Владимирский государственный университет

ОСНОВЫ ВЕБ-ПРОГРАММИРОВАНИЯ

Методические указания к лабораторным работам по дисциплине «Программирование для Интернет и веб-дизайн»

Владимир 2005

Владимирский государственный университет

ОСНОВЫ ВЕБ-ПРОГРАММИРОВАНИЯ

Методические указания к лабораторным работам по дисциплине «Программирование для Интернет и веб-дизайн»

Владимир 2005

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Владимирский государственный университет
Кафедра физики и прикладной математики

ОСНОВЫ ВЕБ-ПРОГРАММИРОВАНИЯ

Методические указания к лабораторным работам по дисциплине
«Программирование для Интернет и веб-дизайн»

Составители:
А.Ю. ЛЕКСИН
Д.В. МИТРОФАНОВ

Владимир 2005

УДК 004.41

ББК 32.973.2-018

О75

Рецензент
Кандидат технических наук, доцент
кафедры управления и информатики в технических
и экономических системах
Владимирского государственного университета
Д.Н.Васильев

Печатается по решению редакционно-издательского совета
Владимирского государственного университета

Основы веб-программирования: метод. указания к лабор. работам по дисциплине «Программирование для Интернет и веб-дизайн» / сост.: А.Ю.Лексин, Д.В.Митрофанов ; Владим. гос. ун-т. – Владимир, Ред.-издат. комплекс ВлГУ, 2005. – 28 с.

Методические указания содержат описания семи лабораторных работ, выполняемых в рамках учебного курса «Программирование для Интернет и веб-дизайн», и призвано помочь в освоении базовых знаний и получении практических навыков в области сразу нескольких технологий веб-программирования: языков разметки, каскадных листов стилей, языка программирования клиентских приложений JavaScript и технологии серверного программирования ASP

Пособие предназначено для студентов специальности 010501 – «Прикладная математика и информатика», а также для преподавателей, проходящих обучение во внутривузовской системе повышения квалификации.

Табл. 1. Ил.3. Библиогр. 6 назв.

УДК 004.41
ББК 32.973.2-018

ВВЕДЕНИЕ

Одной из основных тенденций современного общества является быстрый рост роли информации во всех сферах человеческих отношений. Развитие и снижение стоимости средств связи привело к возникновению такого уникального информационного пространства как глобальная компьютерная сеть Интернет (зачастую ее сокращенно называют «Сеть»). В последние годы наблюдается лавинообразное увеличение числа пользователей Сети не только в высокоразвитых с точки зрения систем коммуникации государствах, но и в таких странах как Россия. В результате единое информационное пространство Интернета оказывает все большее влияние на человеческое общество. И для того, чтобы влияние это было управляемым, а развитие Сети проходило наиболее эффективным образом, актуальной является задача подготовки соответствующих специалистов в области веб-технологий (в качестве основных здесь выступают технологии веб-дизайна и веб-программирования).

Предлагаемое методическое пособие призвано помочь в освоении базовых знаний и получении практических навыков в области сразу нескольких технологий веб-программирования: языков разметки, каскадных листов стилей, языка программирования клиентских приложений JavaScript и технологии серверного программирования ASP. Представляемые семь лабораторных работ выполняются студентами специальности 0102 «Прикладная математика» в рамках рабочей программы по дисциплине «Программирование для Интернет и веб-дизайн», изучаемой в восьмом семестре. Все работы выполняются и тестируются в рамках локальной вычислительной сети с использованием учебного веб-сервера. Перед выполнением каждой работы обязательным является ознакомление со справочным материалом по соответствующей тематике, размещенным на веб-сайте курса.

Лабораторная работа №1

ЯЗЫК РАЗМЕТКИ HTML: ФОРМАТИРОВАНИЕ ТЕКСТА И СОЗДАНИЕ ГИПЕРССЫЛОК

I. Цель работы: освоение основ языка разметки гипертекста HTML, дескрипторов форматирования текста и технологии создания гиперссылок.

II. Теоретическое введение.

Язык разметки гипертекста HTML представляет собой определенный способ формирования информации, который указывает браузеру как следует отображать страницу. Это основное средство кодирования документов, передаваемых по глобальной информационной сети Интернет.

Базовые синтаксис и семантика HTML определены в стандарте, разработанном международной организацией World Wide Web Consortium (W3C), занимающей рассматривающей и стандартизующей все технологии, связанные с веб-средой. Спецификации уже принятых и проекты новых стандартов можно найти по адресу http://www.w3.org.

HTML-документ состоит из текста, образующего содержимое документа, и дескрипторов (тегов), определяющих его структуру и внешний вид. Каждый дескриптор представляет собой идентификатор, заключенный в угловые скобки. Дескрипторы могут быть одиночными и парными. В последнем случае они определяют некоторый структурный блок информации. Закрывающий дескриптор блока отличается от открывающего наличием косой черты перед именем дескриптора.

