наоборот, задав численные значения, получить новый оттенок.

Ещё одно направление применения цвета: элемент HR, создающий горизонтальную линию, допускает использование ряда атрибутов. С их помощью линию можно превратить в цветной прямоугольник. Его можно использовать для разделения частей страницы или в качестве маркеров списка.

HTML – код, рисующий радугу на Web-странице:

<FONT color = “ red “> Радуга </ front>

<TABLE border = 0 width = 100% >

<TR><TD bgcolor = #FF3030 >K

<TR><TD bgcolor = FFD000 >0

<TR><TD bgcolor = #F3FF5F >Ж

<TR><TD bgcolor = #00FF00 >З

<TR><TD bgcolor = #6FD3F7 >Г

<TR><TD bgcolor = # 5F72FDF >С

<TR><TD bgcolor = # B568F4 >Ф

</ table>

Цветовая проработка Web-страницы является альтернативой многочисленных рисунков и позволяет обеспечить более быструю загрузку документа.

# Конструктор документов

Пора от технических деталей программирования перейти к более важным вопросам:

* Как составить сценарий гипертекстового документа?
* Как спроектировать внешний вид документа на экране?

## Приложение. Структура приложения

Когда гипертекст состоит из нескольких файлов, термин "документ" к нему уже не очень подходит. В дальнейшем будем называть документом однофайловый гипертекст. А для случая, когда файлов много, введем термин "приложение".

Итак, есть идея создать красивое, полезное и удобное для пользователя гипертекстовое приложение. С чего начать работу?

Можно, конечно, сразу писать теги, а там -- как получится. Сомнительно, что получится хорошо.

Лучше сначала решить, какой способ организации гипертекста больше всего подходит для поставленной задачи, то есть, подумать над общей структурой приложения.

## Цепочка

Это самая простая структура. Документы, составляющие приложение, просматриваются пользователем по порядку, как страницы в книге. Каждый документ оканчивается ссылкой на следующий. Пользователь двигается по цепочке "до упора", то есть до последнего документа. В последнем документе ссылки "вперед" уже нет. Теперь можно окончить работу или, пользуясь кнопкой браузера "Назад" ("Back"), пройтись по документам "вспять".

Цепочку можно изобразить такой схемой:



Пример цепочки

Для удобства работы с приложением, можно помещать на странице документа ссылки не только вперед, но и назад. В этом случае отпадает необходимость использовать навигационные кнопки браузера.

Ссылки-связи можно оформить в виде пиктограмм.

Пример красивой цепочки

Комментарии к последнему примеру.

* Работая с документом, очень легко увидеть программу, которая этот документ строит. В браузере для этой цели есть специальная позиция "В виде HTML" ("Page Source") главного меню "Вид" ("View"). Войдите еще раз в приложение "Пример красивой цепочки" и посмотрите программы документов.
* В приложении "Пример красивой цепочки" использована необычная фоновая иллюстрация (о фоновых картинках подробнее смотрите ниже в этом уроке). Посмотрите ее здесь. Это очень узкая и длинная GIF-картинка размером 1x1200 пикселов, то есть ширина картинки всего один пиксел, а высота -- 1200 пикселов. Паркет из таких плиточек выглядит как гамма цветовых горизонтальных полосок с нарастающей вниз плотностью цвета.

Если бы документы были длинными, то через некоторое время, прокручивая экран по вертикали, мы дошли бы до второго ряда плиточек и увидели бы, как темная горизонтальная полоска внезапно обрывается и начинается светлая (новый горизонтальный паркетный ряд). Это не происходит, потому что в примере документы короткие, а паркетная плиточка длинная. Длина 1200 пикселов выбрана в соответствии с разрешением экрана 1600x1200. Такой размер пользовательского экрана в пикселах -- пока еще большая редкость и встречается в основном на профессиональных графических станциях. Обычные компьютеры работают с разрешением экрана 640x480, 800x600 и 1024x768.

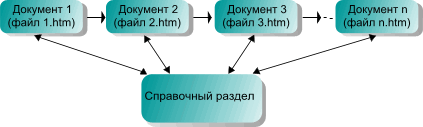
Картинки, подобные vert.gif, редко используют для фона страницы, ведь в длинных документах, переход от одного горизонтального ряда такого паркета к другому выглядит некрасиво и воспринимается как погрешность. Однако, узкие картинки с цветовой "растяжкой" HTML-программисты охотно используют для создания фона в таблицах (об этом разговор пойдет позже).

Гораздо чаще на практике используют паркет, состоящий из очень широких (по горизонтали) и коротких (по вертикали) плиточек. Пример такого паркета можно посмотреть здесь. Размер паркетной плиточки (картинка hor.gif): 1024x1.

В чистом виде цепочка используется проектировщиками очень редко. Обычная практика -- это цепочка документов с дополнительным набором HTML-файлов, образующих справочный раздел. В этот раздел выносятся большие иллюстрации, длинные примеры, комментарии, толкование терминов, литературные источники, лирические отступления.

Таким образом, дополнительно к ссылкам движения по цепочке, в приложении появляются ссылки, уводящие пользователя на документы справочного раздела. Изучив дополнительный материал, пользователь возвращается в цепочку и продолжает последовательное движение.

Схема цепочки со справочным разделом выглядит так:



Возможны различные варианты организации справочного раздела:

* Раздел состоит из нескольких независимых документов.
* Некоторые документы раздела имеют ссылки друг на друга или даже образуют цепочку.
* Справочный раздел состоит из одного документа с метками, на которые настроены ссылки в документах цепочки.
* Каждый документ в цепочке имеет свой собственный справочный раздел, и эти разные разделы могут содержать ссылки друг на друга.

### Применение цепочек

Цепочки, несмотря на свою простоту, часто используются в гипертекстовой практике. Особое значение эта схема имеет на большом информационном объеме. Материал разделяется на последовательные фрагменты, которые и образуют цепочку.

Гипертекстовый учебник, который вы сейчас читаете, тоже построен в виде цепочки. Об этом однозначно говорят навигационные кнопки в конце каждого урока. И хотя учебник имеет оглавление, по ссылкам которого можно напрямую попасть в любой раздел, предполагается, что читается он по порядку, урок за уроком.

## Иерархия

Иерархия -- это зависимость по подчинению, или включению одних объектов в другие.

Примером иерархии служит система классификации животных. Животные подразделяются на типы, каждый тип в свою очередь подразделяется на подтипы. Подтипы состоят из классов, классы из отрядов, отряды из подотрядов, подотряды из семейств. Семейства включают в себя роды, роды -- виды и, наконец, виды в этой системе классификации, состоят из отдельных представителей животного мира.

Другой пример иерархической структуры, более близкий нам по теме -- структура книги. Книга делится на тома, тома на части, части на главы, главы на параграфы, параграфы на пункты. В конкретной книге некоторые элементы из этой иерархии могут быть опущены (например, деление на тома).

Иерархическую структуру книги хорошо отражает оглавление (содержание). В правильно оформленных книгах оглавление помещают на первых страницах.

Особую роль оглавление играет в справочной литературе. Ведь такие книги не читают подряд от корки до корки, а обращаются к избранным страницам по мере надобности.

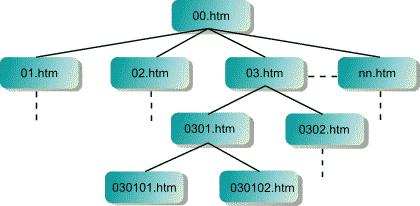
Оглавление оказывается полезным, даже если книга предназначена для последовательного чтения.

Оглавление:

* дает представление о книге в целом;
* позволяет быстро открыть нужный раздел для продолжения чтения;
* помогает найти отдельные, полюбившиеся фрагменты.

Иерархическое гипертекстовое приложение имеет такое же устройство, как и обычный бумажный справочник. Только строки в оглавлении являются гиперссылками, и не нужно вручную листать страницы.

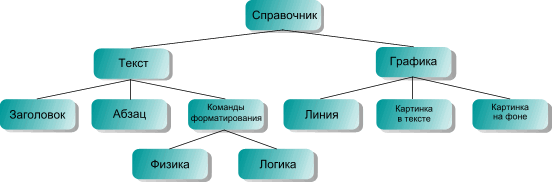
Схему иерархического приложения можно изобразить следующим образом:



Документ 00.htm (корень иерархии) содержит ссылки на разделы 01.htm, 02.htm, 03.htm, ..., nn.htm (ветви иерархии). Эти разделы, в свою очередь, содержат ссылки на подчиненные документы и так далее, вплоть до таких документов как 030101.htm и 030102.htm: они уже не имеют ссылок (листья иерархии).

Пример гипертекстовой иерархии

Схема этого примера выглядит следующим образом:



Обратите внимание на то, как реализована навигация в приведенном выше иллюстративном приложении. От корня к листьям пользователь продвигается по ссылкам, расположенным в основной части документа. Перемещаться назад он может, использую навигационные кнопки браузера, или меню, которое приводится в конце каждого документа и отделяется от основной части горизонтальной линией.

Меню позволяет попасть в любой пройденный документ напрямую, минуя последовательное перелистывание страниц кнопками браузера. Это, конечно, очень удобно. Обратные ссылки-возвраты настроены при помощи меток не на начало документа, а на те точки, из которых был выполнен прямой переход. Это предоставляет пользователю дополнительный навигационный комфорт.

Еще одна полезная навигационная идея: на первой странице иерархического документа представить ссылками полную его схему. Теперь пользователь имеет возможность выбирать нужный раздел справочника в самом начале работы.

Модифицированный иерархический справочник

В модифицированном справочнике на основной странице сформирован еще и гипертекстовый алфавитный указатель тегов. Это дополнение показывает заботу разработчика о пользователе.

## Уложим картинку в фоновый паркет

Теперь, после "высоких" сфер, одна приятная (но и опасная для новичков!) техническая подробность.

Пользуясь атрибутом **bgcolor** команды **BODY**, можно выкрасить фон документа в какой-нибудь один цвет.

Вместо одноцветной заливки можно использовать картинку. Эта картинка будет, повторяясь, укладываться в экранное поле наподобие паркетных плиточек.

Чтобы "паркет" смотрелся красиво, необходима визуальная стыковка плиточек между собой. Это надо учитывать при создании фоновой иллюстрации. И еще: очень важно, чтобы фон оставался именно фоном, а не отвлекал внимание пользователя от главных элементов документа, не мешал читать текст.

Фоновая картинка задается в команде **BODY** атрибутом **background**:

<**BODY** background=имя файла с картинкой>

Примеры



|  |  |
| --- | --- |
| Этот фон задается командой:  <**BODY** background=./pic/fon1.jpg>  Посмотреть на полном экране | Картинка, которая используется как фоновая плиточка: |



|  |  |
| --- | --- |
| Этот фон задается командой:  <**BODY** background=./pic/fon2.jpg>  Посмотреть на полном экране | Картинка, которая используется как фоновая плиточка: |



|  |  |
| --- | --- |
| Этот фон задается командой:  <**BODY** background=./pic/fon3.jpg>  Посмотреть на полном экране | Картинка, которая используется как фоновая плиточка: |



|  |  |
| --- | --- |
| Этот фон задается командой:  <**BODY** background=./pic/fon4.gif>  Посмотреть на полном экране | Картинка, которая используется как фоновая плиточка: |

Антипримеры



|  |  |
| --- | --- |
| Этот фон задается командой:  <**BODY** background=./pic/bfon1.jpg>  Посмотреть на полном экране  Плохая стыковка плиточек. | Картинка, которая используется как фоновая плиточка: |



|  |  |
| --- | --- |
| Этот фон задается командой:  <**BODY** background=./pic/bfon2.gif>  Посмотреть на полном экране  Очень плохая стыковка плиточек. | Картинка, которая используется как фоновая плиточка: |



|  |  |
| --- | --- |
| Этот фон задается командой:  <**BODY** background=./pic/bfon3.jpg>  Посмотреть на полном экране  **Берегите глаза!** | Картинка, которая используется как фоновая плиточка: |



|  |  |
| --- | --- |
| Этот фон задается командой:  <**BODY** background=./pic/bfon4.jpg>  Посмотреть на полном экране  **Берегите глаза!** | Картинка, которая используется как фоновая плиточка: |

Как видите, с фоновыми картинками нужно обращаться очень осторожно. Если нет уверенности, что паркет улучшает восприятие документа, лучше использовать одноцветную заливку: на гладком фоне текст всегда читается лучше.

# Графика

Иллюстрации в тексте радуют глаз и делают документ более привлекательным. Пора и нам записать в свой актив вывод картинок средствами HTML.

Браузеры "понимают" два графических формата GIF и JPG.

## Формат GIF

Картинки в этом формате поддерживают 256 цветов. В большинстве случаев этого вполне достаточно для получения качественных иллюстраций, но до фотографического качества, конечно, далеко.

GIF-формат имеет три приятные дополнительные возможности:

* *Мультипликация*. Используя анимационный GIF-редактор, легко собрать простой мультик: нарисовать отдельные кадры, настроить время задержки в последовательности показа, "зациклить кино" или показывать кадры только один раз. Когда анимация готова, она записывается на диск как обычный GIF-файл. Хотя в этом файле не одна, а целая серия GIF-картинок и разная настроечная информация.

В HTML нет различия между заданием вывода на экран простого GIF или анимированного. О том, что картинка "живая" -- браузер узнает в момент загрузки GIF-файла (с диска или сети) и поступает соответственно. Обычные картинки он просто размещает на экране и "забывает" про них. Анимированные ему приходится опекать постоянно, меняя кадры на экране в соответствии с программой, заложенной в самом GIF.

* *Прозрачная графика*. В GIF-формате один цвет можно объявить прозрачным. Это позволяет избавиться от строго прямоугольных иллюстраций и вписывать рисунок в документ более привлекательным образом. Прозрачный цвет задается в редакторе при записи GIF-файла на диск. Обычный GIF легко сделать прозрачным. Нужно просто загрузить его в GIF-редактор или PhotoShop и снова записать на диск, указав прозрачный цвет.
* *Чересстрочная развертка*. Применяется для больших GIF. Иллюстрация разделяется на четыре части, размером с оригинальную картинку. Первая часть содержит 1, 5, 9,... строки исходного изображения. Вторая -- 2, 6, 10,... Третья - 3, 7, 11,... Четвертая -- 4, 8, 12,...

Браузер строит на экране сначала первую часть картинки, потом вторую, затем третью и четвертую. Получается эффект постепенного проявления изображения. На медленных линиях Интернета этот метод позволяет пользователю еще до загрузки всех частей получить представление об изображении -- ведь части строятся браузером по мере их поступления на компьютер. Если картинка загружается не по сети, то различия вывода обычного GIF и чересстрочного на быстрых компьютерах практически незаметны.

## Формат JPG

Этот формат был разработан специально для передачи фотографий. Он поддерживает миллионы цветов и позволяет получать изображения очень высокого качества.

