МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Московский технологический университет"
МИРЭА

Институт информационных технологий

Подлежит возврату
№

ИНТЕРНЕТ-ТЕХНОЛОГИИ И СИСТЕМЫ

Методические указания
по выполнению лабораторных и практических работ
Часть 2

Для студентов
Направление подготовки
09.03.04 **Программная инженерия**

Профиль подготовки
Интеллектуальные программные системы и комплексы
(Работа № 2)

МОСКВА 2017

Составитель И.И. Холкин
Редактор В.М. Панченко

Методические указания к лабораторным и практическим работам по дисциплине «Интернет-технологии и системы», Часть 2 предназначены для студентов 4-го курса очной формы обучения (квалификация Бакалавр. Направление подготовки 09.03.04 Программная инженерия. Профиль подготовки Интеллектуальные программные системы и комплексы).

Предполагается, что студенты изучили языки HTML, JavaScript, визуальный WYSIWYG редактор для разработки и управления Web-сайтом и умеют применять основные приемы Интернет - программирования и Web- дизайна на практике, выполнив 8 лабораторных и практических работ [1] и лабораторную и практическую работы (№1 [2]. Данные лабораторные и практические работы (№ 2-4) закрепляют и расширяют полученные знания в области Интернет- технологий. Печатается по решению редакционно-издательского совета университета.

Рецензенты: Б.Б. Чумак,
А.Н. Райков

Лабораторная работа № 2
Практическое занятие № 2
Dynamic HTML. Создание визуальных эффектов. Динамические блоки. Визуальные фильтры. Слои и их использование.

Цель работы

Ознакомление с принципами, лежащими в основе технологии **Dynamic HTML**. Формирование навыков создания визуальных эффектов с помощью динамических блоков, визуальных фильтров и слоев.

Методические указания

1. Dynamic HTML

Dynamic HTML, или динамический HTML, не является отдельным языком программирования или разметки документа. Это всего лишь технология, реализующая электронные документы с динамически изменяющимся содержанием [3].
Реализация динамического HTML строится на трех компонентах:

- HTML— HyperText Markup Language, простой язык разметки гипертекстовых документов;
- CSS — Cascading Style Sheets, каскадные таблицы стилей HTML-документа;
- JavaScript — клиентский язык программирования (выполняется не на сервере, а непосредственно в браузере пользователя на его локальном компьютере). Возможные аналоги: VBScript, JScript.

Эти три компонента образуют важнейшую структуру под названием Document Object Model, DOM (Объектная модель документа), которая дополняет простоту HTML и изящество CSS возможностью динамического изменения содержания без перезагрузки электронного документа.
К сожалению, объектная модель DHTML (Dynamic HTML), входящая в состав Internet Explorer, начиная с версии 4.01, может не поддерживаться браузерами других производителей), что накладывает определенные ограничения на процесс разработки интерактивных документов с динамическим содержанием.

1.1. Создание визуальных эффектов

Динамический HTML дает возможность разработчику создать в пределах электронного документа рабочий инструментарий, позволяющий пользователю манипулировать содержанием страницы, видом и расположением объектов, элементов и т. д.
Рассмотрим действие Dynamic HTML на примере динамических информационных блоков и применения визуальных фильтров для графических изображений.

1.1.1. Динамические блоки

Область применения **динамических информационных блоков** чрезвычайно широка, поэтому остановимся лишь на одном, достаточно распространенном случае [3].

Предположим, на сайте расположена карта России с нанесенными на ней названиями городов. Посетитель должен быстро и в удобном виде получить информацию по каждому из городов (рис. 1.).

Какие могут быть способы реализации данной задачи? Разрезать карту на части или задать активные области для карты-изображения (Imagemap)? Неудобно, т. к. каждый раз посетитель будет переходить по ссылке и назад, что снова потребует загрузки графического файла.

Поместить нужный текст в поле **ALT**, чтобы тот отображался в виде всплывающей подсказки при наведении курсора? Бессмысленно, т. к. подсказка через несколько секунд исчезнет, а текста может быть много.

Самый подходящий и эффективный способ — прибегнуть к помощи **Dynamic HTML** в совокупности с Imagemap (листинг 1.). В этом случае при наведении курсора мыши на заданную активную область карты в определенном месте страницы появляется информация о городе (как текстовая, так и графическая). Такой подход имеет два важных достоинства: корректная работа и в Internet Explorer, и в других браузерах, а также экономия времени посетителя (графика загружается только один раз, а все действия выполняются непосредственно из кода текущего электронного документа).

Прежде всего необходимо определить JavaScript-сценарий в разделе **HEAD**.

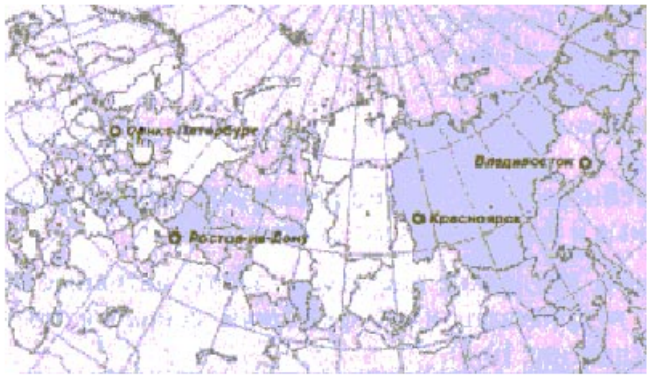

Рис. 1. Применение динамических информационных блоков
(в примере использованы некоторые стили CSS)