В соответствии со стандартом, содержимое HTML-документа заключается в пару тегов <HTML>…</HTML> и разделяется на заголовок (<HEAD>…</HEAD>) и тело документа (<BODY>…</BODY>). В заголовке дается название документа HTML и указываются другие параметры, которые браузер может использовать при отображении документа. Например, дескриптор <META http-equiv="Content-Type" content="text/html; charset=windows-1251"> приводит к включению в HTTP-ответ сервера поля заголовка Content-Type, значение которого будет указывать на использование кодировки windows-1251. Тело – это фактическое содержимое доку-

мента, в которое входят отображаемый текст и управляющие форматом отображения теги. Пример простейшего HTML-документа:

```
<HTTP>
<HEAD>
   <TITLE>Лабораторная работа №1</TITLE>
</HEAD>
<BODY>
   <P><B>I. Цель работы:</B> освоение основ языка
   разметки гипертекста HTML.
   </P>
</BODY>
</HTTP>
```

Помимо имени, практически все теги могут использовать набор атрибутов, управляющих способом интерпретации тега браузером. Атрибуты указываются в открывающем теге в виде пар «имя="значение"». Значение заключается в двойные или одинарные кавычки, но может и не сопровождаться подобными ограничителями. Однако в связи с распространением стандарта XML рекомендуется значения атрибутов заключать в двойные кавычки. Наличие атрибутов не является обязательным, и в случае отсутствия атрибута используется его значение по умолчанию.

Все HTML-дескрипторы можно разделить на несколько групп: служебные теги, влияющие на общие свойства отображения и интерпретации документа (например, приведенный выше тег <META>); теги, определяющие структуру документа (<HTML>, <HEAD>, <BODY> и т.п.); теги, влияющие на внешний вид содержимого (<FONT>, <B>, <I> и т.п.); теги, указывающие на необходимость вставки некоторых объектов (<IMG>, <OBJECT>, <SCRIPT> и др.).

Особого внимания заслуживает дескриптор <A>, применяемый для создания гиперссылок – того механизма, который и объединяет (с точки зрения пользователя) информационные ресурсы в единую глобальную компьютерную сеть. Основными его атрибутами являются атрибуты href и name, используемые для указания назначаемого гиперссылкой ресурса или назначения имени идентификатора фрагмента (анкера). Например, для того, чтобы создать анкер с именем info в некотором месте документа

doc.html, необходимо использовать тег <A> с атрибутом name: **<A name="info">Информация</A>**. Для создания гиперссылки на этот анкер необходим атрибут href, в котором имя анкера добавляется к имени файла через диез: **<A href="doc.html#info">Ссылка на информацию</A>**.

III. Задание и порядок выполнения работы.

Перед выполнением работы по материалам лекции, практического занятия и размещенной на учебном сервере справочной информации изучить основные дескрипторы описания структуры гипертекстовых документов, форматирования текста и создания гиперссылок (HTML, HEAD, BODY, TITLE, A, B, I, U, P, BR, NOBR, CENTER, H1,…,H6, PRE, FONT, META, CITE, CODE, COMMENT, KBD, Q, SAMP, STRONG).

1. Освоить процедуру входа в свой рабочий каталог на учебном сервере LASER_MATH (192.168.15.35) и размещения там материалов.
2. Создать в своём каталоге файл index.htm, в котором разместить информацию о себе (ФИО, группа) и создать список лабораторных работ (Л.р.№1, Л.р.№2, ...). Этот список будет пополняться в течение семестра.
3. Создать файл (главную страницу лабораторной работы №1), содержащий перечисление каких-либо объектов (например, список товаров, список туров и т.п.). В файле index.htm создать гиперссылку на эту страницу.
4. Создать набор файлов с описаниями этих объектов. В предыдущем файле расставить гиперссылки на эти описания.

При выполнении задания обязательно использовать спецсимволы (например,   ©). При создании файлов обращать внимание на кодировку и заголовки страниц. Файлы HTML должны создаваться с использованием стандартного текстового редактора Windows. Тестирование выполнять, обращаясь к ресурсу через адресную строку браузера или по ссылке со списка группы. То есть, начав с адреса http://192.168.15.35/, «найти себя», а затем просто обновлять страницу средствами браузера (меню View|Refresh или клавиша [F5]).

IV. Содержание отчета.

1. Цель работы.

2. Краткая теоретическая часть.
3. Перечень всех пунктов задания и директивы-ответы на них с комментариями по особенностям их выполнения.
4. Выводы.

Лабораторная работа №2

ЯЗЫК РАЗМЕТКИ HTML: НЕТЕКСТОВЫЕ ЭЛЕМЕНТЫ

I. Цель работы: изучение принципов использования нетекстовых элементов гипертекста: изображений, таблиц, списков, форм.

II. Теоретическое введение.

Изначально язык разметки гипертекста был предназначен лишь для описания структурных элементов документа: заголовков, разделов, списков, таблиц и т.п. Однако конкуренция на рынке браузеров, развернувшаяся в основном между двумя гигантами, Netscape и Microsoft, привела к отходу от такой доктрины и появлению большого количества дескрипторов и атрибутов, имеющих исключительно оформительское назначение: настройка цветовых и пространственных решений, вставка изображений, звукового и видео-сопровождения, создание интерактивных элементов, фреймов и многих других элементов веб-страниц. До введения стандарта HTML 4.0 процесс разработки веб-ресурсов осложнялся необходимостью знать многочисленные различия в наборах подобных тегов у различных фирм-производителей. После введения данного стандарта и появления всё новых версий браузеров различия эти свелись к минимуму и на сегодняшний день проявляются только на уровне отдельных атрибутов. Несмотря на это, необходимо помнить о возможных отличиях и проверять создаваемые документы, как минимум, в наиболее распространённых версиях браузеров.