Конечно, высокое качество отражается на размерах файла. Но формат JPG имеет одну особенность: в JPG-редакторе, при записи на диск, можно указать ту степень качества, которая нужна и, тем самым, балансируя на планке "качество-размер файла", достичь нужного компромисса. В самом деле, если из миллиона цветов отсканированной фотографии оставить только 10 000, заменив остальные цвета на близкие -- на глаз такую подмену будет обнаружить практически невозможно, а размер файла существенно сократится.

JPG-формат поддерживает режим, похожий на чересстрочную развертку GIF. Но в отличие от GIF, деление на части происходит не по строкам, а по алгоритму выделения из картинки полноформатных частей. При этом каждая следующая часть по качеству выше предыдущей. JPG в таком режиме называется прогрессивным.

## Как программировать картинки

### Картинка в тексте

Картинку можно помещать в документ почти так же, как текстовый символ. Для вставки этого необычного знака нужно воспользоваться командой <IMG>. Команда не имеет парного закрывающего тега, но имеет много атрибутов.

**Атрибут src = имя файла**

Самым главным атрибутом команды является атрибут **src**, при помощи которого можно задать имя файла с картинкой. Например, команда <**IMG** src=img.gif> заставит браузер отобразить на экране графический файл img.gif из текущего каталога.

Обычно графические файлы не смешивают с HTML-текстами, а помещают в отдельный каталог pic, который является подкаталогом для каталога с программами (html-файлами). Тогда команда вывода графики будет иметь вид: <**IMG** src=./pic/img.gif>

**Атрибут alt = "текст надписи"**

Если браузер не находит картинки в указанном месте на диске, он вместо нее рисует на экране маленький прямоугольник и вписывает в него надпись, которая задана атрибутом **alt**:

|  |
| --- |
| <**IMG** src=monstr.jpg alt="Страшный зверь"> |

Атрибут **alt** рекомендуется использовать всегда. Даже если картинка найдена браузером, задание надписи не будет лишним: стоит пользователю остановить курсор на картинке, как надпись появится в маленьком окошке и сообщит дополнительную информацию. Проверьте это на следующей картинке:

|  |
| --- |
| <**IMG** src=./pic07/cat7.gif alt="А нас и тут неплохо кормят!"> |

**Атрибуты width = n и height = m**

Эти атрибуты задают ширину и высоту (в пикселах) прямоугольника, в который выводится картинка.

Если атрибуты не заданы, картинка рисуется в естественных размерах.

Когда размеры прямоугольника не совпадают с размерами картинки, браузер производит масштабирование картинки, подгоняя ее под указанные величины.

Рекомендуется всегда записывать эти атрибуты и указывать в них естественные размеры картинки. Если размеры указываются неверно -- картинка выводится браузером с искажением. Если указание размеров вовсе опущено -- на экране происходит маленькое землетрясение в момент прихода картинки из сети. Ведь, не зная размеров картинки, браузер оставляет под нее на экране стандартный прямоугольник, продолжая строить остальные элементы. Когда картинка загружена, браузеру приходится полностью перересовывать экран.

**Атрибут border = n**

Атрибут позволяет задать рамку иллюстрации толщиной в n пикселов.

При отсутствии атрибута, или при значении n=0 рамка не рисуется.

Ниже расположены ссылки на испытательные стенды. На них можно проверить работу атрибутов **src**, **alt**, **width**, **height**, **border**, как по отдельности, так и в совокупности.

**Атрибут align**

Этот атрибут позволяет определять положение иллюстрации по отношению к соседним элементам документа. Можно использовать следующие значения атрибута:

|  |  |  |
| --- | --- | --- |
| **align=top** |  | вертикальное выравнивание по верхнему краю |
|  |  |  |
| **align=middle** |  | вертикальное выравнивание по центру |
|  |  |  |
| **align=bottom** |  | вертикальное выравнивание по нижнему краю |
|  |  |  |
| **align=left** |  | горизонтальное выравнивание по левому краю |
|  |  |  |
| **align=right** |  | горизонтальное выравнивание по правому краю |
|  |  |  |

Давайте зададим небольшой текстовый абзац и разместим внутри него картинку:

|  |
| --- |
| <**P**>  Посмотрите на картинку.  <**IMG** src=./pic/4.jpg align=middle>  Красивая птичка!  </**P**> |

Теперь можно заглянуть в испытательный уголок и посмотреть, что происходит с изображением при разных значениях атрибута **align**.

### Картинка как ссылка

Очень легко заставить работать картинку как гиперссылку. Достаточно вложить команду **IMG** внутрь команды **A**:

|  |
| --- |
| <**A** href=переход><**IMG** src=файл></**A**> |

Картинка ведет себя так же, как текст. Курсор при попадании на картинку-ссылку меняет свою форму. Мышиный щелчок по картинке заставляет браузер выполнять переход. Отличие только в том, что картинка-ссылка не подчеркивается, как текст, и не выделяется цветом, а обрамляется в рамочку. Ниже приведен пример графической ссылки. Щелкните по ней мышкой и увидите команду, которая эту ссылку задает.

Для картинки с часами не задан атрибут **border**. Тем не менее, мы видим рамку. Эта рамка появляется потому, что картинка является гиперссылкой.

Часто рамка вокруг картинки нежелательна, даже когда она означает ссылку. Ведь о ссылке однозначно говорит изменение формы курсора, а если картинка -- прозрачный GIF -- рамка смотрится некрасиво. Избавиться от рамки, даже когда картинка -- ссылка, можно явным заданием **border=0**.

Эти ссылки заданы так:

|  |
| --- |
| <**A** href=05ex0401.htm><**IMG** src=./pic/auto.gif border=0  width=200 height=68  alt="Попробуй, догони!"></**A**>  <**A** href=05ex0401.htm><**IMG** src=./pic/auto.gif  width=200 height=68  alt="Эх, не догнать... Посадили в клетку!"></**A**> |

# Построение таблиц.

В практическом HTML-программировании таблицы совершенно незаменимы для...  
создания левых и правых полей страницы, выравнивания элементов на экране, многоколонной верстки, наложения картинок друг на друга. И, наконец, для отображения на экране табличного материала, но это -- в последнюю очередь. В 90% случаях таблицы используются, как ни парадоксально это звучит, не для отображения таблиц.

Но обо всем по порядку.

## Что такое таблица

Всем хорошо известна таблица умножения. Она представляет собой прямоугольник, расчерченный на клетки. Вертикальные ряды клеток образуют столбцы, горизонтальные -- строки. Каждая строка и столбец в таблице может иметь свой заголовок. Заголовки записываются в первой строке и первом столбце.

|  |  |
| --- | --- |
| Заголовками строк и столбцов в таблице умножения служат цифры от одного до девяти. Пользоваться таблицей очень просто. Если требуется узнать, например, результат умножения числа 6 на число 8, то нужно прочитать ответ в клетке на пересечении столбца с заголовком "6" и строки с заголовком "8". Или в клетке на пересечении строки с заголовком "6" и столбца с заголовком "8". Это все равно, потому что 6·8 = 8·6. В обеих этих клетках написано число 48. |  |

В общем случае таблицы используют для хранения информации с двумя характерными признаками. Значения одного признака расписано в заголовках строк, другого -- в заголовках столбцов. А в каждую клетку таблицы заносят значение, одновременно обладающее признаком строки и столбца, на пересечении которых она расположена.

Для таблицы умножения признаками служат значения сомножителей. Первый признак -- это значение первого сомножителя, второй -- значение второго.

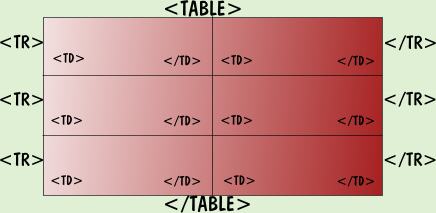
## Как задать таблицу

Таблица задается командой <TABLE> ... </TABLE>.

Внутри этих тэгов задаются строки командами <**TR**> ... </**TR**>.

Наконец, внутри строк задаются клетки (ячейки) командами <**TD**> ... </**TD**>.

Таким образом, по строкам, описывается вся структура таблицы.



Рассмотрим следующий пример.

|  |  |
| --- | --- |
| **Программа** | **Таблица на экране** |
| <**TABLE** border=1>  <**TR**> <**!--** Первая строка -->  <**TD**>(1,1)</**TD**> <**!--** Первая ячейка -->  <**TD**>(1,2)</**TD**> <**!--** Вторая ячейка -->  </**TR**>  <**TR**> <**!--** Вторая строка -->  <**TD**>(2,1)</**TD**> <**!--** Первая ячейка -->  <**TD**>(2,2)</**TD**> <**!--** Вторая ячейка -->  </**TR**>  <**TR**> <**!--** Третья строка -->  <**TD**>(3,1)</**TD**> <**!--** Первая ячейка -->  <**TD**>(3,2)</**TD**> <**!--** Вторая ячейка -->  </**TR**>  </**TABLE**> | |  |  | | --- | --- | | (1,1) | (1,2) | | (2,1) | (2,2) | | (3,1) | (3,2) | |

### Атрибуты команды TABLE

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Значение** | **Описание** |
| **align** | left, right | Выравнивание по горизонтали |
| **width** | число или процент | Ширина таблицы |
| **cellpadding** | число | Расстояние между содержимым ячейки и рамкой |
| **cellspacing** | число | Расстояние между ячейками таблицы |
| **bgcolor** | цвет | Цвет фона таблицы |
| **background** | файл | Фоновая картинка |
| **border** | число | Ширина линий рамки |
| **bordercolor** | цвет | Цвет линий рамки |
| **bordercolordark** | цвет | Цвет рамки (снизу и справа) |
| **bordercolorlight** | цвет | Светлый цвет рамки (сверху и слева) |

В приведенных ниже испытателях приводится более подробное описание действий атрибутов, а главное, на стендах можно пощелкать переключателями, подергать за "веревочки" и сразу увидеть, что из этого получилось.

### Атрибуты команды TR

HTML-таблица состоит из строк, каждая из которых задается командой

|  |
| --- |
| <**TR**>описание ячеек табличной строки</**TR**> |
|  |

Варьируя атрибутами команды **TR**, можно установить те или иные свойства одновременно для всех ячеек строки.

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Значения** | **Описание** |
| **align** | left, center, right | Выравнивание по горизонтали |
| **valign** | top, center,bottom, baseline | Выравнивание по вертикали |
| **bgcolor** | цвет | Цвет фона |
| **bordercolor** | цвет | Цвет линий рамки |
| **bordercolordark** | цвет | Цвет рамки (снизу и справа) |
| **bordercolorlight** | цвет | Светлый цвет рамки (сверху и слева) |

### Атрибуты команды TD (TH)

Каждая табличная строка состоит из ячеек, которые последовательно описываются командами **TD** (обычная ячейка) и (или) **TH** (ячейка-заголовок):

|  |
| --- |
| <**TH**>описание содержимого ячейки</**TH**>    <**TD**>описание содержимого ячейки</**TD**> |
|  |

Содержимое обычной ячейки по умолчанию горизонтально выравнивается по левому краю, а ячейки-заголовка -- по центру. Кроме того, текст в ячейке-заголовке выделяется полужирным шрифтом.

Команды **TD** и **TH** имеют следующие атрибуты:

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Значения** | **Описание** |
| **align** | left, center, right | Выравнивание по горизонтали |
| **valign** | top, center, bottom, baseline | Выравнивание по вертикали |
| **width** | число или процент | Ширина ячейки |
| **bgcolor** | цвет | Цвет фона |
| **background** | файл | Фоновая картинка |
| **bordercolor** | цвет | Цвет линий рамки |
| **bordercolordark** | цвет | Цвет рамки (снизу и справа) |
| **bordercolorlight** | цвет | Светлый цвет рамки (сверху и слева) |
| **nowrap** |  | Выключение автоматического разрыва строк |
| **colspan** | число | Ширина большой ячейки (в столбцах) |
| **rowspan** | число | Высота большой ячейки (в строках) |

## Многоликие таблицы

Как уже говорилось, использование таблиц в практике HTML-программирования очень неожиданное. Вероятно, авторы языка и не предполагали, что таблицы будут использоваться проектировщиками гипертекстовых страниц как элемент дизайна.

### Страничные отступы

В обычных книгах текст не начинается с самого края -- страничные поля выделяют информационные блоки и, следовательно, помогают глазам более комфортно воспринимать материал.

В гипертекстовом документе можно создать поля, если размещать информацию внутри таблицы, состоящей всего из одной клетки с невидимыми границами. При изменении значения атрибута **cellpadding**, меняется ширина полей.

|  |  |
| --- | --- |
| **Обычное размещение** | **Размещение в таблице** |
| Малютку сына - баю-бай! - Прижми покрепче к сердцу И никогда не забывай Задать ребенку перцу!  *"Алиса в стране Чудес"* | |  | | --- | | Малютку сына - баю-бай! - Прижми покрепче к сердцу И никогда не забывай Задать ребенку перцу!  *"Алиса в стране Чудес"* | |

Можно задать поля и при помощи пустых столбцов:

|  |  |
| --- | --- |
| **Размещение в таблице** | **Здесь border=1** |
| |  |  |  | | --- | --- | --- | |  | Баюкай сына своего Хорошею дубиной - Увидишь, будет у него Характер голубиный!  *"Алиса в стране Чудес"* |  | | |  |  |  | | --- | --- | --- | |  | Баюкай сына своего Хорошею дубиной - Увидишь, будет у него Характер голубиный!  *"Алиса в стране Чудес"* |  | |

Обычной практикой является комбинирование этих двух способов:

|  |  |
| --- | --- |
| **Размещение в таблице** | **Здесь border=1** |
| |  |  |  | | --- | --- | --- | |  | Баюкай сына своего Хорошею дубиной - Увидишь, будет у него Характер голубиный!  *"Алиса в стране Чудес"* |  | | |  |  |  | | --- | --- | --- | |  | Баюкай сына своего Хорошею дубиной - Увидишь, будет у него Характер голубиный!  *"Алиса в стране Чудес"* |  | |

Один из примеров табличных отступов вы видите сейчас на экране. В качестве страничного фона использована картинка с вертикальной надписью "Robotland" по левому краю. Для того чтобы элементы страницы не налезали на этот "корешок", все они выводятся на экран внутри таблицы так, как это показано в следующем примере.

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | Крокодильчики мои Цветики речные! Что глядите на меня Прямо как родные? | Это кем хрустите вы В день веселый мая, Средь нескушанной травы Головой качая?  *"Алиса в стране Чудес"* | |

### Замечание

Не рекомендуется моделировать страничный отступ при помощи таблиц для больших документов. Во-первых, прорисовка документа, "погруженного" в таблицу начинается только после его полной загрузки. Пауза перед чистым экраном действует на пользователя самым отрицательным образом. Во-вторых, браузеры Netscape на таких табличных страницах "теряют ориентацию" при возврате из гиперссылки. Пользователь щелкает на ссылке, браузер показывает новый документ, а точность возврата к старому месту уже проблематична.

