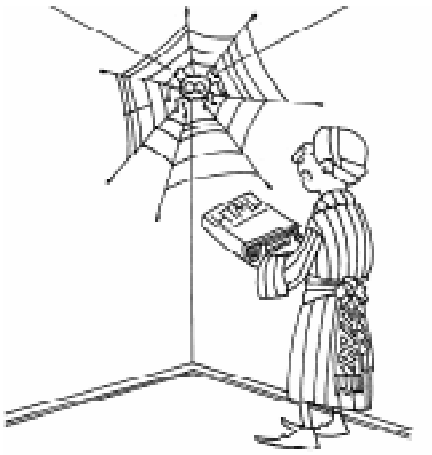
Дмитриева Марина Валерьевна

JAVASCRIPT. ЗАНЯТИЕ 6. JAVASCRIPT И ДИНАМИЧЕСКИЙ HTML

Web-страницы, составляющие всемирную паутину, создаются с помощью языка HTML. Язык HTML задает общую



структуру HTML-документа, он предназначен для построения документов, отображающихся одинаково независимо от платформы и типа программы просмотра.

Динамический HTML (DHTML) является дальнейшим развитием языка HTML, добавляет к HTML набор средств, позволяющих определять внешний вид документа и управлять слоями. При создании Web-страницы с помощью средств DHTML у разработчика появляются дополнительные средства, перечислим некоторые из них.

- Возможно детальное форматирование документов HTML с помощью каскадных таблицей стилей.
- Пользователь может управлять отображением элементов на странице.
- Для элементов страницы предусмотрены различные способы позиционирования их на странице. Абсолютное позиционирование позволяет создавать, в частности, анимационные эффекты.

Детальное форматирование документов HTML достигается использованием технологии каскадных таблиц стилей (Cascade Style Sheet – CSS). Таблицу стилей можно рассматривать как шаблон, который управляет форматированием тегов HTML в Web-документе. Использование таблицы стилей позволяет сократить время форматирования HTML-документов и придать документам привлекательный вид.

Таблица стилей представляет собой набор правил форматирования элементов HTML. В свою очередь любое правило каскадных таблиц стилей состоит из двух частей: селектора и определения. Селектором может быть любой тег HTML, для которого правило определяет, каким образом необходимо его форматировать. Само определение состоит также из двух частей: свойства и значения, разделенных знаком двоеточия. Рассмотрим правило

```
h3 {color: silver}
```

Селектором является элемент **h3**, а определение, заключенное в фигурные скобки, задает значение цвета шрифта (свойство **color**) серым цветом (значение **silver**).

Таблица стилей связывается с документами различными способами. Рассмотрим следующие способы:

- *Связывание*: используется одна таблица стилей для форматирования одной или нескольких страниц HTML.
- *Внедрение*: правила таблицы стилей задаются непосредственно в самом HTML-документе и относятся ко всему документу.
- *Встраивание*: правила задаются в теге документа и позволяют изменять форматирование конкретных элементов страницы.

СВЯЗЫВАНИЕ ТАБЛИЦЫ СТИЛЕЙ С ДОКУМЕНТОМ

Для связывания таблицы стилей и документа следует использовать тег **<LINK>**, задаваемый в разделе **<HEAD>** документа

```
<LINK REL="STYLESHEET"
type="text/css" href="Tabl1.css">
```

В файле с именем **Tabl1.css** задается таблица каскадных стилей, определяющая формат того документа, с которым она будет связана, например, следующего вида:

```
BODY {background-color: yellow}
H3 {text-align: center}
P {color: blue}
```

Правила предписывают определить желтый цвет фона для всего документа, заголовок третьего уровня расположить по центру, цвет шрифта в документе должен быть синим.