Отдельно обращает на себя возможность создания элементов диалога веб-страницы с пользователем с помощью форм. До появления форм единственными элементами веб-страницы, реагирующими на действия пользо-

вателя, являлись гиперссылки. Механизм форм (дескриптор <FORM>) позволяет создавать интерфейс, во многом схожий с интерфейсом диалоговых окон обычных оконных приложений и содержащий такие объекты как поля ввода текста, кнопки, селекторные переключатели, списки и т.п. Более того, данные, введённые с помощью элементов формы, могут быть переданы в HTTP-запросе клиента и обработаны сервером (см. лабораторные работы №6 и №7). Для указания HTTP-метода передачи данных используется атрибут method дескриптора <FORM>, а для указания URL серверного приложения, получающего эти данные – дескриптор action. Например, отправка значений элементов формы серверному сценарию guest.asp методом POST будет выполнена в случае следующего тега: <FORM method="POST" action="guest.asp">.

III. Задание и порядок выполнения работы.

Перед выполнением работы по размещенной на учебном сервере справочной информации изучить дескрипторы описания изображений, таблиц, списков и форм, а также атрибуты цветовых настроек (<BODY background bgcolor>, <A href title>, <IMG src alt width height>, <TABLE align width bgcolor>, <TR bgcolor>, <TD width>, <OL>-<LI>, <UL>-<LI>, <FORM>, <INPUT type="text, checkbox, radio, submit, reset">, <SELECT>-<OPTION>, <TEXTAREA rows cols wrap>).

1. Взять за основу набор страниц, созданный в рамках лабораторной работы №1.
2. Задать страницам фоновый цвет и фоновое изображение.
3. Придумать и нарисовать логотип сайта (простейшее изображение формата GIF). Разместить его на всех страницах, сопровождая справа названиями страниц. На всех внутренних страницах логотип должен являться ссылкой на главную страницу сайта. Логотип должен сопровождаться альтернативным текстовым описанием.
4. На странице с перечислением объектов (главная страница лабораторной работы №1) представить краткую информацию о них в табличной форме, отформатировав таблицу по образцу форматирования списка группы (чередующиеся цветные полосы). Ширина таблицы должна составлять 75% от ширины окна, расположение – по центру.

5. Гиперссылки на описания объектов сопроводить всплывающими подсказками.
6. Продублировать перечень объектов на главной странице, оформив его в виде списка (маркированного или нумерованного) и связав гиперссылками получившиеся элементы списка с соответствующими страницами.
7. Создать страницу с опросной формой (поля ввода: ФИО, возраст, адрес, телефон, адрес электронной почты; флажки: области интересов; селекторные переключатели: основная сфера деятельности; выпадающий список: набор оценок сайта (1, 2, 3, 4, 5, "Затрудняюсь ответить"); многострочное поле ввода: комментарий; кнопки: "ОК", "Очистить форму"). Ссылку на эту страницу разместить на главной странице

При создании файлов обращать внимание на кодировку и заголовки страниц. Файлы HTML должны создаваться с использованием стандартного текстового редактора Windows. Тестирование выполнять, обращаясь к ресурсу через адресную строку браузера или по ссылке со списка группы. То есть, начав с адреса http://192.168.15.35/, «найти себя», а затем просто обновлять страницу средствами браузера (меню View|Refresh или клавиша [F5]).

IV. Содержание отчета.

1. Цель работы.
2. Краткая теоретическая часть.
3. Перечень всех пунктов задания и директивы-ответы на них с комментариями по особенностям их выполнения.
4. Выводы.

Лабораторная работа №3

КАСКАДНЫЕ ЛИСТЫ СТИЛЕЙ

I. Цель работы: освоение базовых принципов использования технологии каскадных листов стилей (CSS) и методологии разделения содержания и представления гипертекстовой информации.

II. Теоретическое введение.

С момента возникновения HTML большее предпочтение в нем отдавалось содержанию, нежели стилю. Основной задачей автора было предоставление информации высокого качества, а заботы о том, как эта информация выглядит, доставались браузеру. Такая методология остается актуальной (и даже наиболее предпочтительной) и на сегодняшний день. Несмотря на то что использование дескриптора <font> и связанных с ним атрибутов позволяет добиться ярких эффектов, разумное применение таблиц стилей вносит в документы порядок и единообразие. Таблицы стилей позволяют автору документа HTML управлять атрибутами представления во всех тегах документа или целой группы документов с помощью одной главной таблицы стилей.

На самом простом уровне стиль является всего лишь правилом, сообщающим браузеру, как отображать отдельные теги HTML. Правило определяет отдельное значение для одного или более свойств тега. Например, у большинства дескрипторов есть свойство color, значение которого определяет цвет, используемый для отображения содержимого дескриптора. В число других свойств входят атрибуты гарнитуры шрифта, межстрочный интервал, размеры полей и границ.

Правило стиля состоит, по крайней мере, из следующих основных частей: селектора тега, на который распространяется правило стиля, и последующих фигурных скобок, заключающих между собой разделенные точкой с запятой пары объявлений «свойство_стиля: значение», например:

```
H1 {text-align: center}
```

Помимо одиночных селекторов тегов возможно указание множественных и контекстных селекторов. В первом случае селекторы перечис-

ляются через запятую, а свойства стиля применяются к каждому из перечисленных дескрипторов в случае наличия такового в гипертекстовом документе. Во втором случае селекторы перечисляются через пробел, что указывает на отношение вложенности дескрипторов. Стиль применяется только в том случае, если выполнено контекстное условие вложенности для последнего из перечисленных селекторов. Например, строка «OL OL LI {list-style: upper-roman}» означает, что заглавными латинскими цифрами будут помечаться элементы списка второго уровня.