### Многоколонная верстка

Используя таблицы, можно выводить на экран информацию в несколько колонок:

|  |  |  |  |
| --- | --- | --- | --- |
| При свете дрянного керосинового фонаря Остап прочел из путеводителя:   |  |  |  | | --- | --- | --- | | "На правом высоком берегу - город Васюки. Отсюда отправляются лесные материалы, смола, лыко, рогожи, а сюда привозятся предметы широкого потребления для края, отстоящего на 50 километров от железной дороги. |  | В городе 8000 жителей, государственная картонная фабрика с 320 рабочими, маленький чугунолитейный, пивоваренный и кожевенный заводы. Из учебных заведений, кроме общеобразовательных, лесной техникум." | |

### Наложение картинок

Еще одна необычная практика использования таблиц: построение коллажей.

*Постановка задачи*. Имеются две картинки:

Как их наложить друг на друга? Как поместить животное на экран телевизора?

*Общая идея*: первую картинку сделать фоном для таблицы, а вторую разместить внутри.

Построим таблицу, состоящую из одной клетки. Укажем для этой клетки ширину, равную ширине рисунка, который будем использовать как фон:

Да... Получилось не очень хорошо. Для задания высоты ячейки (и таблицы) создатели HTML не придумали атрибута. Мы видим только верхний кусочек фоновой картинки.

*Идея*: использовать еще одну вспомогательную картинку. По высоте она должна равняться фоновой, а по ширине должна быть такой, чтобы закрывать часть телевизора. Тогда лось, расположенный вслед за вспомогательной картинкой попадет в центр экрана.

Нарисовать вспомогательную картинку очень просто -- это должен быть одноцветный GIF, в котором единственный цвет объявлен прозрачным. Тогда пользователь на экране эту картинку и не увидит:

### Замечание

На самом деле рисовать невидимый GIF можно любой величины. Оптимальной является картинка в один пиксел. Ведь мы всегда можем указать экранные размеры в команде **IMG**, и наша вспомогательная картинка будет растягиваться до нужного размера. Искажений можно не опасаться -- их не будет, а кроме того, эту картинку нельзя увидеть на экране ни в какую лупу!

Для того, чтобы использование невидимой картинки empty.gif (из каталога pic) было более понятным, ниже она заменена на такую же однопикселную картинку empty1.gif, но без режима "прозрачности", а картинка с лосем показана с рамкой:

# Таблицы стилей

## Уровни CSS

Впервые каскадные таблицы стилей CSS были реализованы в браузере Internet Explorer 3.0. Однако в то время развитие CSS находилось в зачаточном состоянии, поэтому правила составления стилевых шаблонов были весьма разрозненными.

С момента своего возникновения структура CSS была несколько раз пересмотрена, в нее были добавлены новые элементы и убраны (видоизменены) старые. Существуют три уровня CSS, определяемых наличием завершенной редакции структуры. Это: CSS 1 (первый уровень структуры стилевых шаблонов, окончательно утвержденный 11 января 1999 года), CSS 2 (второй уровень стилевых конструкций, начало обсуждения которого датируется маем 1998 года) и CSS 3 (третий уровень стилевого оформления электронных документов, принятый к обсуждению 23 мая 2001 года, на момент написания книги находился в стадии проработки).

В завершение разговора об уровнях CSS следует добавить, что переход от одного уровня к другому, в основном, сопровождался некоторыми видоизменениями в структуре и в правилах стилевого оформления, технологическими дополнениями, а также попытками систематизировать применение CSS.

Именно третий уровень (CSS 3) позиционируется разработчиками в качестве некой единой системы представления стилей в электронном документе, основанной на использовании специальных модулей.

## Способы определения таблиц стилей

Как уже было сказано, любая таблица стилей CSS должна быть интерпретирована браузером для того, чтобы правила CSS, обозначенные для конкретных элементов электронного документа, вступили в силу.

Определение таблицы стилей (стилевого шаблона) возможно четырьмя способами:

*ссылка на внешний файл.* Если все стилевые шаблоны для конкретного HTML-документа разместить в одном текстовом файле (с расширением ess), то с помощью специального тега <link> из текущего документа можно сделать ссылку на внешний CSS-файл стилевых шаблонов, например:

<LINK REL="stylesheet" TYPE="text/css" HREF="style.css">

Браузер, анализируя HTML-код, обратится по указанному пути и, обнаружив указанный файл стилевого оформления, отобразит элементы страницы в соответствии с определенными правилами CSS.

Следует помнить, что конструкция указания пути к внешнему CSS-файлу должна находиться в пределах раздела head HTML-документа;

*внедрение в документ.* Под внедрением в документ подразумевается задание стилевой конструкции внутри самой HTML-страницы, например:

<STYLE TYPE="text/CS3">

<! —

BODY { font-family: Arial, Helvetica; }

INPUT { background-color: ICECECE; }

— >

</STYLE>

Данная конструкция также должна присутствовать в разделе head. Для браузеров, не поддерживающих CSS вообще или поддерживающих лишь отдельные правила стилевого оформления, описание шаблонов заключается между символами комментариев (при отсутствии поддержки CSS браузер пропустит содержание стилевых шаблонов, если же поддержка есть, то браузер интерпретирует правила CSS);

*включение в теговые конструкции.* Любой отдельный HTML-элемент может быть подвергнут форматированию средствами CSS. Для этого необходимо задать определенное правило реализации того или иного тега, например:

<Р ALIGN="justify" STYLE="color: 1000000; font-family: Verdana;"> Текст параграфа...

В этом случае задано отдельное правило для конкретного параграфа. Также можно присваивать отдельному HTML-элементу определенный класс стилевого шаблона:

<TABLE>

<TR>

<TD CLASS="header"X/TD>

<TD CLASS="text"X/TD>

</TR>

</TABLE>

Описание классов должно строиться следующим способом (на примере внедрения стилевого шаблона в документ):

<STYLE TYPE="text/css">

<! —

.header { font-weight: bold; color: gray; }

.text { color: black; font-size: llpx; }

-->

</STYLE>

В данном случае текст табличной ячейки класса .header будет отображаться жирным начертанием и серым цветом, а ячейки класса . text -обычным начертанием, черным цветом и размером шрифта 11 пикселов;

*импортирование.* Импортирование стилевого шаблона CSS, по сути, аналогично указанию ссылки на внешний файл:

<STYLE TYPE-"text/css">

<! —

@import: url(style.ess);

-->

</STYLE>

Все четыре способа определения стилевого шаблона CSS можно использовать одновременно в пределах одного HTML-документа (листинг 9.1). Такая возможность позволяет задавать основное правило CSS, к примеру, в виде внешнего файла шаблонов, а для исключительных или редких HTML-элементов — отдельные конструкции либо в теге <style>, либо в кодовых конструкциях самих тегов.

**Листинг. Совмещение различных способов определения стилевого шаблона CSS**

<HTML> <HEAD>

<Т1ТЬЕ>Совмещение различных способов определения CSS</TITLE>

<LINK REL="stylesheet" TYPE="text/css" HREF="style . css"> <STYLE TYPE="text/css"> <! —

P { text-align: justify; color: green; }

.title { color: blue; font-weight: bold; font-size: 16px; } — >

</STYLE> </HEAD>

<BODY BGCOLOR="#FFFFFF" TEXT="black" LINK="|OOFFOO" ALINK="IOOFFOO" VLINK="blue">

<FONT CLASS="title">Cnoco6bi определения шаблонов CSS</FONT>

Ниже перечислены существующие способы определения стилевых шаблонов CSS, даны характеристики каждого способа, приведены примеры их использования.

<UL>

<Ы>Ссьшка на внешний файл

<Ы>Внедрение в документ

<Ы>Включение в теговые конструкции

<Ы>Импортирование

</UL>

</BODY>

</HTML>

Также следует сказать, что использование каждого способа определения стилевых шаблонов CSS может быть связано с некоторыми минусами.

В случае ошибки интерпретации HTML-кода браузером, плохой связи с сервером и пр. внешний файл CSS может не загрузиться, вследствие чего стиль для нужных элементов HTML не будет переопределен.

Если внешний файл CSS включает слишком большое количество стилевых шаблонов (что отражается на конечном размере файла), то существует вероятность того, что браузер не сумеет полностью интерпретировать файл CSS или вообще исчерпает лимит времени по загрузке данных. В первом случае стили для части элементов не будут переопределены (браузер успеет "обнаружить" только те правила, которые размещены в верхней части CSS-файла). Во втором случае все элементы страницы останутся без изменения, загрузившись по умолчанию.

При использовании способа включения стиля в сам документ наличие слишком большого количества шаблонов CSS заметно увеличит конечный размер HTML-страницы, что скажется на времени загрузки документа в браузере.

CSS *и Dynamic HTML — просто и привлекательно! 187*

При сочетании различных способов определения стилевых шаблонов следует учитывать особенности браузеров. К примеру, Netscape "плохо относится" к использованию символа нижнего подчеркивания ( \_ ) в указании классов для элементов HTML (.news\_titie, .\_about и т. д.). Также ряд браузеров (и Netscape в том числе) не подключает стилевой шаблон класса, присвоенного ячейке таблицы <то>/<тн>: конструкция <то CLASS="text">TeKCT</TD> выведет текст по умолчанию, a <tdxfont CLASS="text">TeKCT</FONT></TD> отобразит текст заданным стилем.

## Запись шаблона CSS

### Группировка и наследование

Любое правило таблицы стилей CSS состоит из селектора и определения шаблона.

Селектор выступает в роли указателя стилевого правила для определенного HTML-элемента или внутреннего класса (идентификатора). Определение шаблона — это описание стилевых правил для обозначенных элементов HTML. Правила чередуются через точку с запятой и заключаются в фигурные скобки.

НЗ { color: blue; font-family: Tahoma, Verdana, Arial; }

В данном примере селектором является элемент заголовка нз, для шаблона которого следует такое определение: цвет — синий, шрифт -- Tahoma, либо Verdana, либо Arial.

Как видно из примера, для одного селектора приведено описание, содержащее два правила — по цвету заголовка и наименованию гарнитуры. Это говорит о том, что CSS позволяет группировать несколько стилевых правил для одного селектора в рамках единого описания шаблона.

Сравнив запись вида:

НЗ { color: blue; }

НЗ { font-family: Tahoma, Verdana, Arial; }

И

НЗ { color: blue; font-family: Tahoma, Verdana, Arial; }

можно сделать вывод о том, что группировка правил по селектору позволяет, во-первых, экономить размер CSS-файла, во-вторых — систематизировать структуру описания шаблона.

Другой особенностью таблиц стилей CSS является свойство наследования стилевых правил для нескольких селекторов одновременно, например:

TD, ТН, Р, DIV { text-align: justify; color: gray: font-size: Юрх; }

Такая запись назначает единый стиль отображения текстовой информации для элементов ячейки таблицы (<то>, <тн>), а также параграфов (<р>) и блоков (<div>), а именно: тивыравнивания — по ширине, цвет — серый, размер шрифта — 10 пикселов.

### Селекторы

В качестве селектора CSS могут выступать:

О *элементы HTML.* Переопределение стиля для конкретного элемента страницы:

BODY { color: orange; }

В этом случае весь текст в пределах раздела body будет оранжевым. При добавлении, например, таблицы назначение стилевого шаблона пропадет для текста внутри ячеек;

*классы.* Использование классов позволяет переопределять стиль как для конкретного элемента, так и для любого элемента, которому присвоен данный класс. Наименование класса начинается с точки и обычно пишется строчными буквами (допускается использование латинских букв и цифр, но наличие специальных символов, нижних подчеркиваний и прочих нестандартных элементов не рекомендуется).

.red { color: red; }

В этом случае любой элемент HTML, позволяющий менять цвет, будет отображаться красным, если ему присвоить класс . red:

<FONT CLASS="red">TeKCT красным цветом</ГОЫТ>

ИЛИ

<HR CLASS="red">

Если мы дополним селектор класса наименованием конкретного HTML-элемента, то действие стилевого правила будет распространяться только на данный элемент:

HR.red { color: red; }

При указании классов стилевого шаблона следует внимательно следить за тем, поддерживает ли HTML-элемент присваемый типереопределения стиля. Например, запись вида:

HR { text-align: justify; }

будет бессмысленной, т. к. горизонтальный разделитель относится к области структурного форматирования и не может содержать текст, который, согласно стилевому правилу, следует растянуть по ширине;

*идентификаторы.* Запись идентификатора начинается с символа # (диез) и заканчивается наименованием:

fblack { background-color: black; }

Например, присвоив данный идентификатор тегу то, мы получим ячейку таблицы, залитую черным цветом:

<TD 1П="Ыас1<">Ячейка черного цвета</ТО>

Сравнив функции селектора класса и идентификатора, можно задаться вполне закономерным вопросом — чем же отличаются эти селекторы? Действительно, формат определения селектора обоих типов аналогичен по структуре и присвоению HTML-элементам. Однако селектор идентификатора часто применяется для задания уникального имени элементу, который задействован в программном сценарии (скрипте). В отличие от него, селектор класса ограничивается, в основном, применением в стилевых шаблонах.

В заключение необходимо обратить особое внимание на невозможность сочетания селекторов различных типов. Нельзя одновременно переопределить стиль для стандартного элемента HTML и для него же, но по конкретному классу/идентификатору.

### Псевдоклассы

Псевдоклассами называют определенные условия форматирования HTML-элемента, в соответствии с которыми браузер подставляет необходимые стилевые правила отображения данных. При этом в исходной структуре электронного документа такие классы не присутствуют, они создаются в процессе интерпретации HTML-кода браузером.

В основном, псевдоклассы предназначены для задания различных типов форматирования по нескольким разновидностям элементов. Рассмотрим функциональность псевдоклассов на примере гиперссылок.

Согласно спецификации HTML и стандарту CSS, гиперссылка может принимать четыре состояния: непосещенная ссылка (link), посещенная ссылка (visited), активная ссылка (active) и ссылка при наведении курсора мыши

(hover). **Первые три СОСТОЯНИЯ** (link, visited, active) **обычно ПрОПИСЫ-**

ваются в теге <body> HTML-документа (уровень CSS 1). Четвертое состояние (hover) относится к уровню CSS 2 и подразумевает изменение цвета ссылки при наведении на нее курсора мыши пользователя (событие работает в браузерах Internet Explorer 4 и выше, Opera 5 и выше, Netscape 6 и выше, Mozilla 1.0; браузеры более ранних версий, а также некоторые не особенно распространенные браузеры не поддерживают состояние hover).

Эти состояния и будут являться псевдоклассами при записи правил отображения гиперссылок в стилевом шаблоне:

A:link { color: blue; }

A:active { text-decoration: underline; }

A:visited { color: gray; }

A:hover { color: orange; }

В данном случае все присутствующие в электронном документе гиперссылки будут отображаться в соответствии с заданным стилевым правилом. Однако часто возникает необходимость визуально выделить одни ссылки относительно других. Для этого наряду с псевдоклассами используются обычные селекторы классов:

A:active.red { color: red; } A:hover.red { color: blue; } A:active.white { color: white; } A:hover.white { color: black; }

### Применение таблиц стилей CSS

Прежде всего следует отметить, что при определении стилевых таблиц далеко не всегда свойства стандартного HTML-элемента соответствуют описанию шаблона стиля. Например, в HTML для жирного начертания используется тег-контейнер <в> (<strong>), а в CSS — конструкция font-weight: bold; (для элемента или селектора). Для выделения текста подчеркиванием в HTML предусмотрен тег <и>, а в CSS используется запись вида

text-decoration: underline; И Т. Д.