Листинг 1. JavaScript-сценарий для динамических информационных блоков |

```
<SCRIPT LANGUAGE="JavaScript1.2">

<!--
var ie=document.all ? 1 : 0;
var ns = document.layers ? 1 : 0;
var topcss = 165;
if (ns)
topcss = 200;
function showLayer(name)
```

```
{
if (ie)
document.all[name].style.visibility = "visible";
else if (ns)
document.layers[name].visibility = "show";
}
function hideLayer(name)
{
if (ie)
document.all[name].style.visibility = "hidden";
else if (ns)
document.layers[name].visibility = "hide";
</SCRIPT>
```

Далее определяем координаты информационных слоев (листинг **2**), которые невидимы до тех пор, пока курсор мыши не переместится над активной областью (размещаются в начале раздела body).

Листинг 2. Координаты информационных слоев

```
<STYLE TYPE="text/css">

<!--
#llink1
      { position: absolute; left: 160px; top: 250px; visibility: hidden; }
#link2
      { position: absolute; left: 160px; top: 250px; visibility: hidden; }
 #link3
      { position: absolute; left: 160px; top: 250px; visibility: hidden; }
#link4
      { position: absolute; left: 160px; top: 250px; visibility: hidden; }
//-->
</STYLE>
```

В данном случае в качестве селектора стилевого шаблона используется идентификатор. Поле

```
#link3
      {position: absolute; left: 160px; top: 250px; visibility: hidden; }
```

означает, что при активации области link3 скрытый до этого момента информационный слой, соответствующий ей, появится на странице на расстоянии 160 пикселов от левой границы окна и 250 пикселов— от его верхнего края. Здесь описание шаблонов заключается между символами комментариев. При отсутствии поддержки CSS браузер пропустит содержание стилевых шаблонов; если поддержка есть — браузер интерпретирует правила CSS. Position: absolute означает, что графическое изображение абсолютно спозиционировано и размещается в 250 пикселах от верхнего и в 160 пикселах от левого края своего родительского элемента (в качестве родительского элемента здесь выступает верхняя левая точка загруженной страницы).

Далее переходим к наполнению самих информационных слоев (листинг 3), которые в плане HTML лучше всего реализовать с помощью структурного тега <DIV>. Теги текстовых блоков <DIV> </DIV> используются для придания специальных свойств от-

дельному фрагменту текста. Изменение свойств осуществляется посредством назначения выбранному фрагменту текста стиля CSS

Листинг 3. Наполнение информационных слоев

```
<DIV ID='link1'>Санкт-Петербург</DIV>
<DIV ID='link2' >Ростов-на-Дону</DIV >
<DIV ID='link3'>Красноярск</DIV>
<DIV ID='link4'>Владивосток</DIV>
```

Внутри тега-контейнера <DIV> можно разместить таблицы, графические объекты, нумерованные и маркированные списки и пр.

Наконец, последнее, что нужно сделать, — задать активные области при помощи Imagemap и привязать их к функциям сценария по активации/дезактивации слоев (листинг 4).

Листинг 4**.** Определение активных областей Imagemap и привязка к JavaScript-сценарию

```
<IMG SRC="russiamap.jpg" WIDTH="400" HEIGHT="230" BORDER="0"
USEMAP="#russia">

<MAP NAME="russia">
<AREA SHAPE="rect" COORDS="63,70,162,85" HREF="link1.html"
onMouseOver="showLayer('link1');"onMouseOut="hideLayer('link1');">
<AREA SHAPE="rect" COORDS="98,137,193,157" HREF="link2.html"
onMouseOver="showLayer('link2');" onMouseOut="hideLayer('link2');">
<AREA SHAPE="rect" COORDS="249,125,326,141" HREF="link3.html"
 onMouseOver="showLayer('link3');" onMouseOut="hideLayer('link3');">
 <AREA SHAPE="rect" COORDS="289,89,368,105" HREF="link4.html"
onMouseOver="showLayer('link4');" onMouseOut="hideLayer('link4');">
</MAP>
```

Конструкции onMouseOver и onMouseOut являются событиями языка JavaScript (наведение и снятие курсора мыши с активной области на карте). Каждой активной области с заданными координатами должен соответствовать информационный блок со своим идентификатором (link1, link2, link3 и т. д.).

В результате при перемещении курсора, например над надписью "Санкт-Петербург" на карте, аналогичная надпись, но уже в текстовом виде, появляется прямо под изображением карты России (рис. 1, листинг 4**.**). Как уже было сказано, наполнение слоев можно осуществлять в соответствии с индивидуальными предпочтениями и оформительскими требованиями: вместо текстовых блоков вставлять графику, таблицы, гиперссылки и др.