Согласно спецификации CSS, возможно описание классов и псевдоклассов стилей. Они позволяют описать различные свойства отображения для одного и того же дескриптора (или группы дескрипторов). Отличие в описании класса стиля заключается в наличии дополнительного идентификатора (имени класса), отделяемого от селектора тега точкой. Например, строка «P.abstract {font-style:italic; margin-left:0.5cm; margin-right:0.5cm}» описывает класс стиля абзаца с именем “abstract”. Для использования класса стиля при отображении дескриптора необходимо задействовать атрибут `class`, значением которого должно являться требуемое имя класса. Псевдоклассы аналогичны обычным классам, но соединяются с именем тега через двоеточие, а не точку, имеют предустановленные имена и не требуют наличия атрибута `class`.

Существует три способа присоединения стиля к тегу: внутренние стили (описание на уровне отдельного дескриптора), стили уровня документа (описание заключается в пару дескрипторов `<STYLE>…</STYLE>` в заголовочной части документа) и внешние таблицы стилей (описание находится в файле с расширением .css). В документах допускается использование одного или нескольких видов стилей. При этом браузер соединяет и определяет свойства стилей, начиная с внешних таблиц через локальные стили документа и заканчивая внутренними стилями. Такое каскадирование свойств и правил стилей и дало название стандарту.

III. Задание и порядок выполнения работы.

Перед выполнением работы изучить размещенный на учебном сервере теоретический материал, касающийся технологии CSS.

1. Используя механизм классов стилей и идентификаторы объектов, создать и применить набор стилей для описания таблицы, аналогичной

показанной на рис.4.1. Заголовочные строки и столбцы выделены цветом, начертанием и расположением текста, отделены двойными линиями. У внутренних ячеек свой способ оформления содержимого. Должна быть возможность «группировки» ячеек по примеру того, как отделены четвёртая и пятая строки, а также четвёртый и пятый столбцы.

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5

Рис.4.1.

2. С помощью псевдостилей (A:link, A:hover, A:active, A:visited) описать и применить свои способы отображения гиперссылок (изменить цвет и начертание шрифта для непосещённых, выделяемых, обрабатываемых и посещённых ссылок).
3. Описать и применить набор контекстных стилей для создания списка по образцу, показанному на рис.4.2.

A. HTML
B. CSS
C. CGI и Perl
I. Обзор CGI
II. Включения на стороне сервера
III. Модуль CGI.pm
a. Генерация тегов HTML
b. Использование JavaScript
c. Отладка
1. работа в автономном режиме
2. передача параметров в командную строку

Рис.4.2.

4. Описать класс абзаца с необязательной информацией: текст в нём отображается курсивом и меньшим кеглем по отношению к остальному тексту; слева и справа от абзаца – отступы.
5. Опробовать полученный стилевой файл на страницах, созданных в ходе выполнения лабораторных работ №1 и №2.

Все стилевые определения должны быть описаны во внешней таблице стилей. Использование стилей уровня документа или внутренних стилей допустимо только для модификации стилей более высокого уровня, если это необходимо для выполнения задания.

IV. Содержание отчета.

5. Цель работы.
6. Краткая теоретическая часть.
7. Перечень всех пунктов задания и директивы-ответы на них с комментариями по особенностям их выполнения.
8. Выводы.

Лабораторная работа №4

СТАНДАРТНЫЕ И НЕЗАВИСИМЫЕ ОБЪЕКТЫ JAVASCRIPT

I. Цель работы: изучение технологии написания клиентских приложений с использованием языка сценариев JavaScript.

II. Теоретическое введение.

JavaScript – это облегченный объектно-ориентированный язык сценариев. Ядро этого языка встроено и м Netscape Navigator, и в Microsoft Internet Explorer, а также в другие браузеры, и расширено для веб-программирования добавлением объектов, представляющих окно браузера и его содержимое. Версия JavaScript для клиентской стороны позволяет включать в веб-страницы выполняемое содержимое. С помощью JavaScript можно выйти за пределы статического HTML и создавать страницы, включающие в себя программы, которые взаимодействуют с пользователем, управляют браузером и динамически создают HTML-содержимое. JavaScript является наиболее распространенным языком сценариев, выполняющихся на стороне клиента.

Суть языка JavaScript можно выразить в следующих основных принципах:

JavaScript можно внедрить в HTML. Это осуществляется с помощью дескриптора <SCRIPT>…</SCRIPT> (сценарий описывается непосредственно в документе HTML или в подключаемом внешнем файле с расширением .js) или в виде кода, обрабатывающего события, связанные с элементами форм.

JavaScript зависит от среды. Для выполнения кода требуется поддержка браузером используемых в этом коде средств.

JavaScript – интерпретируемый язык. JavaScript не компилируется в двоичный код наподобие .exe, а, оставаясь частью документа HTML, интерпретируется браузером.

JavaScript – слаботипизированный язык. Нет необходимости объявлять переменные специального типа, а одна и та же переменная может хранить в разных точках программы данные разного типа (числовые, строковые и другие).

JavaScript – объектно-ориентированный язык. Работа ведется с объектами, которые инкапсулируют данные (свойства) и поведение (методы). Однако объектная модель JavaScript основывается на экземплярах, а не на концепции наследования. Различают стандартные объекты, независимые объекты и пользовательские объекты. Первые служат для работы с элементами гипертекста и браузера. Набор стандартных объектов и их свойств может несколько отличаться у различных браузеров. Независимые объекты стандартизованы Европейской ассоциацией производителей компьютеров (ECMA), образуют стандартизованную версию JavaScript – язык ECMAScript – и предназначены для работы с такими данными как строки, массивы, время, математические функции и т.п. Пользовательские объекты описываются разработчиками и служат для решения специфических задач.