ВНЕДРЕНИЕ ТАБЛИЦЫ СТИЛЕЙ В ДОКУМЕНТ

При внедрении таблицы стилей в документ правила, составляющие таблицу стилей, располагаются между тегами **<STYLE TYPE="text/css">** и **</STYLE>**, которые должны размещаться в разделе **<HEAD>** документа. Обозреватели, не поддерживающие теги, игнорируют их, но интерпретируют содержимое тегов. Для того чтобы этого не произошло, содержимое тегов **<STYLE TYPE="text/css">** и **</STYLE>** заключается в теги комментариев. В следующем примере таблица стилей внедрена в документ. Правилами этой таблицы задается синий цвет шрифта. При написании заголовка текст идет «в разрядку». Свойство **letter-spacing** влияет на расстояние между символами при отображении текста. Значение параметра определяет пробел, добавляемый к тому пробелу, который принимается по умолчанию. В рассматриваемом случае пробел увеличен на величину **1.5 em**. Первая строка абзаца начинается с отступа, величина которого задана третьим правилом определенной в документе таблицы стилей.

При задании значений свойств, определяющих некоторые размеры, в таблицах стилей определяются относительные и абсолютные единицы длины. Относительные единицы задают длину относительно значения другого свойства, определяющего длину. Относительная единица измерения **em** определяет высоту шрифта.

HTML-код документа с внедренной таблицей стилей представлен в листинге 1.

На рисунке 1 приведен документ, HTML-код которого рассмотрели.

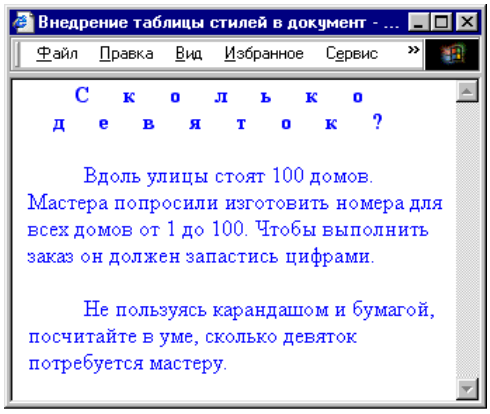


Рисунок 1. Пример документа с внедренной таблицей стилей.

Листинг 1. Внедрение таблицы стилей в документ

```
<html>
<head><title>Внедрение таблицы стилей в документ</title>
<STYLE type="text/css">
<!--
body {color: blue}
h4 {letter-spacing:1.5em; text-align:center}
p {text-indent:2.5em}
-->
</STYLE>
</head>
<body>
<h4>Сколько девяток?</h4>
<p>Вдоль улицы стоят 100 домов. Мастера попросили изготовить номера
   для всех домов от 1 до 100.
   Чтобы выполнить заказ он должен запастись цифрами.</p>
<p>Не пользуясь карандашом и бумагой, посчитайте в уме,
   сколько девяток потребуется мастеру.</p>
</body></html>
```

ВСТРАИВАНИЕ ОПРЕДЕЛЕНИЙ СТИЛЕЙ

В DHTML предоставлена возможность форматирования конкретных элементов страницы, в этом случае правила задаются непосредственно в теге документа.

Создадим документ, в котором первый заголовок и абзац заданы шрифтом красного цвета и расположены слева, второй заголовок и абзац зеленого цвета и расположены по центру. В каждом теге HTML можно задать параметр STYLE и определить значение свойств тега в соответствии с синтаксисом каскадных таблиц стилей. Например, заголовок пятого уровня, отображенный шрифтом синего цвета и выровненный по правому краю, задается следующим образом:

```
<h5 STYLE="color:blue;
                text-align:right">
```

В каждом из тегов **<h5>** и **<p>** задаются значения параметров, определяющих цвет шрифта и горизонтальное выравнивание элемента.

HTML-код документа, в котором для некоторых тегов заданы стили, определяющие форматирование тега, приведен в листинге 2.

На рисунке 2 показано, как располагаются в документе абзацы и заголовки.

Напомним, что параметр **id** задает уникальное имя документа, которое мож-

Рисунок 2. Пример документа с встроенными стилями.