Листинг 4. Применение динамических информационных блоков (итоговый листинг)

```
<HTML>
<HEAD>
<TITLE>Применение динамических информационных блоков</TITLE >

<SCRIPT LANGUAGE="JavaScript1.2">
<!--
var ie = document.all ? 1 : 0;
var ns = document.layers ? 1 : 0;
function showLayer(name)
```

```
{
if (ie)
document.all [name].style.visibility = "visible";
 else if(ns)
document.layers [name].visibility = "show";
}
function hideLayer(name)
{
if(ie)
document.all [name].style.visibility = "hidden";
 else if (ns)
document.layers [name].visibility = "hide";
}
//-->
</SCRIPT>
</HEAD>

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

<STYLE TYPE="text/css">
<!--
#llink1
       { position: absolute; left: 160px; top: 250px; visibility: hidden; }
#link2
       { position: absolute; left: 160px; top: 250px; visibility: hidden; }
 #link3
       { position: absolute; left: 160px; top: 250px; visibility: hidden; }
#link4
       { position: absolute; left: 160px; top: 250px; visibility: hidden; }
//-->
</STYLE>

<DIV ID='link1'>Санкт-Петербург</DIV>
<DIV ID='link2' >Ростов-на-Дону</DIV >
<DIV ID='link3'>Красноярск</DIV>
<DIV ID='link4'>Владивосток</DIV>

<IMG SRC="russiamap.jpg" WIDTH="400" HEIGHT="230" BORDER="0"
USEMAP="#russia">

<MAP NAME="russia">
<AREA SHAPE="rect" COORDS="63,70,162,85" HREF="link1.html"
onMouseOver="showLayer('link1');"onMouseOut="hideLayer('link1');">
<AREA SHAPE="rect" COORDS="98,137,193,157" HREF="link2.html"
onMouseOver="showLayer('link2');" onMouseOut="hideLayer('link2');">
<AREA SHAPE="rect" COORDS="249,125,326,141" HREF="link3.html"
 onMouseOver="showLayer('link3');" onMouseOut="hideLayer('link3');">
 <AREA SHAPE="rect" COORDS="289,89,368,105" HREF="link4.html"
onMouseOver="showLayer('link4');" onMouseOut="hideLayer('link4');">
</MAP>
</BODY>
```

</HTML>

1.1.2. Визуальные фильтры

В **Dynamic HTML** под фильтром принято понимать некую функцию, так или иначе преобразующую визуальное представление элемента на Web-странице. Преобразование осуществляется непосредственно в браузере, т. е. на стороне клиента (подключение сервера не требуется).

Следует отметить, что применение визуальных фильтров возможно только в браузерах Internet Explorer 4.x и выше (последняя версия Netscape, по утверждению разработчиков, способна поддерживать только лишь часть функциональных возможностей DHTML). Тем не менее, для горячих поклонников браузера от Microsoft **Визуальные фильтры** будут как нельзя более кстати и смогут существенно улучшить визуальное представление данных, размещаемых в электронных документах.

Прежде всего, визуальные динамические фильтры можно применить не ко всем элементам HTML-документа, а только к тем, которые способны определять блок прямоугольного вида при интерпретации браузером и при этом сами не являются окнами (к примеру, "плавающие" фреймы).

Формат записи фильтра достаточно прост и аналогичен правилам задания свойств элементов с помощью тега <style>. Запись производится в следующем виде:

filter: название_фильтра(параметры)

где параметры определяются в стандартном для HTML виде:

название_параметра=значение_параметра

Прежде чем перейти к рассмотрению фильтров и возможностей их применения, необходимо сказать, что, во-первых, допускается использование сразу нескольких фильтров (если это не противоречит окончательному визуальному результату отображения элемента под воздействием наложенных фильтров), во-вторых, при указании фильтров, не имеющих никаких параметров, присутствие круглых скобок (без пробелов) после названия фильтра обязательно.

1.1.2.1. Общие свойства и описание некоторых фильтров

При использовании фильтров следует помнить, что некоторые из них имеют общие свойства, влияющие на характер действия заданных параметров фильтра. К таковым относятся **enabled** (со значениями **true** и **false**, соответственно, разрешающим или запрещающим применение присоединенного к элементу документа фильтра), **direction** (определяет направление действия таких фильтров, как **shadow, blur** и др.), **strength** (задает интенсивность действия фильтра со значением от 0 до 255) и т. п. Часть фильтров, помимо общих свойств, имеет различные методы их определения (например, фильтр **light**).

Браузер Internet Explorer 4.x (и выше) поддерживает достаточно большое количество фильтров (табл. 1).

Таблица 1**.** Характеристика визуальных фильтров, работающих в браузере Internet Explorer

Название фильтра	Описание действия
alpha	Определение степени прозрачности объекта

blendTrans	Настройка контрастности отображения объекта
blur	Размытие объекта
chroma	Установление прозрачности пикселам заданного цвета
dropShadow	Создание сплошного силуэта объекта
glow	Создание эффекта свечения внешних границ объекта
light	Создание эффекта освещения объекта
revealTrans	Эффект появления/исчезновения объекта
wave	Эффект искривления объекта по вертикали
xray	Изменение глубины цвета объекта (эффект рентгеновского снимка)

Рассмотрим на примерах возможные варианты действия некоторых фильтров.

Фильтр *WAVE*

Фильтр **wave** создает синусоидальное искривление объекта в вертикальном направлении и имеет свойства add, enabled, freq, lightStrength, phase и strength.

Свойства enabled и strength были описаны выше, а остальные имеют следующее назначение:

- add— определяет необходимость добавления исходного вида объекта в его отфильтрованную интерпретацию;
- freq — задает количество максимумов в волне искривления объекта;
- lightStrength— добавляет эффект трехмерности гребням волны искажения объекта;
- phase — определяет фазу смещения волны (ее значение задается в процентах относительно начальной фазы, равной 0).