JavaScript – язык, управляемый событиями. HTML-элементы, подобные кнопкам, спискам или текстовым полям, усовершенствованы с целью поддержки обработчиков событий. Большинство написанных на JavaScript кодов как раз и оказываются связанными с теми или иными событиями.

JavaScript – это не Java. Java и JavaScript разрабатывались разными компаниями. Различия между Java и JavaScript очень велики и проявляются на уровне базовых принципов этих языков. Можно даже сказать, что об-

щим у этих языков является только си-подобный синтаксис. Основная причина сходства имен кроется в маркетинговых соображениях.

JavaScript – многофункциональный развивающийся язык. Области применения JavaScript в клиентском программировании очень широки, а большое количество версий языка с различными функциональными возможностями позволяет очень гибко подходить к решению задач, стоящих перед разработчиком.

III. Задание и порядок выполнения работы.

Перед выполнением работы изучить размещенный на учебном сервере справочный материал, касающийся базового синтаксиса языка сценариев JavaScript, а также его системы стандартных и независимых объектов.

1. Для формы из лабораторной работы №2 выполнить проверку на наличие незаполненных полей. Проверка должна выполняться после нажатия кнопки <ОК>. При наличии незаполненных полей пользователь должен увидеть соответствующее предупреждение (например, «Не введена фамилия!»)
2. Сделать проверку правильности ввода телефонного номера. он должен быть введён в формате «(XXX) XXXXXXX», где 'X'-десятичная цифра. Общее количество цифр в скобках и за скобками должно равняться десяти.
3. Сделать проверку корректности ввода адреса электронной почты.

IV. Содержание отчета.

1. Цель работы.
2. Краткая теоретическая часть.
3. Перечень всех пунктов задания и директивы-ответы на них с комментариями по особенностям их выполнения.
4. Выводы.

Лабораторная работа №5

СОЗДАНИЕ ДИНАМИЧЕСКИХ СРЕДСТВ НАВИГАЦИИ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА СЦЕНАРИЕВ JAVASCRIPT

I. Цель работы: освоение принципов написания клиентских сценариев JavaScript, управляемых событиями.

II. Теоретическое введение.

Как отмечалось выше, язык сценариев JavaScript предназначен для написания приложений, работающих на стороне веб-клиента. Поэтому основой объектной системы являются объекты, представляющие свойства клиента и отображаемой его средствами гипертекстовой информации. Иерархия основных объектов показана на рис.5.1.