В рамках данной книги, посвященной применению языка разметки HTML, сложно рассказать обо всех свойствах переопределения стиля с помощью CSS. Поэтому мы остановимся лишь на некоторых аспектах использования каскадных таблиц стилей, а именно: на форматировании текста, структурном форматировании и организации пользовательских форм.

### CSS в форматировании текста

CSS предоставляет разработчику электронных документов гораздо более широкий набор возможностей работы с текстовой информацией, нежели стандартный HTML. Помимо способов выделения текста (подчеркивание, курсив, жирное начертание, выбор гарнитуры и размер шрифта), с помощью средств CSS возможно изменять такие параметры, как межбуквенный и межстрочный интервал, тирегистра (строчные и прописные буквы) и многое другое.

### Единицы измерения в таблицах стилей

Все единицы измерения свойств элементов в CSS можно разделить на абсолютные и относительные (табл. 9.1).

Абсолютные единицы измерения используются в случае, когда известны характеристики того устройства, которое отображает информацию.

Относительные единицы измерения определяют масштаб форматируемого элемента относительно других элементов, что позволяет сохранить перво-зданность документа при выводе на передающее устройство, характеристики которого заранее не известны.

*Единицы измерения CSS*

**Абсолютные**

**Относительные**

in (дюйм ~ 2,5 см)

mm (миллиметр)

cm (сантиметр)

pt (пункт ~ 1/7 дюйма)

рс (пика =12 пунктов)

em (высота шрифта элемента)

ех (высота буквы х)

рх (пиксел)

% (процентное соотношение)

В табл. 9.2 приведены наиболее распространенные свойства форматирования текста в CSS.

*Свойства форматирования текста в CSS*

**Свойство**

font-family

font-size

font-style

font-weight

font-variant

text-decoration text-align text-transform letter-spacing

line-height Color Background-color

**Формат записи**

font-family: Tahoma, Arial;

font-size: llpx; font-style: italic; font-weight: bold;

font-variant: small-caps;

text-decoration: underline;

text-align: right;

text-transform: uppercase;

letter-spacing: lem;

line-height: 5mm; color: tFFFFFF;

background-color: white;

**Функция**

Выбор гарнитуры для отображения (допускается перечисление нескольких названий через запятую)

Размер шрифта

Выбор наклона текста (курсив)

Наличие/отсутствие жирного начертания

Преобразование строчных букв в прописные с изменением пропорций

Подчеркивание текста

Определение типа выравнивания текста

Выбор регистра букв Межбуквенный интервал

Межстрочный интервал Цвет текста Цвет фона текста

<FONT CLASS="text">

CSS предоставляет разработчику электронных документов гораздо более широкий набор возможностей работы с текстовой информацией, нежели стандартный HTML. Помимо способов выделения текста (<FONT STYLE="text-decoration: underline;">подчеркивание</ГОМТ>, <FONT ID="kursiv">KypCMB</FONT>, '<FONT STYLE="font-weight: bold;">жирное начертание</ГОЫТ>, <FONT STYLE= "font-family: Times New Roman;">выбор гарнитуры</ГОЫТ> и <SPAN CLASS="font">pa3Mep шрифта</ЗРШ>), с помощью средств CSS возможно <FONT С1АЗЗ="со1ог">изменять такие параметры</ГОЫТ>, как <FONT STYLE= "letter-spacing: Зрх;">межбуквенный</РОМТ> и межстрочный интервал, <FONT STYLE="text-transform: uppercase;">тирегистра</ЕОМТ> (строчные и прописные буквы) и многое другое.

</FONT> </BODY> </HTML>

### Структурное форматирование

Также каскадные таблицы стилей CSS привнесли много нового и заметно расширили функциональность форматирования структурных элементов электронного документа: р, div и пр.

В табл. 9.3 приведены наиболее распространенные свойства структурного форматирования в CSS.

***Таблица 9.3.*** *Свойства структурного форматирования в CSS*

list-style-image list-style-image: url(«bullet.gif");

**Свойство**

margin

border-width border-style border-color list-style-type

**Формат записи**

border-width: 20px;

border-style: solid;

border-color: gray;

list-style-type: square;

margin: Ipx 2px 3px 4px;

**Функция**

Ширина границы структурного элемента

Ти декоративного отображения границы элемента

Цвет границы структурного элемента

Тинумерованного или маркированного списка

Указание пути к графическому маркеру

Определение размера поля относительно верхнего, правого, ниж-

# JavaScript

JavaScript - язык подготовки сценариев, позволяющий сделать Web-страницы более интерактивными и функциональными. После его изучения вы сможете заниматься разработкой Web-приложений на качественно новом уровне. Для написания сценариев JavaScript обязательно требуется знание языка HTML. В этой главе вы получите первое представление о JavaScript еще до того, как начнете создавать свой первый сценарий.

Здесь приводится следующая информация:

Q важность изучения JavaScript; а история JavaScript;

а некоторые различия между языками подготовки сценариев и языками программирования; а инструменты, необходимые для написания сценариев JavaScript.

## Использование JavaScript

Присмотритесь к сайтам, которые вы посещаете. На каждой странице содержится текст и некоторое количество картинок. Быть может, есть часы, показывающие время, или бегущий текст в строке состояния браузера. Возможно, на сайте имеется форма, которую нужно заполнить. Если пропустить какую-либо графу анкеты, то появится сообщение об ошибке. На некоторых страницах встречаются движущиеся по экрану изображения или текст, изменяющийся при щелчке мыши.

Таким образом, вы можете наблюдать JavaScript в действии, а создать подобные эффекты может любой человек, знающий этот язык. Причем без особого труда.

***Рекомендация*** *Следует понаблюдать за тем, что создают с помощью JavaScript другие разработчики. Это отличный способ найти свежее решение.*

Прочитав эту книгу, вы не только освоите JavaScript (вместо того, чтобы просто копировать и вставлять в Web-страницу готовые сценарии), но и узнаете множество интересных идей, реализовать которые вы сможете сами - с помощью JavaScript.

## Важность изучения JavaScript

Полезно изучить JavaScript хотя бы из-за его широкого распространения в сети. Огромное количество Web-страниц сделано с использованием сценариев (по самым разным причинам), и сайты без них кажутся блеклыми и скучными. Каким бы интересным не было содержание сайта, некоторые посетители сразу захотят уйти. Конечно, нет смысла оспаривать важность текстового содержания для любой Web-страницы, однако использование JavaScript не только улучшит подачу материала, но и сделает вашу страницу более запоминающейся.

***Рекомендация*** *Запомните: единственное, что отличает пользующиеся успехом сайты от неудачных, - их внешний вид. Чем интереснее оформлен сайт, тем он популярнее.*

***Предупреждение*** *Остерегайтесь использования найденных в сети сценариев, предназначенных для копирования и вставки. Как правило, они не отличаются хорошим качеством и, что хуже всего, дублируются на тысячах сайтов. Изучение JavaScript избавит вас от такой практики и позволит создавать оригинальные запоминающиеся Web-страницы.*

Можно найти еще множество причин для изучения и использования JavaScript. Сценарий всего в несколько строк помогает посетителям сайта отыскать страницу, соответствующую их браузеру и установкам, либо автоматически подсчитать сумму колонок в форме заказа. Подобные «мелочи» не рассчитаны на внешний эффект, но они свидетельствуют о высоком уровне профессионализма разработчика и знании им этики деловых отношений.

## JavaScript в действии

Ознакомившись с этой главой, вы сможете наконец создать свой первый сценарий. Вы начнете осваивать азы JavaScript на конкретных примерах. На ваше рассмотрение предлагается следующее:

а основные понятия JavaScript, такие как синтаксис, разметка, комментарии и др.; а некоторые термины языка JavaScript и причины, по которым он относится

к объектно-ориентированным языкам программирования; Q создание и чтение сценария JavaScript; а использование JavaScript для вывода и ввода информации.

### Ввод/вывод информации с помощью JavaScript

*В этой главе приводятся образцы сценариев JavaScript. Тщательно проработайте каждый из них, вводя код с клавиатуры. Следует потренироваться на элементарных примерах, прежде чем переходить к более сложным.*

Исследование возможностей ввода и вывода информации - это не только замечательный повод научиться создавать сценарии JavaScript, обеспечивающие двустороннюю связь с пользователем, но и прекрасное начало вашего профессионального пути.

В вычислительной технике все основано на вводе и выводе данных. Без этого ничего не происходит. Текстовый процессор не выполняет никаких действий, пока пользователь не введет какую-либо информацию (символы, набираемые на клавиатуре), и лишь затем эта информация отображается на экране, распечатывается или сохраняется на жестком диске.

Теперь вы научитесь как вводить информацию с помощью JavaScript, так и выводить ее в виде разнообразных окон сообщений. (Если вам приходилось блуждать по сети дольше нескольких минут, то вам они уже встречались.)

На рис. 3.1, 3.3 и 3.5 показаны окна сообщений трех типов в браузере Internet Explorer, а на рис. 3.2, 3.4 и 3.6 - в браузере Netscape Navigator.

Все эти окна созданы на языке JavaScript:

a Alert (Предупреждение) - служит для вывода информации;

a Confirm (Подтверждение) - предназначено для вывода информации и позволяет пользователю сделать выбор в форме ответа Да/Нет на вопрос;

a Prompt (Запрос) - служит для вывода информации и позволяет пользователю ввести ответ с клавиатуры.

### Объекты, методы и свойства

Вы, вероятно, слышали о том, что JavaScript - объектно-ориентированный язык. Но что это означает? Чтобы это понять, вам следует ознакомиться с тремя терминами:

Q объекты;

а методы;

а свойства.

Сначала рассмотрим их в общих чертах. Чем дальше вы продвинетесь в изучении JavaScript, тем чаще вам придется ими пользоваться, так что более близкое знакомство оставим на будущее.

***Объекты***

Говоря простым языком, *объект* (object) - это какой-либо предмет. Подобно тому, как в реальном мире все одушевленные и неодушевленные предметы являются объектами (машины, собаки и пр.), объектами считаются и составляющие компьютерного мира.

Что касается JavaScript, его объекты находятся внутри браузера. Это, в частности, окно браузера, формы и их части, например кнопки и текстовые окна. В JavaScript также имеется собственная группа встроенных объектов, к которым относятся массивы, данные и т.д. Сейчас вам не обязательно фиксировать на этом внимание, поскольку все эти объекты будут рассмотрены позже. Пока вы должны усвоить лишь необходимые определения.

Именно благодаря наличию объектов язык JavaScript считается объектно-ориентированным. Язык организован вокруг объектов, а не действий, или, иначе говоря, ориентирован на данные, а не на логику. При объектно-ориентированном программировании первоочередное внимание уделяется объектам, с которыми производятся некоторые манипуляции, а не логическим правилам, необходимым для таких манипуляций. Преимуществом такого подхода является не только облегчение программирования (или написания сценария), но и в то, что каждое действие можно выполнить разными способами.

***Методы***

*Метод* (method) - это действия, которые может выполнять объект. В реальном мире у объектов тоже имеются какие-либо методы. Машины ездят, собаки лают, доллар покупается и т.д. В нашем случае alert () является методом объекта Window, то есть объект Window может выдавать пользователю какое-либо предупреждение в окне сообщений. Примерами других методов являются открытие и закрытие окон, нажатие кнопок. Здесь речь идет о трех методах: open (), close () и click (). Обратите внимание на круглые скобки. Они означают, что методы, в отличие от свойств, используются.

***Свойства***

У всех объектов имеются *свойства* (properties). Если вы и далее будете следовать аналогии с объектами реального мира, то обнаружите, что все предметы обладают какими-то свойствами: у машин есть колеса, а у собаки - шерсть. Что касается JavaScript, то у такого объекта, как браузер, имеется название и номер версии.

## Управление данными с помощью переменных

В предыдущей главе вы научились использовать JavaScript для отображения на экране окон сообщений трех типов, служащих для ввода-вывода информации. Однако примеры сценариев, которые вы видели, не отличались гибкостью. Вы всего лишь дополняли сценарий сообщениями, которые хотели вывести на экран. Любой текст, помещенный в окне запросов, исчезал после щелчка по кнопке выбора. В данной главе вы узнаете, как изменить эту ситуацию. Здесь рассмотрено использование значений и переменных в JavaScript.

### Значения в языке JavaScript

Самое ценное в нашем мире — это информация. Каждый фрагмент информации в JavaScript рассматривается как *значение* (value). Так как информация может быть очень разнообразной, имеются различные категории значений. Простейшие типы данных в JavaScript принято называть *основными типами* (primitive types).

Три основных типа данных:

а строка (string); а число (number); а булево выражение (boolean).

***Строки***

Возможно, самым распространенным типом данных является строка. *Строка -*связанный набор символов, включающий в себя буквы, знаки препинания и цифры. В JavaScript строки чаще всего представляют собой какой-либо текст:

Здравствуйте и добро пожаловать!

Кто вы?

Мой рост - шесть футов.

Строки, вставляемые в сценарий JavaScript, заключаются в двойные или одинарные кавычки, например:

"Здравствуйте и добро пожаловать!"

Причина использования двух типов кавычек заключается в том, что двойные кавычки (") могут содержаться внутри строки, заключенной в одинарные кавычки ('), и наоборот. Например:

"J'm 6 feet tall"

'"Кто вы?" - спросил он.'

Строка может и вовсе не содержать символов. В этом случае она называется пустой строкой и обозначается пустыми кавычками:

***Числа***

JavaScript воспринимает два типа чисел. Это *целые числа* (integer) и *числа с плавающей точкой* (floating-point number).

***Целые числа***

Включают в себя положительные целые числа, например *1,2,3,* отрицательные целые числа, например -1, -2, -3 и нуль - 0.

Хотя большинство чисел, используемых в JavaScript, записываются в десятеричной системе счисления, могут применяться также восьмеричная и шестнадца-теричная системы.

***Примечание*** *В книге используется только десятеричная система счисления.*

***Числа с плавающей точкой***

Числа с плавающей точкой представляют собой числа с дробной десятичной частью:

3.1415926535897932384626433832795

Либо это числа, выраженные в экспоненциальном виде:

3.76е2

***Примечание*** *При экспоненциальной записи числа символ <se» в верхнем или нижнем регистре означает «10 в степени»-.*

Число, начинающееся с нуля и содержащее десятичную точку, считается числом с плавающей точкой.

***Предупреждение*** *Число, начинающееся с нескольких нулей и содержащее десятичную точку (например, 005.5) воспринимается как ошибка.*

В табл. 4.1 приводятся примеры чисел, использующихся в JavaScript, чтобы вам проще было освоиться с ними.