На рис. 2 показано три вида объекта — слева направо:

- исходный объект без действия фильтра wave
- объект с действием фильтра wave и заданными свойствами strength=5, add=0, lightStrength=20
- объект с заданными свойствами strength=5, add=0, phase=50, light-Strength=20, freq=30

Рис. . 2**.** Использование визуального фильтра wave

Листинг 5 Использование визуального фильтра wave

```
<! — Рисунок 1 —>
<IMG SRC="flower.jpg" WIDTH="243" HEIGHT="262">
<!— Рисунок 2 —>
<IMG SRC="flower.jpg" WIDTH="243" HEIGHT="262" STYLE="filter:
wave(strength=5, add=0, lightstrength=20)">
<!—  Рисунок 3  —>
<IMG SRC="flower.jpg" WIDTH="243" HEIGHT="262"  STYLE="filter:
wave(strength=5, add=0, phase=50, lightstrength=20, freq=30)">
```

1.2. Слои

Слои обозначаются английским словом **layers**. Слои появились с возникновением версии 4.0 браузера **Netscape Navigator** [4]. При помощи слоев стало возможным позиционирование элементов Web-страницы при помощи задания абсолютных координат слоя или координат относительных, задающих положение слоев друг относительно друга.

Использование координат слоев предоставляет возможность перемещения объектов, расположенных на данном слое, по HTML-странице. Можно установить частичную видимость тех или иных объектов на слое, использовать возможности языка **JavaScrip**t для манипулирования объектами слоев.

Чтобы понять, что представляют собой слои, приведем пример. Возьмем несколько листов чистой бумаги. На одном листе мы напишем текст, на другом мы нарисуем картинку, на третьем мы снова напишем какой-нибудь текст, и так далее.

Расположим наши листы бумаги на столе. Каждый лист бумаги — это слой. Это своеобразный носитель той информации, которую мы в этот слой вложили — на листах мы написали текст, нарисовали рисунок, эта информация содержится на наших листах, на слоях. Слои содержат объекты.

Возьмем лист с рисунком и будем передвигать его по столу. Рисунок передвигается вместе с листом, повторяя все его движения. Точно так же происходит и со слоем на Web-страничке. Слой может содержать несколько объектов, например, рисунков, форм, текстов и так далее, все эти объекты могут быть расположены на HTML-страничке и могут передвигаться по ней вместе со слоем.

Слой — это носитель этих объектов. Если слой передвинуть, то все объекты данного слоя переместятся вместе со слоем. Слои могут накладываться друг на друга точно так же, как можно наложить друг на друга листы бумаги на столе. Каждый слой может иметь прозрачные части.

Если вырезать на листе бумаги круг, то через получившееся отверстие можно видеть часть другого листа, расположенного под тем, на котором вырезан круг. Круглая дырка

— это прозрачная часть нашего слоя. Через прозрачные части слоя видно содержание того слоя, который расположен непосредственно за данным слоем

Как создать слои?

Для того, чтобы создать слой, используется тег <layer> или тег <ilayer>. Слои могут обладать следующими свойствами:

- name="layerName" .... Название слоя;
- left=xPosition..... Горизонтальная (x) координата верхнего левого угла слоя;
- top=yPosition..... Вертикальная (y) координата верхнего левого угла слоя;
- z-index=layerIndex..... Номер слоя, его "индекс глубины";
- width=layer Width ..... Ширина слоя в пикселях;
- clip="x1, y1, x2, y2" ..... Данные координаты определяют расположение видимой области слоя;
- above="layerName"..... Это свойство определяет имя слоя, над которым будет расположен данный слой;
- below="layerName" ..... Это свойство определяет имя слоя, под которым будет располагаться данный слой;
- Visibility=show|hide|inherit(наследовать) Свойство задает параметр видимости слоя;
- bgcolor=”rgbColor"..... Свойство задает цвет фона слоя, это либо название стандартного цвета, либо значение rgb цвета;
- background="imageURL" Свойство задает рисунок фона слоя;

Тег <layer> используется для создания слоев, положение которых задается в явном виде посредством свойств left и top. Если положение слоя на страничке не указано, то слой будет расположен в левом верхнем углу окна странички.

Тег <ilayer> используется при создании слоя, который располагается в соответствии с условиями формирования HTML-документа по ходу его создания.

Начнем наше знакомство со слоями с простого примера. Создадим два слоя, на первом слое поместим рисунок, а на втором слое — текст. Мы хотим показать текст поверх рисунка

Исходный текст программы представлен на **Листинг 6**.:

Листинг 6.

```
<html>
<layer name=pic z-index=0 left=130 top=50> <img src="img.gif" width=160 height=120> </layer>
<layer name=txt z-index=l left=60 top=50> <font size=+2> <i> Пример со слоями </i> </font>
</layer>
</html>
```

С помощью тегов <layer> мы определили два слоя. Координаты слоя мы задали с помощью свойств left и top, которые отсчитываются от верхнего левого угла страницы. Все, что располагается между тегами <layer> и </layer> (или между тегами <ilayer> и </ilayer>), принадлежит этому слою.