Листинг 2. Встраивание стиля в тег документа

```
<html>
<head><title>Встраивание стиля в тег документа</title></head>
<body>
<h5 id=el1 STYLE="color:red; text-align:left">Афродита</h5>
<p STYLE="color: red; text-align: left">
 В античной мифологии Афродита - богиня любви и красоты, покровительница
 брака. Во многих мифах Афродита воспевалась как богиня плодородия,
 дарующая жизнь растительному и животному миру. </p><hr>
<h5 STYLE="color:green; text-align:center">Гера</h5>
<p STYLE="color:green; text-align:center">
 В античной мифологии Гера - супруга Зевса, главная среди богинь Олимпа,
 покровительница брака и семьи. В произведениях искусства богиня Гера
 изображалась в диадеме, со скипетром, часто с символом звездного неба -
 павлином. </p><hr>
<h5 STYLE="color:blue; text-align:right">Гермес</h5>
<p STYLE="color:blue; text-align:right">
 Гермес - вестник богов, проводник душ умерших в подземное царство Аида,
 а также хитрый и изворотливый покровитель торговли. Иногда Гермес
 почитался и как покровитель искусства. </p><hr></body></html>
```

но использовать для ссылок на элемент в сценариях. Параметр **id** можно задать в любом теге. Все объекты, расположенные на странице, содержатся в наборе **document.all**. Доступ к свойствам элемента с уникальным именем **el1** осуществляется с помощью конструкции **document.all['el1'].style**. Если бы требовалось изменить цвет шрифта заголовка, то сделать это можно было бы с помощью оператора присваивания

```
document.all['el1'].style.color = 'blue'
```

ИЗМЕНЕНИЕ СВОЙСТВ ЭЛЕМЕНТА ПРИ ДЕЙСТВИЯХ ПОЛЬЗОВАТЕЛЯ

Просматривая страницы при работе в Интернет, Вы, наверно, обратили внимание, что часто при попадании курсора мыши на некоторое слово цвет шрифта изменяется. Такой эффект используется при организации текстовых меню.

Свойства элемента можно изменять в результате реакции на события, связанные с элементом. Создадим документ, в котором при попадании курсора мыши на текст заголовка изменяется цвет шрифта. При перемещении курсора мыши с области заголовка, восстанавливается первоначальный цвет шрифта.

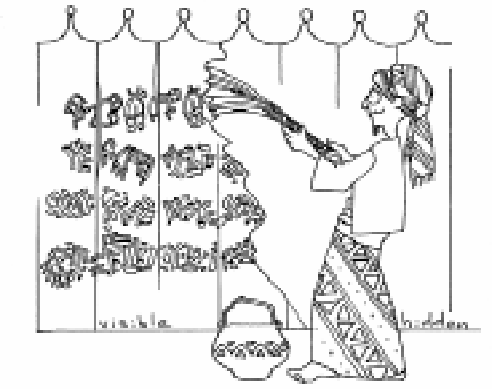
Ключевое слово **this** языка JavaScript используется для ссылки на текущий объект (в данном случае тег), применяется при вызове функции обработки события.

Эта задача очень простая, и для ее решения можно использовать HTML-код, приведенный в листинге 3.

Листинг 3. Изменение цвета заголовка

```
<HTML>
<HEAD><TITLE>Изменение цвета
 заголовка</TITLE></HEAD>
<BODY bgColor=silver>
<H3 align=center
  onmouseout="this.style.color='black'"
  onmouseover="this.style.color='white'">
Изменение цвета заголовка</H3>
</BODY></HTML>
```

ОТОБРАЖЕНИЕ ЭЛЕМЕНТОВ НА WEB-СТРАНИЦЕ



Многих анимационных эффектов можно достичь, скрывая или отображая некоторые элементы на странице. В DHTML предусмотрены средства, которые позволяют управлять отображением и скрытием элементов на странице HTML. Свойство **visibility** управляет отображением элемента. Если его значение равно **visible**, то элемент отображается, если значение равно **hidden**, то элемент на странице не отображается. По умолчанию элемент отображается на странице.