***Большие и маленькие числа***

*Таблица 4.1. Примеры чисел, использующихся в JavaScript*

|  |  |  |
| --- | --- | --- |
| Число | Описание | Десятичный эквивалент |
| 91 | Целое число | 91 |
| 4.56е2 | Число с плавающей точкой | 456 |
| 0.001 | Число с плавающей точкой | 0.001 |
| 00.001 | Ошибка |  |
| 0.001 | Четыре равных числа | 0.001 |
| .001 | с плавающей точкой |  |
| 1е-3 |  |  |
| 1.0е-3 |  |  |

Числа, используемые в JavaScript, могут быть как очень большими, так и очень маленькими. Под очень большими подразумеваются величины до 10308 (единица с тремястами восемью нулями), а под очень маленькими - Ю~308 (нуль целых с тремястами семью нулями и единицей после запятой).

***Булевы выражения***

Булевы выражения отличаются от строк и чисел тем, что могут принимать лишь два значения: true (истина) и false (ложь).

*Булевыми эти выражения названы в честь английского математика Джорджа Буля (1815-1864).*

Ниже помещены примеры булевых выражений:

Собака лает = true

У собаки пять ног = false

К булевым выражениям вернемся в главе 6, где будут рассматриваться выражения и условия. Пока вам нужно просто знать, что такой тип данных существует и может использоваться в JavaScript.

***Особые типы данных: числа, неопределенные и неопределяемые выражения***

Помимо рассмотренных типов данных, встречаются еще несколько основных, менее очевидных типов. Далее приводятся четыре числовых значения:

а положительная бесконечность;

О отрицательная бесконечность;

а положительный и отрицательный нуль;

а несуществующее число (not a number - NaN) .

Имеются еще два специальных типа данных:

а неопределенный (null);

а неопределяемый (undefined).

Что касается первых четырех типов, то вам не придется работать с ними (не будете же вы вводить в программу бесконечное число). Под несуществующим числом понимается результат бессмысленной математической операции (например, деления на нуль). Бесконечностью считается значение, превышающее 10308 (например, результат возведения 10300 в квадрат), и оно также не имеет для вас особого значения (если только не свидетельствует об ошибке).

Неопределенный тип может иметь только одно значение - null. В данном случае это свидетельствует о полном отсутствии полезной информации или каких-либо данных.

Понять, что такое неопределяемый тип данных, несколько сложнее. Здесь тоже возможно только одно значение (undefined), являющееся чаще всего тем же, что и null. В наиболее экстремальных случаях undefined становится плохим признаком (особенно если выдается в виде результата посетителю страницы) и свидетельствует о недочетах вашего сценария.

### Переменные в языке JavaScript

Теперь, когда вам известно об используемых в языке JavaScript значениях, вы готовы к тому, чтобы войти в мир *переменных* (variable) JavaScript.

Переменные имеют огромное значение не только в JavaScript, но и во всех языках программирования. Без их использования трудно обойтись, а с их помощью вы сможете управлять всеми типами данных.

Переменная - это имя, присваиваемое ячейке памяти компьютера, которая хранит определенные данные во время исполнения сценария JavaScript. Возможно, это определение кажется несколько сложным, но на самом деле пользоваться переменными очень просто.

***Создание переменных***

В языке JavaScript переменные создаются довольно легко. Давайте вернемся к стандартному шаблону HTML, чтобы сразу ввести вас в курс дела:

<html>

<head>

<11Ь1е>Простая страница</(:И:1е>

<script language="JavaScript">

<!-- Маскируемся!

// Снимаем маскировку. -->

</script>

</head>

<body>

</body> </html>

Прежде всего нужно создать переменную. Это можно сделать двумя путями: объявив ее заранее или создав «на лету». Сначала будут рассматриваться переменные, объявляемые заранее. Позже в этой главе рассказывается, каким образом они создаются «на лету».

**Пример.** Чтобы объявить (создать) переменную, в языке JavaScript используется оператор var, вслед за которым указывается имя, которое вы хотите присвоить переменной. В данном примере объявляется переменная под названием msg:

<html>

<head>

<Ь1(:1е>Простая страница</{:1(;1е>

<script language="JavaScript">

<!-- Маскируемся!

var msg;

// Снимаем маскировку. -->

</script>

</head>

<body>

</body> </html>

*Оператор равноценен законченному предложению в английском языке. Более подробно речь об этом пойдет в седьмой главе.*

Вот что следует знать об именах переменных:

а в именах переменных можно использовать символы нижнего и верхнего регистра либо сочетание того и другого;

а до объявления переменной ее значением считается undefined;

Q имя переменной не может начинаться с цифры;

а в именах переменных недопустимы пробелы; если необходим разделитель, используется символ подчеркивания (\_);

а в именах переменных следует избегать символа доллара ($) поскольку он не воспринимается, браузерами Internet Explorer 3.02 (поддерживающим JScript 1.0) и Netscape Navigator 2.02;

а следует избегать использования имен переменных, отличающихся только символами верхнего и нижнего регистра (например, msg и MsG), поскольку JScript 1.0 не сумеет их различить.

Рассмотрим следующие примеры корректных имен переменных:

msg

Hello\_all Msgl Msg\_l

Приведенные ниже имена переменных либо являются недопустимыми, либо их следует избегать:

a Imsg - начинается с цифры;

Qhelloall- содержит пробел;

a var- зарезервированное слово JavaScript;

Q dollar$ - в имени содержится символ $;

а msg и Msg - не используйте такие имена в одном сценарии, поскольку различаются они только регистром.

**Пример.** Если вы хотите присвоить значение только что объявленной переменной, это можно сделать в той же строке:

<html>

<head>

<title>npocTaH cTparoma</title>

<script language="JavaScript">

<!-- Маскируемся!

var msg = "Добро пожаловать в мир переменных JavaScript!";

// Снимаем маскировку. -->

</script>

</head>

<body>

</body> </html>

## Массивы JavaScript и Escape-последовательности

В предыдущей главе вы научились пользоваться переменными. В этой главе вы узнаете о новой разновидности переменных, которые способны хранить в себе несколько значений, - о массивах. Здесь рассказывается:

а как пользоваться массивами;

а как создаются массивы;

а как создаются элементы массива;

а как пользоваться знаками переключения кода.

### Как пользоваться массивами

*Массив* (array) - мощное средство программирования для любого языка, в том числе в JavaScript. Массив позволяет сохранять несколько независимых значений в одной переменной. Обычно эти значения как-то связаны (например, названия дней недели). Преимущество массивов заключается в том, что правильное их использование значительно упрощает код и помогает избежать создания множества переменных с похожими именами.

Итак, посмотрим, как следует создавать массивы и пользоваться ими.

**Пример.** В первую очередь вам нужно создать массив:

<html>

<head>

<title>npocTan страница</Ь1(;1е>

<script language="JavaScript">

<!-- Маскируемся!

var days\_of\_week = new Array(7);

// Снимаем маскировку. -->

</script>

</head>

<body>

</body> </html>

Эта простая строка сценария приводит к нескольким различным последствиям:

а создается переменная days\_pf\_week;

а с помощью new Array () указывает, что новая переменная является массивом; а определяется размер массива (в данном случае - 7).

Таким образом, созданы семь пустых ячеек, или *элементов,* которым нужно присвоить некоторые значения.

Ввести значения достаточно просто, только не забывайте отслеживать, какое значение присваивается каждому элементу.

days\_of\_week[x] = *значение;*

В данном случае х означает номер элемента.

Прежде всего обратите внимание, как JavaScript нумерует элементы. Он рассматривает первый элемент массива не как первый (с номером 1), а как нулевой (с номером 0). То есть в данном примере дням недели будут соответствовать порядковые номера от 0 до 6, а не от 1 до 7.

**Пример.** Итак, чтобы присвоить соответствующее значение первому элементу массива (в данном случае это понедельник), выполняются следующие действия:

<html>

<head>

<title>ripocTaH страница</(;1<:1е>

<script language=" JavaScript">

< ! - - Маскируемся !

var days\_of\_week = new Array (7), •

days\_of\_week[0] = "Понедельник";

// Снимаем маскировку. -->

</script>

</head>

<body>

</body> </html>

**Пример.** Следуя этому образцу, вы можете заполнить массив соответствующими значениями:

<html>

<head>

<title>npocTafl CTpaHHt[a</title>

<script language="JavaScript">

<!-- Маскируемся!

var days\_of\_week = new Array(7);

"Понедельник"; "Вторник"; "Среда"; "Четверг"; "Пятница"; "Суббота"; "Воскресенье";

days\_of\_week[0]

days\_of\_week[1]

days\_of\_week[2]

days\_of\_week[3]

days\_of\_week[4]

days\_of\_week[5]

days\_of\_week[6]

// Снимаем маскировку. -->

</script>

</head>

<body>

</body> </html>

Теперь вы можете извлекать из массива значения элементов. Как ранее элементам присваивались значения посредством имени переменной (в данном случае days\_of \_week) и указанного в квадратных скобках номера элемента (например, days\_of\_week[2]), так теперь этот формат используется для извлечения значений из массива.

**Пример.** Таким образом, чтобы вывести на экран значение третьего элемента в окне предупредительных сообщений, выполняются следующие действия:

<html>

<head>

<±1(;1е>Простая cTpamma</title>

<script language="JavaScript">

<!-- Маскируемся!

var days\_of\_week = new Array(7),•

days\_of\_week[0] = "Понедельник";

days\_of\_week[1] = "Вторник";

days\_of\_week[2] = "Среда";

days\_of\_week[3] = "Четверг";

days\_of\_week[4] = "Пятница";

days\_of\_week[5] = "Суббота";

days\_of\_week[6] = "Воскресенье";

alert(days\_of\_week[2]);

// Снимаем маскировку. -->

</script>

</head>

<body>

</body>

</html>

***Примечание*** *Не забывайте, что счет элементов массива начинается с нуля.*

**Пример.** Здесь представлен еще один способ, которым достигается тот же результат:

<html>

<head>

<title>npocT3H cTparama</title>

<script language="JavaScript">

<!-- Маскируемся!

var days\_of\_week = new Array(7);

days\_of\_week[0] = "Понедельник";

days\_of\_week[1] = "Вторник";

days\_of\_week[2] = "Среда";

days\_of^week[3] = "Четверг";

days\_of\_week[4] = "Пятница";

days\_of\_week[5] = "Суббота";

days\_of\_week[6] = "Воскресенье";

var x = 2;

alert(days\_of\_week[x]);

// Снимаем маскировку. -->

</script>

</head>

<body>

</body>

</html>

В данном случае создается переменная х и ей присваивается значение 2. Затем имя переменной х указывается в квадратных скобках, в результате чего из массива извлекается элемент с номером 2.

## Выражения, условия, операции, строки и числа

Из предыдущей главы вы узнали, как в JavaScript пользоваться переменными и массивами для сохранения информации. В этой главе вы закрепите усвоенный материал и научитесь работать с этой информацией. Здесь рассматриваются следующие понятия:

а выражения;

а условия;

а различные операции и способы их использования;

G преобразование строки в число и наоборот.

### Что такое выражения и условия

При изучении любого языка программирования (и JavaScript в том числе) время от времени приходится отвлекаться на попутные замечания. Давайте на минуту оторвемся от упражнений и рассмотрим новые термины.

***Выражения и условия***

С помощью переменных и массивов вы сохраняете в памяти компьютера любую нужную вам информацию в различных видах. Однако впоследствии вы можете сделать с этой информацией нечто большее, чем просто вывести ее на экран или поместить в окно предупредительных сообщений. Вы можете изменить ее, управлять ею или подвергнуть ее проверке. Для этих целей и предназначены выражения и условия.

*Выражения* (expressions) используются для комбинации двух или более значений, в результате чего получается третье, новое значение. Примером может служить сумма в следующем выражении:

1 + 2 = 3

Вы скомбинировали два значения (1 и 2), чтобы получить третье (3). Вот еще один пример:

3 + 3-1 = 5

Здесь комбинация трех значений (3,3 и 1) приводит к появлению нового значения (5).

*Условия* (conditions) позволяют сравнивать величины и определять логическое значение - true или false. Ниже приводится пример условия:

Лимоны желтые? Да.

Или другой пример:

3 + 3 = 6? Да.

Заметьте, что на оба этих вопроса можно дать только два ответа - да или нет (хотя на вопрос о лимонах можно ответить «наверное»). В языке JavaScript (и в других компьютерных языках) использование условий подразумевает только два результата - да или нет.

*Изучение JavaScript полезно еще и тем, что в процессе учебы вы получаете представление о других языках программирования. К тому же после освоения одного языка легче овладевать остальными.*

### Знакомство с операциями

В выражениях и условиях данные комбинируются с помощью *операций* (operator). Если манипуляции осуществляются с какой-либо одной величиной, то такая операция именуется *операцией с одним операндом* (unary operator). Если таких величин две, то операция называется *операцией с двумя операндами* (binary operator), а если три - *операцией с тремя операндами* (ternary operator).

Давайте рассмотрим наиболее часто используемые операции.

***Арифметические операции***

*Арифметические операции* - это всем известные математические действия:

а сложение (+);

1 + 3 = 4 а вычитание (-);

2-1 = 4 Q деление (/);

4/2 = 2 а умножение (\*);

2\*2 = 4 а остаток от деления (%).

9 % 5 = 4

Это были операции с двумя операндами.

Единственная арифметическая операция, нуждающаяся в пояснении, - остаток от деления. То есть 9 делится на 5 с остатком 4. Иногда эту операцию также называют взятием по модулю. При использовании целых чисел результат также будет представлять собой целое число. Но если речь идет о действительных числах (с плавающей точкой, не целых), в результате получится действительное число. Например:

5.5 % 2.2 = 1.1

Будьте внимательны при использовании этих операций, поскольку при некорректном их выполнении они могут привести к результатам NaN (несуществующее число) или Infinity (бесконечность). Примером такой проблемной операции является деление на нуль.

***Примеры сценариев JavaScript***

Далее описывается несколько примеров использования арифметических операций. Здесь приводится только содержимое блока SCRIPT. Все примеры созданы на основе шаблона.

**Пример.** Операция сложения (+) -.

<script language="JavaScript">

<!-- Маскируемся!

var а = 6, b = 4;

alert(a + b);

// Снимаем маскировку. -->

</script>

**Пример.** Операция вычитания (-) -.

<script language="JavaScript">

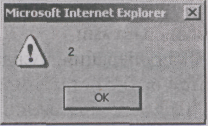
<!-- Маскируемся!

var а = 6, b = 4;

alert(a - b);

// Снимаем маскировку. -->

</script>



*Рис. 6.2. Использование операции вычитания*

**Пример.** Операция деления (/) -.

<script language="JavaScript"> <!-- Маскируемся! var a = 6, b = 3; alert(a / b);

***Операции сравнения***

*Операции сравнения* используются для сопоставления выражений. К ним относятся следующие операции:

а меньше (<);

5 < 6

а меньше или равно (<=);

1. **<= 6 или 6 <= 7  
   а больше (>);**
2. **> 4**

а больше или равно (>=);

5 >= 5 или 5 >= 4

а равно (==);

5 == 5

а не равно (!=).