Мы также использовали свойство z-index. При помощи этого свойства мы установили порядок следования слоев или порядок их наложения друг на друга. Слой с самым

большим значением z-index будет находиться на самом верху. Для значений z-index не обязательно использовать последовательные числа.
Достаточно того, чтобы эти числа были целыми, порядок наложения слоев друг на друга определяется этими числами, чем больше значение z-index, тем "выше" расположен слой. Если, например, мы написали бы в ярлыке <layer> в первом слое выражение z-index=30, то этот слой был бы показан поверх другого, т.е. текст был бы расположен за рисунком, поскольку слой, содержащий текст, имел бы тогда меньшее значение z-index.
Для просмотра **Листинга 6** требуется браузер **Netscape Navigator** версии 4.0 и выше. Однако, если мы будем просматривать **Листинг 6**. с помощью **MIE 6.0**, то результат будет не корректным (рис. 3). Слои не располагаются друг над другом. Их положение не зависит от задаваемых координат и от значения z-indexa.

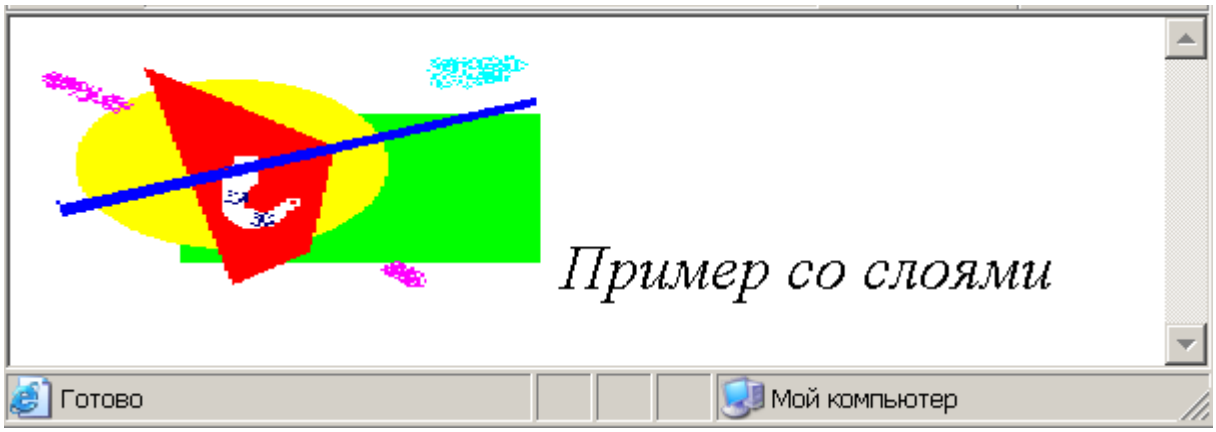

Рис. 3. Результат просмотра **Листинг 6**. с помощью **MIE 6.0**

Перепишем **Листинг 6** , используя стилевое оформление и текстовые блоки.

Листинг 7

```
<html>
<BODY>
<STYLE TYPE="text/css">
<!--
#sloj1
{ position: absolute; left:130px; top: 50px; visibility: visible ;z-index= 0; }
#sloj2
{ position: absolute; left:60px; top: 50px; visibility: visible ;z-index=1; }
//-->
</STYLE>
<DIV ID='sloj1'><img src="img.gif"  width=260  height=120></DIV>
<DIV ID='sloj2' ><font  size=+2>  <i> Пример со слоями </i></DIV>
</BODY>
</html>
```

В данном случае в качестве селекторов стилевого шаблона используются уникальные идентификаторы sloj1, sloj2.

Поле

#sloj1

{ position: absolute; left:130px; top: 50px; visibility: visible;z-index=0; }

означает, что информационный слой области sloj1 появится на странице на расстоянии 130 пикселов от левой границы окна и 50 пикселов— от его верхнего края.
Здесь описание шаблонов заключается между символами комментариев <!-- //-->. При отсутствии поддержки CSS браузер пропустит содержание стилевых шаблонов; если поддержка есть — браузер интерпретирует правила CSS. Position: absolute означает, что графическое изображение абсолютно спозиционировано и размещается в 130 пикселах от верхнего и в 50 пикселах от левого края своего родительского элемента (в качестве родительского элемента здесь выступает верхняя левая точка загруженной страницы).
Наполнение самих информационных слоев, выполнено с помощью структурного тега <DIV>. Теги текстовых блоков <DIV> </DIV> используются для придания специальных свойств отдельному фрагменту текста. Изменение свойств осуществляется посредством назначения выбранному фрагменту текста стиля **CSS**
Результаты просмотра **Листинга 7** приведены на рис. 4. На этом рисунке мы видим, что текст располагается поверх графического изображения.
На рис. 5 текст расположен позади картинки, поскольку в слое #sloj1 установлено z-index= 2.