СКРЫТИЕ И ОТОБРАЖЕНИЕ ТЕКСТА

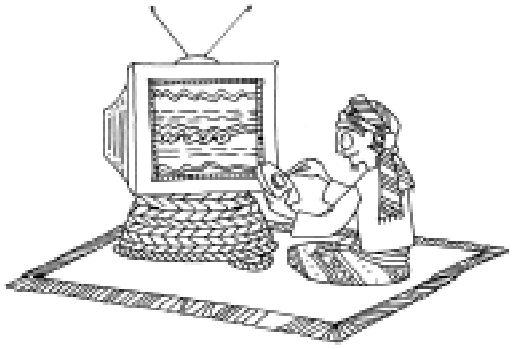
Напишем сценарий, который позволяет отображать или скрывать фрагмент текста. При щелчке по соответствующей кнопке изменяется значение свойства объекта, управляющего отображением элемента на странице. Тег **** используется в случаях, когда требуется отметить фрагмент текста для задания его свойств. В приведенном примере вместо тега **** можно использовать тег **<p>**. Доступ к свойству отображения осуществляется с помощью конструкции **document.all["l1"].style.visibility**. Значение этого свойства зависит от текущего состояния и кнопки, которая была нажата. HTML-код, содержащий сценарий, представлен в листинге 4.

СКРЫТИЕ И ОТОБРАЖЕНИЕ ИЗОБРАЖЕНИЯ

Напишем сценарий, который позволяет скрывать и отображать изображение. Воспользуемся для управления отображением свойством **visibilite**. Управлять отображением будем с помо-

Листинг 4. Фрагмент текста скрывается или отображается

```
<HTML>
<HEAD>
<TITLE>Отображение или скрытие фрагмента текста</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!-- //
function showsl (n)
{ var el=document.all["l1"].style
if (n==1)
 el.visibility='visible'
else
 el.visibility='hidden'
}
//–>
</SCRIPT></HEAD>
<BODY>
<h4 align=center>Текст в документе<h4>
<FORM name="form1">
 <input type="button" value=Показать onClick="showsl(1)">
 <input type="button" value=Скрыть onClick= "showsl(2)">
</FORM><hr>
<span id="l1">Текст, который либо показываем, либо скрываем</span>
</BODY></HTML>
```



щью одной кнопки. На кнопке помещается надпись с указанием действия, которое можно выполнить. Когда изображение отображено на странице, то его можно скрыть, и, наоборот, когда изображение скрыто, его можно отобразить.

Каждый раз при выполнении сценария меняется свойство **value** кнопки, поэтому надпись на кнопке соответствует действию, которое можно выполнить применительно к изображению.

HTML-код документа с изображением, которое можно скрыть и отобразить, приведен в листинге 5.

На рисунке 3 представлен документ, в котором располагается текст, изображение и кнопка, управляющая отображением в документе изображения.

Листинг 5. Объект со свойством **visibility**

```
<HTML>
<HEAD>
<TITLE>Текст и изображение. Свойство visibility</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!-- //
var d=document
function showsl ()
{ if (d.form1.but1.value == "Отобразить")
 {d.all["mypict"].style.visibility='visible'
 d.form1.but1.value="Скрыть"
 }
 else
 { d.all["mypict"].style.visibility='hidden'
 d.form1.but1.value="Отобразить"
 }
}
//-->
</SCRIPT></HEAD>
<BODY>
<center>
<h4>Отображение элемента на странице</h4>
<FORM name="form1">
 <input type="button" value=Скрыть name =but1 onClick="showsl()">
</FORM>
<hr>
<h3>Леонардо да Винчи</h3>
<h4>Мадонна с цветком (Мадонна Бенуа)</h4>
<img id="mypict" src=pict.jpg height=150 STYLE="visibility='visible'" >
<p align=left>Раннее произведение гениального итальянского мастера
 Высокого Возрождения Леонардо да Винчи, жившего в то время во Флоренции,
 известно под названием "Мадонна с цветком".</p>
<p align=left>Мадонна представлена как юная прелестная флорентийка,
 в одеждах богатой горожанки. Она весело смотрит на своего ребенка,
 которому уютно и удобно сидеть у нее на коленях.</p>
</center>
<i>Д.С. Буслович</i>
</BODY></HTML>
```