5 != 3

Это были операции с двумя операндами.

В этих примерах сравниваются числовые данные, но те же самые операции могут выполняться и со строками (об этом речь пойдет позже). Единственное условие состоит в том, что нужно сопоставить величины, относящиеся к одному и тому же типу. В противном случае JavaScript попытается перевести данные из одного типа в другой, что не всегда удается. Чтобы избежать ошибок, сравнивайте данные только одного типа.

***Логические операции***

Принцип действия *логических операций* не так очевиден. Их функции станут понятнее, когда вы начнете использовать их с операторами, например с оператором if.

Далее перечислены три логические операции:

а логическое И (and);

&& а логическое ИЛИ (ог);

I I а логическое НЕ (not).

i

Логическое И и логическое ИЛИ - операции с двумя операндами, а логическое НЕ - операция с одним операндом. Они позволяют свести воедино результаты сравнений нескольких переменных.

Логическое И (&&) означает, что обе части выражения должны быть истинны. В качестве примера из жизни можно взять мысли водителя перед нажатием на педаль тормоза: машина едет слишком быстро && нужно затормозить.

Логическое ИЛИ (II) означает, что, по крайней мере, одна часть выражения должна быть истинной. Снова представим себе мысли водителя перед тем, как он включает фары: темнеет I I плохая видимость.

Логическое НЕ изменяет значение истина/ложь на обратное. Например, фары включают, когда ! светлеет (то есть когда темнеет).

*Возможно, что-то покажется вам непонятным. Не волнуйтесь - у вас будет множество возможностей отработать данные операции, и к тому моменту, когда вы перейдете к составлению настоящих сценариев JavaScript, вам все станет ясно.*

***Операции с одним операндом***

Как видно из названия, эти операции осуществляются с одной величиной. К ним относятся:

а префиксное и постфиксное возрастание (increment);

++

а префиксное и постфиксное уменьшение (decrement);

а унарный плюс;

+ а унарный минус.

Унарный минус изменяет знак выражения на противоположный. Из всех четырех операций это самая простая. Кроме того, унарный плюс используется не для смены знака, а для преобразования операнда в число (например, если это была строка).

Префиксное/постфиксное возрастание/уменьшение позволяет увеличить или уменьшить значение переменной на единицу. Однако результаты выполнения этих операций зависят от того, префиксная это операция или постфиксная.

Результат использования префиксной операции проще предугадать: если а = 5, то++а + 2 = 8, потому что значение переменной *а* было увеличено на единицу, прежде чем к нему прибавили число 2. В то же время

--а + 2 = 6,

потому что значение переменной а было уменьшено на единицу, прежде чем к нему прибавили число 2.

Постфиксные операции действуют иначе, поскольку возрастание или убывание производится только после использования старого значения в выражении. Таким образом, если а = 5, то а++ + 2 = 7, потому что увеличение на единицу будет выполнено после вычисления значения выражения. Это справедливо и для следующего выражения:

а-- + 2 = 6,

потому что уменьшение переменной а на единицу будет выполнено после использования в выражении прежнего значения.

## Создание сценариев с помощью функций и событий

До сих пор ваши сценарии JavaScript состояли из операторов, выполняемых по  
следовательно - от первого до завершающего. Это оптимальное решение для про  
стых сценариев, но в большинстве случаев при загрузке страницы выполнять сце  
нарий целиком не нужно. Например, вы хотите, чтобы несколько операторов были  
задействованы вначале, а остальные - в какой-либо другой момент. Именно тогда  
вам потребуются функции. %

В данной главе рассматриваются следующие вопросы:

а определение функции;

а применение функций;

а использование событий для вызова функций;

а обмен информацией;

а оператор return.

**Что такое функция**

*Функция* (function) - это группа операторов, предназначенных для определенной цели и объединенных под общим именем. Функция имеет следующий вид:

function имяфункции()

{

операторы; }

В начале функции помещается слово function, за которым указывается ее имя (например, yourMessage). После имени ставятся круглые скобки (your-Message () ). Их отсутствие приводит к ошибке.

***Примечание*** *Пока круглые скобки остаются пустыми. Чуть позже вы узнаете, для чего они используются.*

После круглых скобок идут открывающая и закрывающая фигурные скобки, между которыми помещаются операторы.

У каждой функции должно быть имя, причем имена функций, используемых на одной странице, не должны повторяться. Функция запускается (активизируется или вызывается) с Web-страницы.

**Ваша первая функция**

**Пример.** Теория может казаться вам слишком сложной, пока вы не перейдете к практике. Давайте сразу создадим первую функцию. Выполните следующие действия:

1. Откройте HTML-шаблон в текстовом редакторе.
2. Напечатайте в блоке SCRIPT слово function и дайте функции имя, например  
   yourMessage (не забудьте о круглых скобках в конце строки):

<script language=" JavaScript ">

< ! - - Маскируемся !  
function yourMessage ()

// Снимаем маскировку. --> </script>

3. Затем добавьте пару фигурных скобок:

<script language=" JavaScript ">

< ! - - Маскируемся !  
function yourMessage ()

// Снимаем маскировку. --> </script>

***Предупреждение*** *Помните, что в начале ставится открывающая фигурная скобка ({), а в конце - закрывающая (}). Распространенная ошибка новичков заключается в том, что они питают скобки местами, в результате чего сценарий не работает.*

4. Далее добавьте простой оператор:

<script language="JavaScript">

<!-- Маскируемся!

function yourMessage()

{

alert("Ваша первая функция!");

}

// Снимаем маскировку. -->

</script>

5. Сохраните результат, откройте его в браузере и посмотрите, что получилось.

Конечно, сначала вы ничего не увидите. Однако из этого вовсе не следует, что с вашим сценарием что-то не так. Причина в том, что вы еще не *вызвали* функцию. Это обязательно нужно сделать, потому что в отличие от простых операторов в блоке SCRIPT (запускаемых сразу) функции автоматически запускаться на выполнение не могут. Вам придется кое-что добавить в сценарий.

Функции в языке JavaScript могут вызываться несколькими способами. Самый простой, хотя и наименее эффективный - указание имени функции непосредственно в блоке SCRIPT:

<script language="JavaScript"> <!-- Маскируемся!

yourMessage();

function yourMessage()

{

alert("Ваша первая функция!");

}

// Снимаем маскировку. -->

</script>

Сохраните страницу еще раз и откройте ее в браузере, чтобы увидеть результат. Появление окна подтверждающих сообщений доказывает, что ваша функция была вызвана и выполнена.

Способ, с помощью которого вы вызвали функцию в этом примере, - не самый удачный. По сути, он ничем не отличается от простого перечисления операторов в блоке SCRIPT, освоенного ранее.

Предпочтительнее вызывать одну функцию из другой. Но, прежде чем вы научитесь это делать, вам предстоит узнать, каким образом выполняются функции и что такое *события* (events), поскольку именно они дают возможность управлять выполнением функций.

**События**

Необходимо сделать так, чтобы при вызове функций всю работу вместо вас выполняли события. JavaScript - *язык, управляемый событиями* (event-driven). To есть все происходящее в нем является результатом события или вызывает какое-либо событие. Открытие новой страницы в браузере, перемещение курсора, щелчок мыши - все это относится к событиям. В этой книге вам встретятся четыре их вида:

Q onLoad;

а onclick;

Q onMouseover;

Q onMouseout.

Давайте коротко рассмотрим каждый из них.

***Событие onLoad***

Это событие происходит после загрузки чего-либо, например после открытия страницы в окне браузера. Оно считается состоявшимся только после завершения загрузки всей страницы, включая изображения.

***Примечание*** *Событие, происходящее при выгрузке страницы (переходе на другую страницу или закрытии окна браузера), называется onUnload.*

Событие onLoad очень удобно использовать в сценарии, когда необходимо, чтобы функция выполнялась сразу после открытия страницы.

***Событие onClick***

Это событие происходит после щелчка мышью в определенном месте страницы. Позднее вы узнаете, что множество элементов страницы (гиперссылки, изображения, кнопки и пр.) могут реагировать на событие onClick. Его рекомендуется применять, когда вы хотите создать сценарий, взаимодействующий с пользователем.

***Событие onMouseover***

Это событие похоже на событие onCl ick, но происходит не после щелчка мышью, а после наведения курсора на определенный элемент страницы. Событие onMouseover можно связать практически с любым объектом Web-страницы (текстом, изображением, кнопками, гиперссылками и т.д.). Его также можно использовать, чтобы повысить уровень интерактивности сценария.

***Событие onMouseout***

Это событие подобно событию onMouseover, но происходит в тех случаях, когда курсор мыши отводится от объекта.

# Описание браузеров

## Просмотр WEB-страниц

Программы, необходимые для просмотра гипертекста – браузеры, которые используются не только как средство просмотра, но и как почтовые программы, а также как средство загрузки файлов посредствам FTP. Две основные функции браузеров: просмотр WEB-страниц и редактирования их содержимого (элементы HTML).

В мире создано немало программ для просмотра HTML-документов: Mosaic, Cello, Global Network Navigator, Opera и др. Наиболее популярными браузерами, которые обеспечивают корректный просмотр гипертекста, являются Microsoft Internet Explorer иNetscape Communicator.

## Microsoft Internet Explorer

Этот браузер фирмы Microsoft удобен первую очередь тем, что является полностью русифицированным. В данный момент наиболее распространены 4 и 5 версии этого браузера. Microsoft Internet Explorer (MSIE) настолько тесно интегрирована с Windows, что его трудно назвать автономной программой. Скорее, это один из компонентов операционной системы.

Работа с браузером может начинаться с подключения к Интернату или вестись автономно, если все необходимые файлы находятся на локальном диске (то есть, диске своего компьютера). В поле *Адрес* на панели инструментов указывается URL. Адрес предваряется кодом, который определяет, с какой подсистемой Интерната мы хотим работать. Например, если мы ищем сайт компании Microsoft, то адрес должен быть таким: http//microsoft.com.

Код **http** указывает на то, что программа должна работать с системой гипертекстовых документов и использовать соответственный протокол (HyperText Transfer Protocol). Но могут быть и другие варианты адреса. Например, адрес с кодом **ftp** указывает на архив файлов, доступных по FTP, код **mailto** обозначает адрес электронной почты, а код **file –** файл на собственном компьютере.

После указания URL браузер загружает данные из Интернета и демонстрирует гипертекстовой документ, находящийся по заданному адресу.

Найдя какую–нибудь организацию, вначале увидим ее «домашнюю» страницу (home page) . На ней есть подчеркнутые надписи – ссылки на другие страницы. Весь комплекс страниц (сайт) велик и его нельзя увидеть целиком. Гиперссылки позволяют передвигаться по страницам или частям одной страницы. Гиперссылкой может быть не только надпись, но и значок, рисунок или часть рисунка. Браузер имеет кнопки **«Вперед»** и «**Назад»**, которые позволяют вернуться к пройденным страницам. Каждой странице соответствует один или несколько файлов, которые браузер помещает в папку временного хранения (кэш).

Кнопка **«Остановить»** позволяет прервать загрузку страницы. Применять ее можно в нескольких случаях: 1) Когда страница еще загружается, 2) Когда на странице много графики и нет времени ждать, пока она вся загрузится, 3) Когда браузер пытается открыть несуществующие документы.

Кнопка **«Обновить»** используется редко. При изменении файла, чтобы увидеть как он будет выглядеть, можно только обновить документ.



Кнопка **«Назад»** позволяет перейти на страницу, которая была открыта ранее. Эта возможность есть не во всяком документе.



Кнопка **«Вперед»** обеспечивает движение по цепочке просмотренных документов вперед, если были выполнены переходы назад.



**«Домашняя страница»** (home page)- понятие условное. Каждая фирма, работающая в Сети, имеет свою домашнюю страницу. Адрес домашней страницы указывается в качестве параметра программы и может быть изменен.



Поиск в Интернете осуществляется путем перехода к странице поискового сервера.

Кнопка **«Избранное»** указывает каталог необходимых Web-страниц. Используя эту кнопку, можно добавить текущую страницу в список избранных страниц или дать команду браузеру загрузить любую из страниц, перечисленных в списке.



Кнопка **«Печать»** выводит на печать текущий документ.



Кнопка **«Размер»** изменяет размер шрифта на текущей странице.



Просмотр компонентов WEB-страниц, в некоторых случаях, вызывает открытие еще одного окна браузера. При помощи значков на панели задач необходимо закрывать ненужные окна, чтобы не перегружать компьютер.

WEB-страница часто оборудована средствами электронной почты. Другими средствами «обратной связи» являются формы. Заполнив соответственные поля и щелкнув кнопку «отправить» (которая может иметь в форме название **Go, OK, Send, Submit** или другое), можно переслать на сервер необходимую информацию. Так, например, происходит заказ товаров в Интернет-магазинах или заполнение анкет во время виртуальных опросов.

С WEB-страницами и их компонентами (например, рисунками) можно поступить, как с обычными документами: сохранять на диске под определенным именем, или распечатывать.

При помощи WEB-страниц с ярлыками можно существенно облегчить пользование Интернетом или локальными ресурсами. Собранные на странице ссылки по определенной тематике, превращают ее в аналог папки **Избранное**. Иными словами, можно легко сделать набор справочников со ссылками на интересные WEB-страницы.

Можно иметь и несколько папок для избранных страниц. Папка **Избранное** хранит не сами документы, а ссылки на них. Если страница находится не на локальном диске, то для ее просмотра потребуется подключение к сети. В некоторых случаях возникает необходимость иметь копии реальных документов на своем компьютере. На кэш в этом случае полагаться нельзя, так как он регулярно обновляется. Для сохранения просмотренных документов существует несколько команд. При этом копируется не только HTML-документ, но и рисунки.

Средства просмотра HTML-кода дают возможность сохранить документ и даже его модернизировать при помощи гипертекстового редактора Microsoft Front Page. Также можно просмотреть HTML-код, открыв страницу в «Блокноте» (Notepad). Редактировать страницу в режиме источника очень удобно, так как можно сразу увидеть результаты внесенных изменений: после модификации страницы ее надо сохранить в текстовом редакторе, а затем в браузере щелкнуть на кнопку **Обновить**.

Часто, при просмотре страниц возникает проблема со шрифтами. Если шрифт на странице слишком мелкий или наоборот слишком крупный, вид документа легко изменить. Используя команду **«Размер шрифта»,** чтобы изменить размер шрифта на время просмотра.

Основные инструменты настройки браузера собраны в окне **Свойства обозревателя**, который активизируется одноименной командой меню **Сервис**. На просмотр HTML-страницы влияют несколько параметров. На вкладке **Общее** имеются кнопки **Цвета**, **Шрифты**, **Язык** и **Оформление**, с помощью которых можно установить режимы воспроизведения станицы. В браузере назначаются для использования по умолчанию два шрифта. Два разных шрифта необходимы для прорисовки данных из соответственных элементов HTML. Какие именно шрифты надо применять, выберет пользователь. Важен также выбор кодировки: для русскоязычных документов применяются Windows-1251 или KOI-8. Можно выбрать и цвет, которым будет раскрашиваться текст, пройденные и неиспользованные гиперссылки.