Рис. 4. Результат просмотра **Листинг 7**. с помощью MIE 6.0

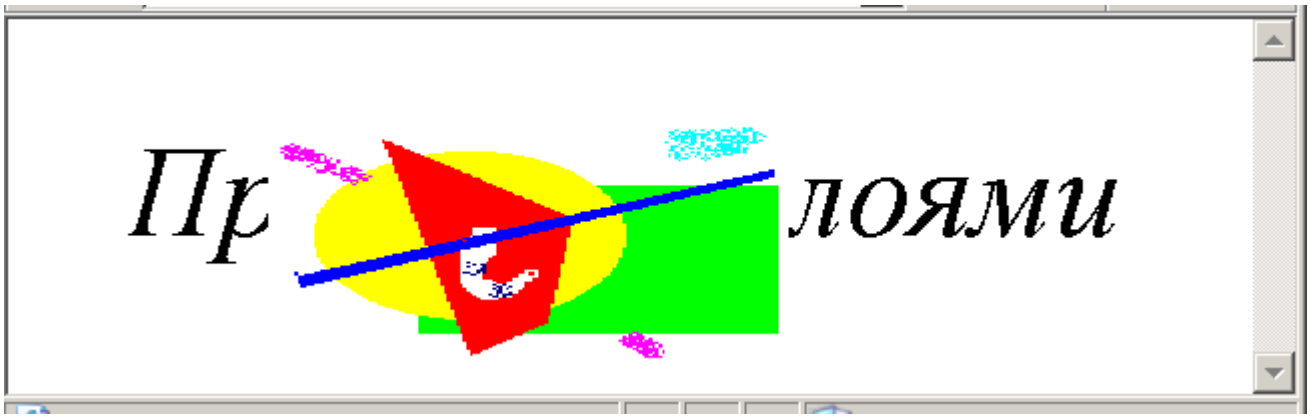

Рис. 5. Результат просмотра **Листинга 7** при z-index= 2 в слое #sloj1.

1.2.1. Слои и JavaScript

Сейчас мы узнаем, как можно управлять слоями при помощи **JavaScript**. И опять мы начнем с рассмотрения примера, в котором пользователь имеет возможность скрыть или показать слой путем нажатия кнопки. Вспомним, что объекты в JavaScript могут быть представлены несколькими способами, и слои здесь не являются исключением.

Если мы будем просматривать **HTML** – страницы со скриптами с помощью **MIE**, то слои следует представлять блоками , помечать их уникальными идентификаторами, а свойства, присущие слоям, задавать в стилевом блоке в соответствии с каждым идентификатором.

Объект layer обладает многими свойствами, которые могут быть изменены средствами языка **JavaScript**.

Рассмотрим пример, когда при помощи нажатия кнопки мы имеем возможность скрыть или показать один единственный слой **(Листинг 8)**. Исходный текст программы таков:

Листинг 8

```
<html>
<head>
<script language="JavaScript">
<!-- hide

function showHide(name)
{
if (document.all[name].style.visibility == "hidden" )
document.all[name].style.visibility = "visible";
 else
document.all[name].style.visibility = "hidden";
}

// -->
</script>
</head>
<body>

<STYLE TYPE="text/css">
<!--
#sloj
{visibility: visible ; }
//-->

</STYLE>

<DIV ID='sloj'>Этот текст расположен внутри слоя </DIV>

<form>
<input type="button" value="Скрыть/показать слой" onClick="showHide('sloj');">
</form>

</body>
</html>
```

При загрузке HTML-документа строка **"Этот текст расположен внутри слоя"** видна (рис. 6), т. к. свойство visibility: visible информационного блока

<DIV ID='sloj'>Этот текст расположен внутри слоя </DIV>

установлено в стилевом блоке

#sloj{visibility: visible ; }

После нажатия кнопки **"Скрыть/показать слой"** строка исчезает (рис. 7) т. к. function showHide('sloj') изменяет свойство visibility

document.all[name].style.visibility = "hidden".

Повторное нажатие кнопки заставляет слой вновь стать видимым (рис. 6).

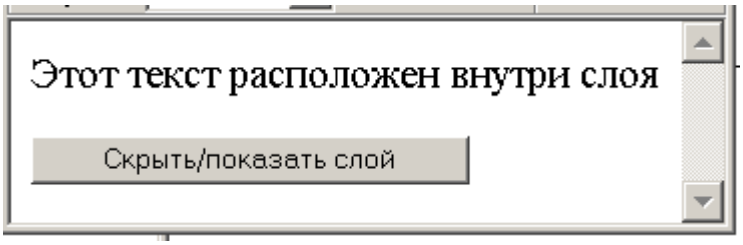

Рис. 6. Результат просмотра **Листинга 8** после загрузки страницы

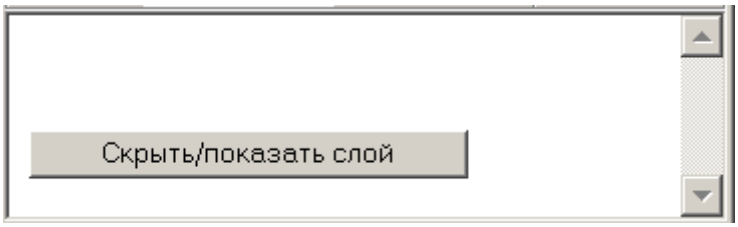

Рис. 7. Результат просмотра **Листинга 8** после нажатия кнопки

1.2.2. Перемещение слоев в окне

Как мы уже знаем, при помощи свойств **left** и **top** можно задать положение слоев. Если изменить значение этих свойств, то расположение слоя изменится. Следующая строчка задает горизонтально расположение слоя'sloj' , равное 100 пикселям:

document.all['sloj'].style.left= 100px;;

Сейчас мы напишем программу, которая будет перемещать слой внутри окна на подобии того, как перемещается простая движущаяся строка по диагонали.

Эти 2 картинки (рис. 8, 9) показывают последовательные положения текста "**Этот текст расположен внутри слоя**" в окне браузера. Текст движется вместе со слоем, относительное положение которого в окне браузера из левого верхнего угла к нижнему правому периодически меняется со временем так, как это описано в нашем скрипте (Листинг 9).