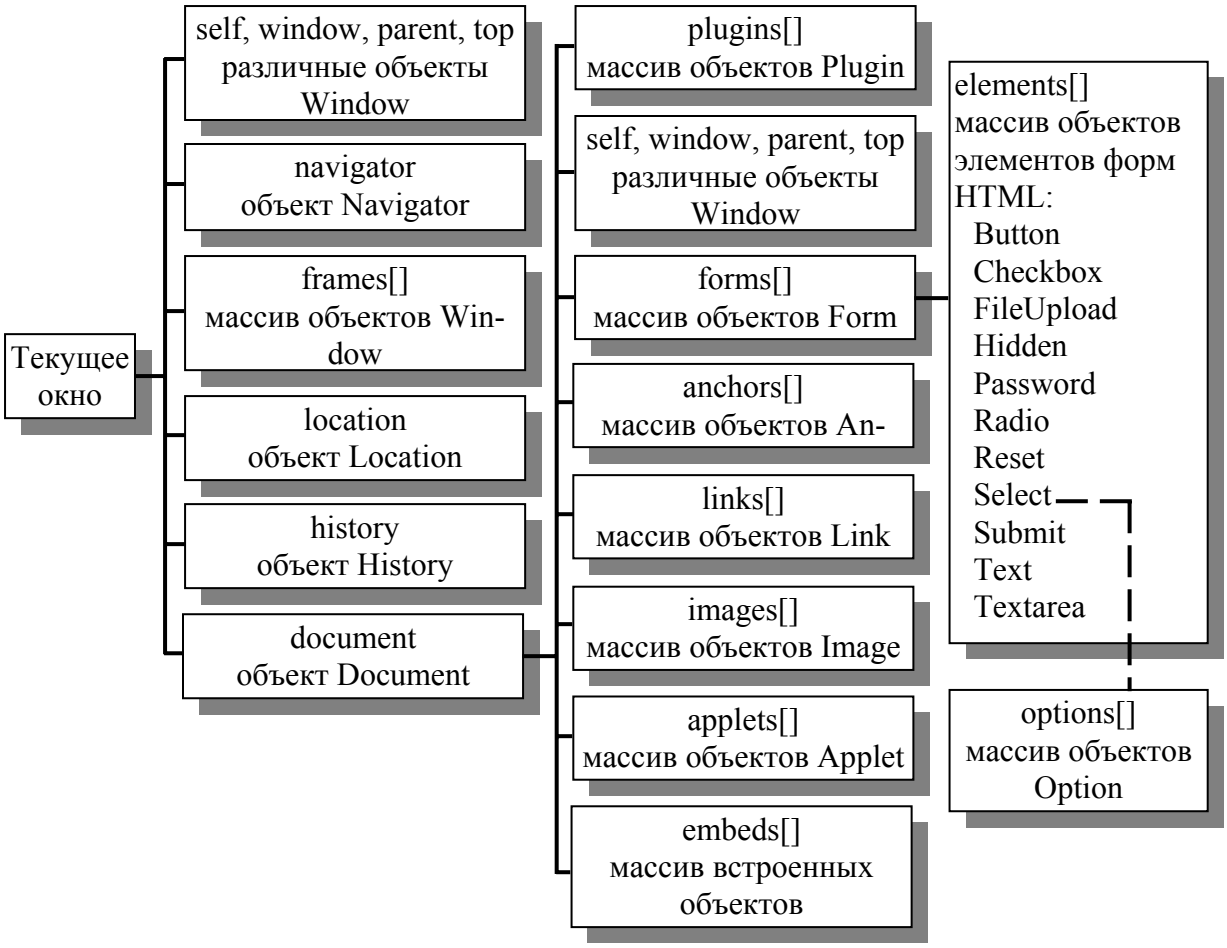

Рис. 5.1.

Необходимость в обработке событий возникает в основном при каких-либо действиях пользователя с элементами форм, то есть с объектами, хранящимися в массиве embeds[]. Но события могут быть связаны и с другими объектами. Например, при загрузке страницы возникает событие onLoad, при перемещении указателя мыши – события onMouseOver, onMouseOut и т.д. JavaScript на стороне клиента поддерживает несколько типов событий. В табл.5.1 перечислены обработчики событий и объекты на стороне клиента, поддерживающие эти обработчики. Генерация некоторых событий, например двойного щелчка onDblClick, не на всех платформах происходит корректно.

Таблица 5.1.

Обработчик событий	**Поддерживающие объекты**
onAbort	Image (JavaScript 1.1)
onBlur, onFocus	Текстовые элементы; Window и все остальные элементы формы (JavaScript 1.1)
onChange	Select, элементы ввода текста
onClick	Элементы-кнопки, Link; для отмены действия по умолчанию нужно возвратить false
onDblClick	Document, Link, Image, элементы-кнопки (JavaScript 1.2)
onError	Image, Window (JavaScript 1.1)
onKeyDown, onKeyPress, onKeyUp	Document, Image, Link, текстовые элементы (JavaScript 1.2); для отмены действия по умолчанию нужно возвратить false
onLoad, onUnload	Window; Image (JavaScript 1.1)
onMouseDown, onMouseUp	Document, Link, Image, элементы-кнопки (JavaScript 1.2); для отмены действия по умолчанию нужно возвратить false
onMouseOver, onMouseOut	Link; Image и Layer (JavaScript 1.2); возвратить true для предотвращения вывода URL
onReset, onSubmit	Form (JavaScript 1.1); для предотвращения сброса или передачи нужно возвратить false

Для связи собственной функции JavaScript с обрабатываемым событием необходимо указать ее имя в качестве значения атрибута (которым служит название события) того дескриптора, работа которого вызывает данное событие. Например обработку нажатия кнопки можно реализовать строкой `<INPUT type='button' onClick='MyFunction()'>`.

III. Задание и порядок выполнения работы.

Перед выполнением работы изучить размещенный на учебном сервере теоретический материал, касающийся обработки событий средствами языка сценариев JavaScript.

1. В файле index.htm (см. лабораторную работу №1) сделать ссылки на лабораторные работы в виде графических кнопок, изменяющих свой вид при наведении на них указателя мыши (графические изображения кнопок создать предварительно в любом графическом редакторе).
2. На каждой из страниц, полученных ранее в рамках лабораторных работ, создать навигационный элемент в виде выпадающего списка, содержащего названия этих страниц. При выборе какого-либо из названий должен происходить автоматический переход на соответствующую страницу. (Для текущей страницы переход выполняться не должен!)

IV. Содержание отчета.

1. Цель работы.
2. Краткая теоретическая часть.
3. Перечень всех пунктов задания и директивы-ответы на них с комментариями по особенностям их выполнения.
4. Выводы.

Лабораторная работа № 6

РАБОТА С ФАЙЛАМИ И БАЗАМИ ДАННЫХ С ПОМОЩЬЮ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ ASP

I. Цель работы: Ознакомление с основами технологии программирования Active Server Pages (ASP). Освоение методов написания серверных ASP-приложений для работы с файлами и базами данных.

II. Теоретическое введение.

ASP – это мощная технология от Microsoft, позволяющая легко разрабатывать веб-приложения (в основном, сценариев серверной части для информационных веб-серверов). Главной идеей и достоинством ASP является возможность встраивать программный код, обрабатываемый сервером, непосредственно в HTML страницу (а не наоборот, как предлагает стандарт CGI). ASP позволяет применять любой язык написания сценариев, удовлетворяющий стандарту ActiveX. По умолчанию в технологии ASP используется простой скриптовый язык VBScript, но распространены и другие языки, например, JScript или Perl.

Файлы с программами создаются с помощью любого текстового или специализированного редактора и имеют расширение .asp. Они должны быть расположены в каталоге с правами на выполнение. Когда браузер клиента запрашивает файл, он интерпретируется сервером, в результате чего генерируется HTML-код, который посылается клиенту.

Таким образом, ASP-документы – это смесь HTML-кода с дополнительным программным кодом, написанным, например, на языке VBScript, причем эти фрагменты заключаются в теги <% %>. Они указывают на сценарий, который должен исполняться сервером.