На вкладке **Дополнительные** можно установить режим, при котором на странице воспроизводятся рисунки или мультимедийные компоненты.

На вкладке **Программы** можно выбрать приложения, которые будут использоваться по умолчанию совместно с браузером (HTML-редактор, программа электронной почты и др.).

## Netscape Communicator

Это комплекс программ (в настоящее время русифицирована версия 4.06) привлекает к себе внимание многими положительными свойствами. Его популярность во всем мире очень высока, и не только потому, что пользователи привыкли работать с его более ранними версиями. Интересными, например, представляются комментарии в строке состояния, которые дают представление о процессе загрузки страницы. Удачно организован доступ к параметрам. Браузер называется Netscape Navigator, а встроенный редактор гипертекста – Netscape Composer. В браузере реализована возможность создания закладок, что существенно облегчает поиск страниц. Несомненным удобством является легкость переключения режимов браузера и редактора.

Рассмотрим основные инструменты программы. Они, в основном, такие же, как и MSIE.

Кнопка **«Назад».** Возврат к странице, которая была открыта раньше текущей.

Кнопка **«Вперед»** Переход к странице, которая, была открыта позже текущей.



Кнопка **«Обновить»**. Повторение загрузки страницы.



Кнопка **«Домой»**. По умолчанию эта кнопка обеспечивает переход на домашнюю страницу Netscape, но пользователь может настроить программу на любую страницу, используя окно параметров.



Кнопка «**Мой Netscape».** Кнопка для перехода на страницу любителей Netscape. “Мой Netscape**”-** виртуальный клуб, в котором можно сделать свою WEB-страничку и получить ящик электронной почты.



Кнопка **«Печать»** используется для печати текущей страницы. В меню Файл имеются команды «Настройки страницы», «Предварительный Просмотр» и «Печать», которые позволяют настроить параметры и выполнить просмотр документа перед печатью.

Кнопка **«Стоп»** осуществляет остановку загрузки текущей страницы. Загрузка может быть продолжена после щелчка на кнопке «Обновить».



Кнопка с **логотипом** фирмы-производителя используется для инициирования загрузки станицы, но теперь такая команда не нужна. В данном браузере нажатие на логотип приводит на домашнюю страницу. Логотип снабжается анимацией, которая действует в течение загрузки текущей страницы.



В этом браузере, как и в предыдущем, есть поле **Адрес**. Но в отличие от MSIE, в раскрывающемся списке этого поля можно выбрать все недавно использующиеся адреса, а не только те, которые введены в строке вручную. Ещё один сервисный инструмент- **файл закладок**. Он является аналогом папки **Избранное** в Windows. Для доступа к закладкам на панели инструментов есть специальная кнопка.



На формат отображаемой страницы влияют команды меню «Вид». С его помощью устанавливается размер шрифта, кодировка страницы и другие параметры. Есть команда **«Остановить анимацию»**, то есть смену изображений в сложных рисунках формата GIF, для возобновления анимации необходимо перезагрузить страницу, используя кнопку **«Обновить»**.

Netscape Communication позволяет создавать варианты настроек программы (их ещё называют профили или конфигурации). Перед началом работы необходимо выбрать одну из конфигураций. Для каждого «пользователя» создаётся собственный файл закладок, журнал, кэш. Журнал связан с файлами кэш, так как при помощи журнала открываются страницы, которые когда-то были загружены из Интернета. Как журнал, так и кэш при необходимости можно очистить кнопкой в окне настроек.

В Netscape Navigator можно выбрать те же параметры просмотра, что и в MSIE: шрифты, используемые по умолчанию, цвета просмотренных и не просмотренных ссылок. Можно выбрать определённый фон для всех просматриваемых страниц или вообще отказаться от фона. Это удобно при разработке собственных страниц.

Браузеры позволяют загружать страницы без рисунков. Это ускоряет работу. Для отказа от изображения рисунков в NN надо сбросить флажок «Автоматическая загрузка изображений» в категории параметров «Дополнительные настройки».

Диалог для открытия файлов, расположенных локально, в MSIE и NN имеет особенность: используется два окна, а не одно, как в большинстве программ. Только во втором окне пользователь получает доступ к традиционному списку дисков, папок и файлов. Для задания адреса необходимой страницы, расположенной на жёстком диске, достаточно ввести путь к HTML-файлу в поле «Адрес».

В меню «Файл» есть команда «Сохранить как», которая используется для копирования текущей страницы в папку, выбранную пользователем. При этом можно выбрать формат файла: HTML или текстовый. В отличие от MSIE, который позволяет сохранить Web-страницу вместе с рисунком, NN копирует только основной файл страницы. Сохранение файла страницы требуется только в определённых случаях, так как вся страница автоматически записывается в кэш. Текстовый формат удобен в тех случаях, когда надо распечатать только текст страницы, не отягощая его элементами оформления и рисунками. Зато, если откроете страницу в редакторе Netscape Composer, то можно сохранить её в новой папке с рисунками.

# Практическая часть

# Терминология

Англоязычное программное обеспечение попадало в нашу страну разными путями, и в вычислительной технике складывалась уникальная лингвистическая ситуация. Существует огромное количество терминов, которые соответствуют по произношению английским словам. Программисты и электронщики разговаривают на своём, хорошо понятном языке. Заимствованные слова имели только одно значение, и это было удобно. Такая система прижилась. Любые попытки вводить отечественную терминологию не имели успеха. Яркий пример- название клавиш. Клавиатура так и осталась нерусифицированной. Подобные феномены объясняются тем, что до недавнего времени никто серьёзно не занимался локализацией фирменного программного обеспечения. Теперь положение изменилось. Появилось много программных продуктов, использующих русский язык. Возникли чисто русские термины: «папка», «карман», «ярлык», «пиктограмма»… В этих случаях теряется однозначность терминов, возникли двойные термины: Internet и Интернет.

В сфере гипертекстовых технологий в этом смысле обстоит иначе. Никто не называет браузер «просматривателем», а для «World Wide Web» есть красивое русское название «Всемирная паутина». Видимо, в каждом случае необходимо своё решение: использовать русское слово или заимствовать английское. Гипертекст породил много специальных терминов.

**Элемент** (element) - конструкция языка HTML.Это контейнер, содержащий данные и позволяющий отформатировать их определённым образом. Любая Web-страница представляет собой набор элементов. Одна из основных идей гипертекста - возможность вложения элементов.

**Тег** (tag) – начальный или конечный маркеры элемента. Теги определяют границы действия элементов и отделяют элементы друг от друга. В тексте Web- страницы теги заключаются в угловые скобки, а конечный тег всегда снабжается косой чертой.

**Атрибут** (attribute) – параметр или свойство элемента. Это переменная, которая имеет стандартное имя и которой может присваиваться определённый набор значений: стандартных или произвольных. Предполагается, что символьные значения атрибутов заключаются в кавычки, но некоторые браузеры позволяют не использовать кавычки. Это объясняется тем, что тип атрибута всегда известен заранее. Атрибуты располагаются внутри начального тега и отделяются друг от друга пробелами.

**Гиперссылка** – фрагмент текста , который является указателем на другой файл или объект. Гиперссылки необходимы для того, чтобы обеспечить возможность перехода от одного документа к другому.

**Фрейм** (frame) –этот термин имеет два значения. Первое- область документа со своими полосами прокрутки. Второе - одиночное изображение в сложном (анимационном) графическом файле (по аналогии с кадром кинофильма).

Примечание - вместо термина «фрейм» в специальной литературе и локализованных программных продуктах иногда можно встретить термин «кадр» или «рамка».

**HTML – файл** или **HTML –** страница – документ, созданный в виде гипертекста на основе языка HTML. Такие файлы имеют расширения htm или html. В гипертекстовых редакторах и браузерах эти файлы имеют общее название «документ».

**Апплет** (applet)- программа, передаваемая на компьютер клиента в виде отдельного файла и запускаемая при просмотре Web- страницы.

**Скрипт** или **сценарий** (skript) – программа, включенная в состав Web- страницы для расширения её возможностей. Браузер Internet Explorer в определённых ситуациях выводит сообщение: «Разрешить выполнение сценариев на странице?». В этом случае имеются в виду скрипты.

**Расширение** (extension) – элемент, не входящий в спецификацию языка, но использующийся, обеспечивая возможность создания нового интересного эффекта форматирования.

**CGI** ( Common Gateway Interface) – общее название для программ, которые, работая на сервере, позволяют расширить возможности Web- страниц. Например, без таких программ невозможно создание интерактивных страниц.

**Программный код** или **код** – аналог понятия «текст программы».

**Код HTML-** гипертекстовой документ в своём первоначальном виде, когда видны все элементы и атрибуты.

**World Wide Web, WWW** или **Web –** Всемирная паутина, распределённая система доступа к гипертекстовым документам, существующая в Интернете, HTML является основным языком для создания документов в WWW. Изучая его, мы изучаем часть этой системы, хотя область применения языка намного шире

**Web – страница –** документ (файл),подготовленный в формате гипертекста и размещённый в World Wide Web.

**Сайт** (site) – набор Web –страниц, принадлежащих одному владельцу.

**Браузер (**browser) – программа для просмотра Web – страниц.

**Пользовательский агент** (user agent) – браузер или другая программа, работающая на компьютере – клиенте.

**Загрузка (**downloading) – копирование файлов с сервера на компьютер – клиент.

**URL ( Uniform Resource Locator ) –** универсальный указатель ресурса – адрес некоторого объекта в Интернете. Типичный URL для WWW имеет вид: http:// www, название, домен/ имя файла.

Здесь название – это часть адреса, который часто употребляется для обозначения владельца сайта, а домен – обозначение крупного «раздела» Интернета: страны, области деятельности и т. д. URL используется для того, чтобы указать конкретную Web – страницу или графический файл в гиперссылках, а также везде, где требуется однозначно определить местоположение Web – страницы или файла.

**Базовый URL** – часть адреса, которая является общей для всех ссылок текущей Web –страницы.

**Базовый цвет –** каждый цветовой оттенок на экране монитора получается соединением трёх базовых цветов: красного, зелёного, синего.

**Цветовой канал** – интенсивность красного, зелёного, синего цвета на экране монитора. Цвет каждого пиксела определяется как комбинация этих трёх величин.

**Гамма – коррекция** – создание нелинейной зависимости между реальной интенсивностью базового цвета и полученной яркостью на экране монитора. Изображения, полученные без гамма – коррекции, оказываются более тёмными, так как монитор воспроизводит различные градации яркости не линейно по отношению к их числовому выражению. Изменение параметра «гамма» часто используют в графических редакторах при работе с изображением.

# Практическая часть

Листинг сайта:

<html>

<HEAD>

<TITLE>Школа №100</title>

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

<META content="" name=Description>

<META content="" name=Keywords>

<META content=Nikolay name=Author>

<SCRIPT language=JavaScript1.2

src="search/search.js"></SCRIPT>

<SCRIPT language=JavaScript1.2

src="search/searchEngine.js"></SCRIPT>

<STYLE>

p,span,form,input,hr,textarea{

font-family: Tahoma, Verdana, Arial, Sans serif;

font-size: 11px;

margin: 0px;padding: 0px;

vertical-align: middle;

}

.vmarg{margin-top:4px;}

.hmarg{margin-left:5px;margin-right:5px;}

.userData {BEHAVIOR: url(newhome0.htm)}

img {margin: 0px;padding: 0px;}

.checkbox{border:none;width:17px;height: 11px;vertical-align: middle;text-align: left;margin:2px;}

.b{font-weight:bold;}

.select{height: 21px; vertical-align:top;border:#A2B1C9 1px solid;background-color: #FFFFFF;}

.button{height: 21px; vertical-align:top; font-family: Verdana, Arial, Sans serif; font-size:10px;}

.but{height: 21px; vertical-align:bottom;font-family: Verdana, Arial, Sans serif; font-size:10px;}

input,textarea{border:#A2B1C9 1px solid;height: 21px; vertical-align:bottom;font-family: Verdana, Arial, Sans serif; font-size:11px;}

.head{background-color: #084D6C;}

.height1{height: 1px;}

.headline{background: #A7C1DD; padding: 0px; margin-top:2px;margin-bottom:2px;}

.headline\_m{padding: 0px; margin:0px;}

.headline1{text-align: left; color: #FFFFFF; font-weight: bold; margin: 2px; margin-left:3px;}

.text{text-align:justify;padding: 2px; margin: 2px; color: #26486E;}

a{text-decoration:none;}

a:hover {text-decoration:none;}

a.nav{color:#FFFFFF;}

a.nav:hover{color:#FF0000;}

a.copyr:visited{color:#074C6B;text-decoration:none;}

a.copyr:hover{color:#074C6B;text-decoration:underline;}

a.copyr{color:#074C6B;}

.w{color:#FFFFFF;}

.r{color:#FF0000;}

a.podrobnee{color: #AB7A2C; padding: 10px;}

a.podrobnee:hover{ text-decoration: underline; }

.mi1{padding: 4px; background-color: #5FA4DE;}

.mi{padding: 2px; background-color: #4D81BA; border-bottom: 1px solid White; padding-left: 4px; }

.mi2{padding: 2px; padding-left: 4px; }

.mlink{padding: 0px; background-color: #529ADA;}

a.al{

color:#FFFFFF;

background-image:none;

width:181px;

height:43px;

text-align:left;

font-weight:bold;

padding: 2px 5px 0px 10px;

border-bottom: 1px solid #FFFFFF;

}

a.al:hover{color:#771C00; background-image:url(but\_on00.gif); height:43px; width:180px;}

.ramka{margin: 3px; border: 1px solid #A2B1C9; background-color: #3273bb; text-align: center;}

.copyright{margin-left: 5px; color:#FFFFFF;}

div.win{

overflow: scroll;

overflow-x: hidden;

scrollbar-face-color: #306CAD;

scrollbar-shadow-color: #A7C1DD;

scrollbar-highlight-color: #A7C1DD;

scrollbar-3dlight-color: #306CAD;

scrollbar-darkshadow-color: #306CAD;

scrollbar-track-color: #ffffff;

scrollbar-arrow-color: #ffffff;

}

body{

scrollbar-face-color: #306CAD;

scrollbar-shadow-color: #A7C1DD;

scrollbar-highlight-color: #A7C1DD;

scrollbar-3dlight-color: #306CAD;

scrollbar-darkshadow-color: #306CAD;

scrollbar-track-color: #ffffff;

scrollbar-arrow-color: #ffffff;

}

</STYLE>

</head>

<body aLink=#006699 link=#000066 bgColor=#ffffff leftMargin=0 topMargin=0 marginheight="0" marginwidth="0" text="black" vlink="purple" style="skrolling:no">

<table align=center border=0 bordercolor=black border="0" cellspacing="0" cellpadding="0">

<tr>

<td colspan=3>

<table background="img\fon1.gif" border=0 >

<tr >

<td><img src="img\admin.jpg">&nbsp;</td>

<td width="100%" align=center> <img src="img\adnin1.gif" align=center> &nbsp;</td>