Листинг 9

```
<html>
<head>
<script language="JavaScript">
<!-- hide
var pos = 0;

function move() {
```

```
  if (pos <= 0) direction= true;
  if (pos > 200) direction= false;
  if (direction) pos++
    else pos- -;
  document.all['sloj'].style.left= pos;
  document.all['sloj'].style.top= pos;
}
// -->
</script>
<STYLE TYPE="text/css">
<!--
#sloj
{position: absolute; left: 0px; top: 0px;}  }
//-->
</STYLE>
</head>
<body onLoad="setInterval('move()', 10)">
<DIV ID='sloj'>Этот текст расположен внутри слоя </DIV>
</body>
</html>
```

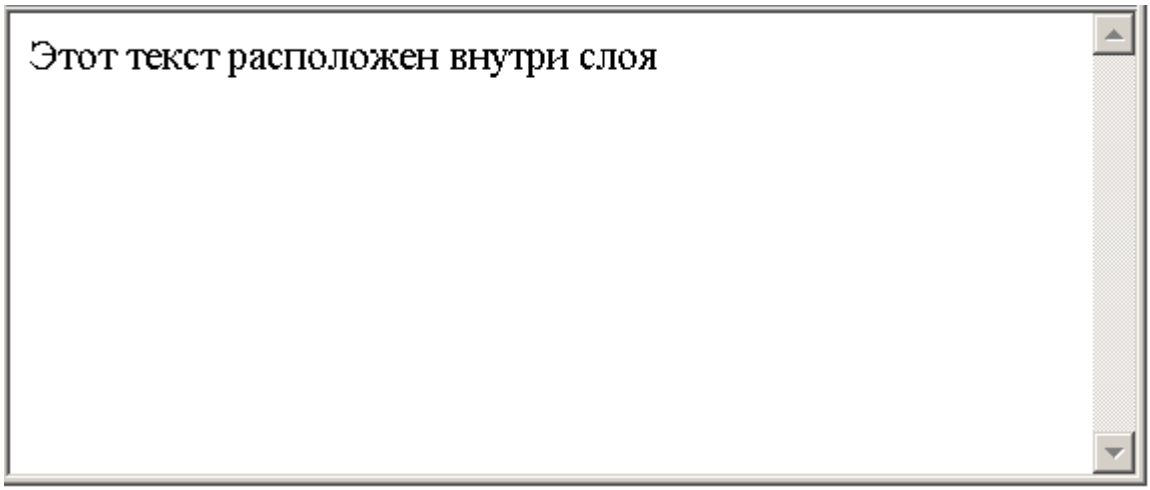

Рис. 8. Результат просмотра **Листинга 9** в момент загрузки

Рис. 9. Результат просмотра **Листинга 9** в следующий момент

Здесь мы создали слой с именем sloj. Внутри тега <body> мы описали свойство onload, потому что мы хотим, чтобы движение этого слоя началось сразу же после загрузки странички. При описании средства обработки событий onLoad мы использовали функцию setInterval(). Этот метод позволяет обращаться к функции множество раз через заданный промежуток времени. Этот метод появился в JavaScript версии 1.2.
Требуется лишь единственное обращение к этому методу. С помощью метода setInterval() происходит обращение к функции move() через каждые **10** миллисекунд. Каждый раз функция move() устанавливает новое положение слоя.
Регулярное обращение к этой функции приводит к появлению эффекта равномерного передвижения слоя по экрану внутри окна браузера. Все что требуется от функции move() — это сосчитать очередные значение координат **X,Y** положения слоя и присвоить их посредством

```
document.all['sloj'].style.left= pos;
document.all['sloj'].style.top= pos;
```

Если пользователь работает со старой версией браузера, "незнакомой" с языком JavaScript версии 1.2, то при выполнении скриптов, приведенных в настоящей главе, возможно появление сообщений об ошибке. Этого можно избежать. Для этого в теге <script> необходимо указать версию языка, т.е. попросту написать так:

```
<script   language=" JavaScript l.2">
<!- hide
...  (здесь расположен текст программы на JavaScript 1.2)
</script>
```

В Листинге 10 и на рис. 10, 11 приведен еще один пример скрипта, демонстрирующего движущиеся в разных направлениях перекрывающиеся слои.

Листинг 10

```
<html>
<head>
<script language="JavaScript">
<!-- hide
var pos = 0;
var v1= "visible"
var v2= "hidden"
function move() {
  if (pos <= 0) {direction= true; v1="visible";v2= "hidden";}
  if (pos > 200) {direction= false;v1="hidden";v2="visible";}
  if (direction) pos++
    else pos--;
    document.all['sloj'].style.left= 100 + pos;
    document.all['sloj2'].style.left=200- pos;
    document.all['sloj1'].style.left=(200- pos)*1.5;
    document.all['sloj1'].style.visibility= v1;
    document.all['sloj2'].style.visibility= v2;
}
// -->
</script>
<STYLE TYPE="text/css">
```

```
<!--
#sloj
{position: absolute; left: 0px; top: 10px;z-index = 1;}
#sloj1
{position: absolute; left: 0px; top: 50px;z-index=2;}
#sloj2
{position: absolute; left: 0px; top: 50px;z-index=0;}
//-->
</STYLE>
</head>
<body onLoad="setInterval('move()', 10)">
<DIV ID='sloj'><img src="img.gif" width=160  height=120></DIV>
<DIV ID='sloj1'>Этот текст расположен перед картинкой </DIV>
<DIV ID='sloj2'>Этот текст расположен за картинкой</DIV>
</body>
</html>
```