В VBScript есть все нормальные конструкции структурного программирования (if, while, case, etc). Есть слаботипизированные переменные и поддержка объектов с обычной точечной нотацией для обращения к их свойствам и методам. Есть ряд встроенных объектов-утилит (Request, Response, Session, Server, Application). Добавляются и другие компоненты, например, для работы с электронной почтой.

При необходимости использовать язык, отличный от VBScript, на это надо указать веб-серверу:

```
<%@ LANGUAGE="JSCRIPT" %>
```

Символ @ обозначает, что в этой строке объявлен язык программирования. Такое объявление следует добавлять один раз на каждой странице.

Следует отметить тег: <%= %>. Он дает команду веб-серверу отобразить на экране значение выражения, находящегося внутри этого тега. Например,

```
<P>Сегодня <%= Now %>.<P>
```

Now – это встроенная функция VBScript, которая возвращает текущее значение даты и времени. Для вывода информации на экран можно использовать и объекты, а именно метод Write объекта Response:

```
<P>Сегодня <% Response.Write Now %>.<P>
```

Результат будет тот же, что в предыдущем примере, т.е. пользователь увидит строку, содержащую текущие дату и время.

Для записи некоторого значения в переменную в VBScript используется оператор Set: **<% Set name="Петров" %>**

Значение в переменную может быть записано и с помощью объекта Request, содержащего данные из пар «имя=значение», переданных в HTTP-запросе пользователя (например, значения элементов форм – см. лабораторную работу №2):

```
test.asp?var=abc          'передача в программу test.asp переменной var
<% var = Request("var")%> 'получение значения переменной из программы
```

Основу работы с файлами средствами ASP составляет ключевой метод объекта Server: Server.CreateObject (ObjectID).

Метод CreateObject позволяет создавать экземпляры других объектов, предоставляя, таким образом, разработчикам доступ к их коллекциям, событиям, методам и свойствам.

Объектом, предоставляющим доступ к файловой системе на сервере, является объект FileSystemObject, позволяющий производить разнообразные операции над текстовыми файлами, папками, а также над логическими дисками посредством ASP-кода.

Для создания экземпляра объекта FileSystemObject необходимо выполнить следующую инструкцию:

```
Set f=Server.CreateObject("Scripting.FileSystemObject")
```

Из ASP можно легко и просто работать с любыми базами данных (БД). Это делается через две промежуточные технологии: ODBC и ADO.

ODBC позволяет организовать доступ к любым базам данных с помощью языка SQL через «источник данных», который настраивается на определенную БД при помощи специальных драйверов. Такие драйверы существуют для всевозможных СУБД (в частности MS SQL Server, Oracle, MS

Access, FoxPro). Источник данных – это совокупность сведений о базе данных, включая ее драйвер, имя компьютера и файла, параметры. Важно, чтобы источник данных был "системным", в отличие от "пользовательского".

ADO – это совокупность объектов, доступных из ASP и позволяющих обращаться к источнику данных ODBC. Фактически нужны лишь два объекта – Connection, представляющий соединение с базой данных и Recordset, представляющий набор записей, полученный от источника. Другими объектами являются Command**,** Parameter, Field, Property, Error.

III. Задание и порядок выполнения работы.

Перед выполнением работы изучить размещенный на учебном сервере теоретический материал, касающийся программирования с помощью технологии ASP.

1. Опросную форму из лабораторной работы №2 связать с серверным сценарием (например, guest.asp). Получив данные, серверный сценарий должен записать их в текстовый файл в виде:

```
<дата>   <время>
IP-address: <IP-адрес узла, приславшего сообщение>
Browser type: <информация о клиенте>
<ФИО>, <возраст>
<адрес>
<телефон>
<адрес электронной почты>
Оценка: <оценка из выпадающего списка>
Сообщение: <сообщение из многострочного поля ввода>
```

2. Создать второй вариант серверного сценария, который перечисленную выше информацию записывает не в текстовый файл, а в базу данных MS Access.

IV. Содержание отчета.

1. Цель работы.
2. Краткая теоретическая часть.
3. Перечень всех пунктов задания и директивы-ответы на них с комментариями по особенностям их выполнения.
4. Выводы.

Лабораторная работа № 7

РАБОТА С СЕССИЯМИ В ASP

I. Цель работы: ознакомление с принципами разработки ASP-приложений, поддерживающих сеансы пользователей.

II. Теоретическое введение.

Для разработки полноценных приложений технология ASP предоставляет возможность хранения информации о сеансе работы для конкретного пользователя. Для этого предназначен объект Session, экземпляр которого создается, когда пользователь обращается к любой из страниц приложения. Сессия действует или в течение заранее определенного периода времени, или пока сеанс не будет закончен явно. В объект Session можно записывать переменные, которые доступны из любой страницы в рамках сессии, и считывать их оттуда:

```
Session("var") = var
var = Session("var")
```

Таким образом, сессия – это еще один метод передачи данных между страницами.

Наряду с объектом Session существует объект Application, который служит для предоставления информации всем пользователям, т.е. он может применяться для совместного использования данных.

Технология ASP дает возможность программистам управлять приложением. Это делается с помощью файла GLOBAL.ASA, в котором объявляются глобальные переменные и определяются обработчики событий, доступные для всех страниц приложения. Этот файл может содержать четыре подпрограммы:

Application_OnStart выполняется при запуске сервера;

Application_OnEnd выполняется, когда сервер завершает работу;

Session_OnStart выполняется при инициализации сессии пользователя;

Session_OnEnd выполняется, когда пользователь завершает свою сессию или по истечении заданного времени.