<td ><img src="img\flag1.gif" align=right>&nbsp;</td>

</table>

</td>

</tr>

<tr>

<td colspan=3 height="7" ></td>

</tr>

<tr>

<td colspan=3 background="img\fon4.gif" height="24">&nbsp;</td>

</tr>

<tr>

<td colspan=3 height="7" ></td>

</tr>

<tr>

<td background="img\fon1.gif" width="10" valign="top">

<a href=index.htm > <img src="img\home.gif" border=0></a><br>

<a href=01.htm ><img src="img\hystor.gif" border=0></a><br>

<a href=02.htm ><img src="img\peopls.gif" border=0></a><br>

<a href=03.htm ><img src="img\model.gif" border=0></a><br>

<a href=04.htm ><img src="img\obraz.gif" border=0></a><br>

<a href=05.htm ><img src="img\sou.gif" border=0></a><br>

<a href=06.htm ><img src="img\kontak.gif"border=0></a><br>

</td>

<td width="65%" valign="top" height="890" bgcolor="#FAF8F8">

<table width="100%">

<tr class="ramka">

<td> <p >&nbsp;</p></td>

</tr>

<tr>

<td><P id=sR align=justify>&nbsp;</td>

</tr>

</table>

</td>

</td>

<td height="100%" valign="top">

<table class="ramka" width="182" border="0" cellspacing="0" cellpadding="2" >

<tr height="18" align="right" >

<td valign="middle" width="100%" background="grad.gif" height="18"><p class="headline1">Опрос</p></td>

</tr>

<tr>

<td colspan="3" bgcolor="#ffffff" height="200" >

<div id="winnews" style="height:350;" >&nbsp;</p></div></td>

</tr>

<table class="ramka" width="182" border="0" cellspacing="0" cellpadding="2">

<tr height="18" >

<td valign="middle" width="100%" height="18"><p class="headline1">Найти</p></td>

<td width="18" height="1" height="18"></td>

</tr>

<tr>

<td height="100" colspan="2" bgcolor="#ffffff" >

<table border="0" cellpadding="0" cellspacing="0">

<div align="center"><center>

<FORM name=searchform onsubmit="find();return false;"> <TBODY>

<tr><td nowrap><p>Поиск по сайту</p><p><input style="width:143px" size="11" value="" name=searchbox maxlenght="100"></p><p align="right"><input onclick=find(); type="button" value="Найти"></p></td></tr>

<tr>

</td>

</tr>

</table>

</table>

</td>

</tr>

<tr>

<td colspan=3 height="7" ></td>

</tr>

<tr>

<td colspan=3 background="img\fon4.gif" height="24">&nbsp;</td>

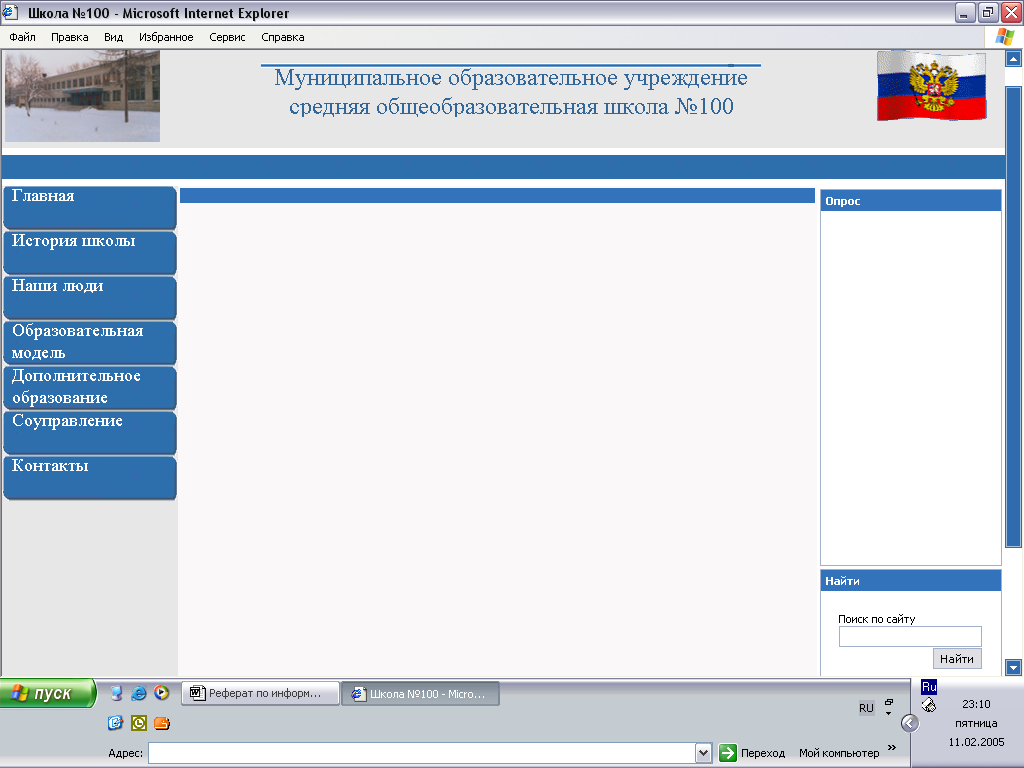
</tr>

<tr>

<td colspan=3 height="100"></td>

</tr>

</table>



Листинг скриптов:

* 1. search.js

<!--

function info(name,link,info) {

this.name=name;

this.link=link;

this.info=info;

}

var searchInfo=new Array();

searchInfo[0]=new info("Главная","index.htm","20.12 Игра Самый умный 6-7 классы 25.12 КВН 27.12 Конференция по борьбе против зависимостей 20.01 Сбор макулатуры (собрано 4 тонны) 11.02 Выборы президента совета школы В дальнейшем планируется 14.02 День Святого Валентина В программе: - концерт - дискотека - работа почты влюблённых 19.02 Смотр строя и песни 22.02 Игра Рыцарский турнир 5 7 классы 26.02 День семьи ");

searchInfo[1]=new info("история 1969 1977","01\_1.htm","I этап – становления, 1969 – 1977 годы, создание единого коллектива <br>педагогов, учащихся, родителей. Сформировались прочные традиции в эстетическом направлении образования, сложились творческие коллективы:<br> хор учащихся, ансамбль педагогов, школьный <br>театр, которые имели успех не только в районных конкурсах, но и городских. Как отмечали учителя школ № 22, 28 , где продолжали обучение дети, выпускники школы- восьмилетки № 100 отличались жизнерадостностью и взаимопониманием.");

searchInfo[2]=new info("История 1977 1997","01\_2.htm","II этап – функционирования, в 1978 году школа приобрела статус средней общеобразовательной.\ Это привело к дальнейшему развитию её потенциала, коллектив учителей пополнился новыми талантливыми педагогами, имеющими опыт работы в средней школе.\ Контингент учащихся определила специфическая социальная среда.\ В основном это были на 65% семьи рабочих второго поколения уралмашевцев и 35% семьи ИТР и интеллигенции. \ В тот период большинство родителей связывали судьбу своих детей с заводом и стремились дать им среднее или высшее техническое образование.\ Поэтому в школе традиционно огромное значение придавалось точным наукам.\В 1988 – 1989 г.г. школа вступила в полосу кризиса, в котором находилась вся советская система.\ Были утрачены былые ценностные ориентиры, которые стали подменяться ценностями материального обогащения любой ценой. \ Наряду с этим, постепенно начались процессы пробуждения и роста общественного самосознания.\ В 90-е годы XX века изменились целевые ориентиры нашего государства.\ Общество обращается к признанию позиции социальных групп и прав конкретного человека.\ Этим обусловлен следующий этап развития школы.\ В 1988 – 1989 г.г. школа вступила в полосу кризиса, в котором находилась вся советская система.\ Были утрачены былые ценностные ориентиры, которые стали подменяться ценностями материального обогащения любой ценой.\ Наряду с этим, постепенно начались процессы пробуждения и роста общественного самосознания. \В 90-е годы XX века изменились целевые ориентиры нашего государства. \Общество обращается к признанию позиции социальных групп и прав конкретного человека.\ Этим обусловлен следующий этап развития школы.\ II этап – поиска идей обеспечения прав личности в образовательном процессе через дифференциацию в обучении.\ В 1992 году принята I программа развития школы.\ Деятельность педагогического коллектива была направлена на обеспечение условий для становления ученика как личности, с разным уровнем способностей и потребностей, для реабилитации отстающих и предоставления возможности продвинутого обучения тем, кто способен учиться с опережением. В период до 1995 года была проведена жёсткая дифференциация по уровню обученности, созданы гомогенные классы: классы коррекционно-развивающие, компенсирующие. Математической и гуманитарной направленности, развивающего обучения по системе Эльконина Д.Б. и Давыдова В.В.\Однако впоследствии мы отказались от ранней жёсткой дифференциации, сохранив лишь специальные коррекционные классы для детей с задержкой психического развития. Так как в гомогенном классе ограничиваются возможности развития личности.\В 1996 году была разработана новая концепция и программа развития школы, где ведущей идеей стала идея адаптивного дифференцированного обучения. Особое внимание в этот период педагогический коллектив уделял конструированию урока в зависимости от склонностей, возможностей и способностей детей в классах смешанного типа. Осуществлялся поиск различных форм и методов обучения для максимального развития каждого ребёнка, апробировались в начальном и среднем звене технологии развивающего обучения. Была начата работа по освоению программы углублённого изучения математики и созданию УМК для классов гуманитарной направленности на завершающей стадии основной школы.\В 1997 году поступил заказ родителей будущих десятиклассников на открытие подготовительных курсов в ВУЗ на базе школы и был заключён договор с УГТУ-УПИ. \Данный этап развития школы характеризуется бурным ростом педагогического мастерства учителей. Методическая работа способствовала становлению статуса педагога как субъекта образовательного процесса, поскольку было направленно не только на повышение теоретических знаний и методических умений, но и на развитие самостоятельности, ответственности, рефлексивности (способности осмысливать свои действия, самопознание). ");

searchInfo[3]=new info("История 1998-2002","01\_3.htm","В 1998 – 1999 г.г. школа прошла процесс лицензирования и аттестации.\ Основные виды услуг зафиксированы в лицензии, и школа получила право на образовательную деятельность по программам углублённого изучения математики, коррекционным образовательным программам, наряду с общеобразовательными программами обучения.\Учитывая социальный заказ родителей и учащихся, в 2000 году был заключён договор с УГПУ на создание подготовительных курсов для классов гуманитарной направленности (9 – 11 кл.), в перспективе планируется совместная работа с УрГСХА (открытие профильных сельскохозяйственных 10 классов).\ Таким образом, в деятельности образовательного учреждения за эти годы произошли серьёзные целевые, содержательные, организационно-методические и управленческие изменения, на анализе которых мы остановимся ниже, и проводимые нами социологические исследования подтверждают наше предположение о том, что на пути традиционного, негуманного подхода к растущей личности не могутнаступить качественные изменения в системе образования, в том числе в работе нашей школы, благодаря которым достигался бы главный результат – воспитание свободной и ответственной, духовно и нравственно развитой личности.\Решение столь важной задачи возможно, по нашему убеждению, в учебном заведении, переведённого из условий стабильного функционирования в развивающиеся, работа которого направлена на развитие личности ребёнка в процессе духовного взаимодействия с постоянно совершенствующейся личностью взрослого.");

searchInfo[4]=new info("Дополнительное образование","04.htm"," В школе действует 18 детских объединений по следующим направлениям : - спортивное - художественно-эстетическое - Прикладное (театр модница затейливый крючок, сделай сам, юный редактор и юнный оформитель) - проедметные (эколлогия урала литертура урала занимательный французский логика речь ) - развивающие кружки (МДО) ");

searchInfo[5]=new info("Дополнительное образование Кружки","04\_1.htm","Театр Юный программист Танцы Хор ");

searchInfo[6]=new info("Дополнительное образование Секции","04\_2.htm","Волейбол Баскетбол Каратэ Киокушинкай ОФП ");

searchInfo[7]=new info("Дополнительное образование Платные услуги","04\_4.htm","Эти занятия с удовольствием посещают дети с 3 по 6 класс.\ На уроках они осваивают азы компьютерной грамотности. \Информатика помогает им развивать внимательность сосредоточенность, память и мышление.\ Ребята узнают , как возникают ошибки, как их предотвращать и разыскивать.\ Но самое главное : «Информатика учит их учиться!»");

// searchInfo[n]=new info("","","");

//-->

* 1. searchEngine.js

var msg="";

var yes=false;

var found=0;

var bold="";

var boldN="";

var resultsN="";

function find ()

{

var search= "";

search = document.searchform.searchbox.value;

if(search.length <3) { return;}

var searchFor=search.toUpperCase();

for(i=0;i<searchInfo.length;i++)

{

for(n=0;n<searchInfo[i].info.length;n++)

{

var results=searchInfo[i].info.substring(n,n+search.length);

if(results.toUpperCase()==searchFor)

{

yes=true;

bold=searchInfo[i].info.substring(0,40);

bold +=".....";

break;

}

}

if(yes==true)

{

found++;

boldN=searchInfo[i].name;

msg+= '<a href="'+searchInfo[i].link+'">'+boldN+'</a> - '+bold+'<br>';

yes=false;

}

}

var table=" <b> РЕЗУЛЬТАТЫ ПОИСКА: </b> Найдено <font size=+1 color=#00008B>"+found+"</font> ссылок "+ '<font color=#FF0000><b>' + search + '</b></font><br><br>';

if(document.all) document.all.sR.innerHTML =table + '\n' +msg;

if(document.layers)

{

//Здесь открыть новое окно, куда все и записать.

var neww = window.open("", "neww");

neww.document.write(table + '\n' + msg);

neww.document.close();

}

window.status="Поиск завершен.";

yes=false;

found=0;

document.forms[0].searchbox.value="";

resultsN="";

bold="";

msg="";

boldN="";

}

# Заключение

В этом проекте был рассмотрен язык разметки гипертекстовых документов HTML, его основные функции свойства и параметры. Сегодня применение HTML практикуется во всех без исключения электронных документах, независимо от тематики, величины и коммерческой направленности Интернет проекта.

В данной работе такие технологии, как CSS JavaScript, были затронуты лишь поверхностно дабы показать эффективность совокупности использования HTML с интерактивными скриптовыми технологиями. Использование последних является отдельной темой, подходящей для отдельного проекта.

# Список литературы

1. Айзекс А. Dynamic HTML BHV-Санкт-Петербург 1998
2. Ганчаров А. Самоучитель HTML. Питер 2000
3. Дарнелл Р. HTML 4 Энциклопедия пользователя ДиаСофт 1999
4. Денисов Internet Explorer 5 Справочник Питер 1999
5. Хоумер А. Dynamic HTML Справочник Питер 1999
6. Петюшкин А.В., HTML. Экспресс-курс. – СПб.: БХВ - Петербург, 2003
7. Кингсли-Хью Э., JavaScript: учебный курс. – СПб.: Питер, 2002
8. http://www.robotland.ru/