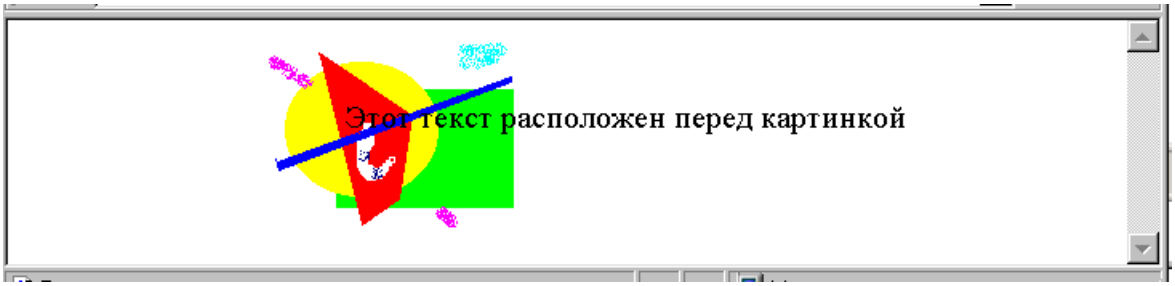

Рис. 10. Результат просмотра **Листинга 10** в момент загрузки

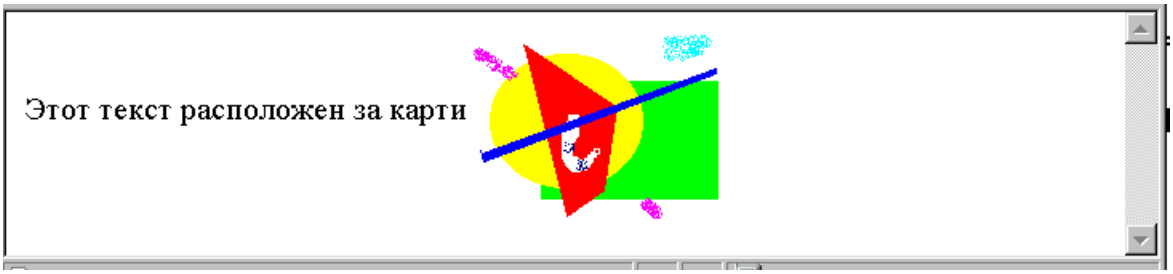

Рис. 11. Результат просмотра **Листинга 10** в следующий момент

Здесь в слое sloj помещен рисунок, а слоях sloj1 и sloj2- текст. При движении sloj1в обратном направлении его скорость увеличивается в 1.5 раза. Он становится видимым, а индекс глубины увеличивается, в следствии чего текст располагается над рисунком. При движении sloj2, который до этого был не видим, в прямом направлении sloj2 становится видимым, а индекс глубины уменьшается, в следствии чего текст располагается под рисунком

Задание

1. Изучите Методические указания или соответствующие разделы пособий по указанной тематике [3,4,7].
2. По аналогии с примером, разобранным в п. 1.1.1, включите в одну из страниц Вашего сайта (**Лабораторная работа № 1**), рисунок типа карты с нанесенными на ней названиями пунктов и другой графической и текстовой информацией. Посетитель сайта должен быстро и в удобном виде получать информацию по каждому из этих пунктов. Для обеспечения этого сервиса примените **динамические информационные блоки.**
3. По аналогии с примерами, разобранными в п. 1.1.2 методических указаний включите в Ваш сайт (**Лабораторная работа № 1**) несколько графических объектов, изменяющих свой вид при наведении указателя мыши. Для изменения вида используйте **визуальные фильтры.**
4. По аналогии с примерами, разобранными в п. 1.2 методических указаний, создайте 2, 3 примера со слоями, повышающие привлекательность страниц Вашего сайта (**Лабораторная работа № 1**), и включите их в Ваш сайт.

Содержание отчета

1. Задание к работе
2. Распечатки страниц Web-узла (.htm-тексты и скриншоты на диске), в которые внесены изменения в соответствии с пунктами 2,3,4 Задания.
3. URL вашего зарегистрированного Web-узла (http://www….)

Библиографический список

1. Холкин И. И. Веб-программирование. Методические указания по выполнению лабораторных и практических работ. Часть 1,2,3. Москва, МТУ МИРЭА, 2017г. (В электронном представлении).
2. Холкин И. И. Интернет- технологии и системы. Методические указания по выполнению лабораторных и практических работ. Часть 1 Москва, МИРЭА, 2017 г. 13 с. (В электронном представлении).
3. Петюшкин А. В. HTML в Web-дизайне. — СПб.: БХВ-Петербург, 2005. — 400 с
4. Николенко Д. В. Практические занятия no JavaScript — СПб: Издательство "НАУКА И ТЕХНИКА", 2000. 128 с
5. Робсон Э., Фримен Э. Изучаем HTML, XHTML и CSS. 2-е изд. — СПб.: Питер, 2014. — 720 с.: ил. — (Серия «Head First O'Reilly»).ISBN 978-5-496-00653-8.
6. Фримен Э., Робсон Э. Изучаем программирование на HTML5. — СПб.: Питер, 2013. — 640 с.: ил.ISBN 978-5-459-00952-1 http://www.ozon.ru/context/detail/id/19024617/
7. Холкин И. И. Интернет-технологии и системы. Методические указания по выполнению лабораторных работ. Часть 2, № 0939, Москва, МИРЭА, 2010 г. 24 с.

Содержание