Примерная структура файла GLOBAL.ASA:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
```

```
SUB Application_OnStart
END SUB
SUB Application_OnEnd
END SUB
SUB Session_OnStart
END SUB
SUB Session_OnEnd
END SUB

</SCRIPT>
```

Допустимо использовать множественные файлы GLOBAL.ASA. Однако следует помнить, что ASP-сценарий ищет самый близкий файл GLOBAL.ASA и использует именно его. GLOBAL.ASA не может содержать тэгов HTML. Недопустимо использование JavaScript. Не рекомендуется писать файл GLOBAL.ASA с помощью каких-либо HTML-редакторов, для этого гораздо лучше использовать простейший текстовый редактор Notepad. Прежде чем вставлять фрагмент программы в файл GLOBAL.ASA, желательно протестировать его в обычном ASP-файле.

Часто при формировании страниц необходимо вставить код какой-либо JavaScript-функции, например, в форме для проверки вводимых данных. В таком случае имеет смысл оформить JavaScript-функции в виде процедур. Это повышает читаемость кода программы и позволяет в случае необходимости изменить алгоритм функционирования вставляемого кода на JavaScript. Например, есть функция подтверждения выполнения действия

```
Sub YesNo_Click() %>
function YesNo_Click()
{ return (confirm("Вы уверены?")); }
<%
End Sub
```

тогда для добавления функции в ASP-модуле достаточно будет записать:

```
<SCRIPT LANGUAGE=javascript>
<%YesNo_Click()%>
</SCRIPT>
```

III. Задание и порядок выполнения работы.

1. Создать базу данных Access для хранения регистрационной информации о посетителях (автоматически генерируемый идентификатор пользователя, ФИО, имя пользователя, пароль).
2. Создать страницу с регистрационной формой (ФИО, имя пользователя, пароль, подтверждение пароля). После заполнения регистрационной формы и проверки средствами JavaScript совпадения строк в полях ввода и подтверждения пароля информация пересылается в базу данных, т.е. пользователь регистрируется.
3. Создать страницу, являющуюся центральной страницей лабораторной работы №7. На ней должна присутствовать форма ввода имени/пароля для зарегистрированных пользователей и ссылка на страницу регистрации (см. предыдущий пункт). В случае корректного ввода имени/пароля должен происходить переход на страницу ввода информации в гостевую книгу, созданную в лабораторной работе №6. Если процедура авторизации не выполняется корректно (имя пользователя или пароль введены неправильно), должно выдаваться соответствующее предупреждение, а гостевая книга должна быть недоступна.
4. При успешном прохождении авторизации, в переменную сеанса ID_user должен занестись идентификатор пользователя. При добавлении информации в гостевую книгу этот идентификатор должен автоматически в гостевой книге фиксироваться (в зависимости от варианта реализации, записываться в текстовый файл или заноситься в соответствующее поле таблицы БД).

IV. Содержание отчета.

1. Цель работы.
2. Краткая теоретическая часть.
3. Перечень всех пунктов задания и директивы-ответы на них с комментариями по особенностям их выполнения.
4. Выводы.

Рекомендательный библиографический список

1. *Спейнауэр С., Экштейн Р.* Справочник вебмастера. (2-е изд.) / Пер. с англ. Маккавеева С. СПб: Символ-Плюс, 2001. – 608 с., ил. ISBN 5-93286-014-6
2. *Холмогоров В.* Основы Web-мастерства. Учебный курс. – СПб.: Питер, 2001. – 352с., ил. ISBN 5-272-00338-1.
3. *Вайк А., Вагнер Р.* JavaScript в примерах. Киев: Издательство «Диа-Софт», 2000. – 304 с., ил. ISBN 966-7393-57-7
4. *Фергюсон Д.* Отладка в ASP. Руководство для разработчиков. / Пер.с англ. Задорожного С.С., Левчука Ю.А. М.: ЗАО «Издательство БИНОМ», 2001. – 400 с.: ил. ISBN 5-7989-0219-6
5. *Николас. Ч.* Active Server Pages 3.0 на примерах. / Пер с англ., под ред. Александрова В.В. – М.: Издательский дом «Вильямс», 2001. – 280 с., ил. ISBN 5-8459-0142-1
6. *Уильямс Э., Барбер К., Ньюкирк П.* Active Server Pages. / Пер с англ. Иноземцева С., под ред. Кондуковой Е. – СПб.: БХВ-Петербург, 2001.– 672 с., ил. ISBN 5-94157-005-8

ОГЛАВЛЕНИЕ

ОСНОВЫ ВЕБ-ПРОГРАММИРОВАНИЯ

Методические указания к лабораторным работам
по дисциплине «Программирование для Интернет и веб-дизайн»

Составители:

ЛЕКСИН Андрей Юрьевич

МИТРОФАНОВ Даниил Владимирович

Ответственный за выпуск – зав. кафедрой профессор С.М. Аракелян

Редактор Л.В. Пукова
Компьютерная верстка С.В. Павлухиной

ЛР №020275. Подписано в печать 14.02.05.
Формат 60x84/16. Бумага для множит. техники. Гарнитура Таймс.
Печать на ризографе. Усл. печ..л. 1.75. Уч.-изд. л. 1.63. Тираж 100 экз.
Заказ 24-2005г.
Редакционно-издательский комплекс
Владимирского государственного университета.
600000, Владимир, ул. Горького, 87.