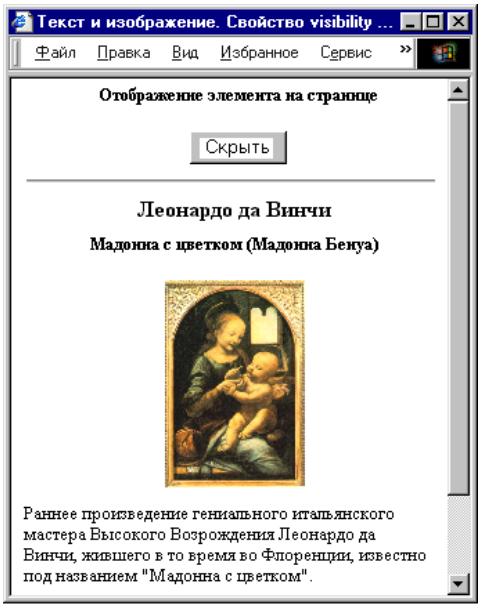


Рисунок 3. Отображение изображения в документе.

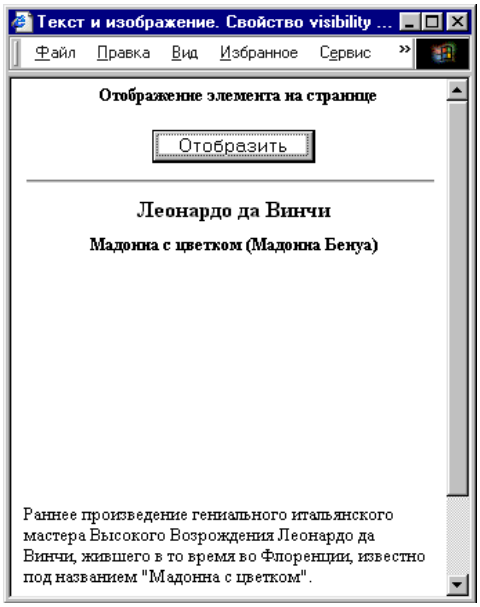
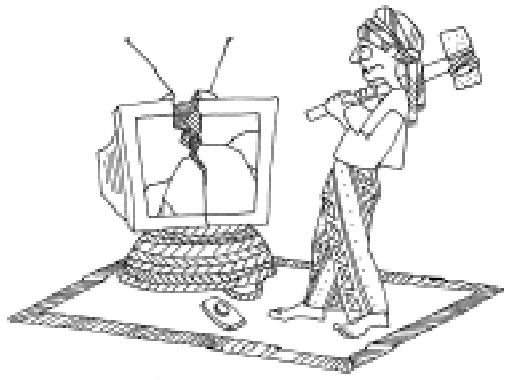


Рисунок 4. Документ со скрытым изображением, обладающим свойством **`visibility`**.

Если нажать на кнопку **Скрыть**, то рисунок не отображается в документе, но место под него оставлено (рисунок 4). После нажатия на кнопку **Отобразить** документ примет первоначальный вид.

УПРАВЛЕНИЕ ОТОБРАЖЕНИЕМ С ПОМОЩЬЮ СВОЙСТВА `DISPLAY`



Управлять отображением документа на странице можно и с помощью свойства **`display`**. Объект со свойством **`display`** ведет себя иначе, чем объект со свойством **`visibility`**. Если значение **`display`** равно **`none`**, то элемент не только не отображается на странице, но на странице нет и пустого блока, соответствующего этому элементу.

Напишем сценарий управления отображением на странице объекта, обладающего свойством **`display`**. Как и в предыдущем примере, будем использовать одну кнопку для выполнения действий. Приведем в листинге 6 HTML-код документа, содержащего сценарий управления

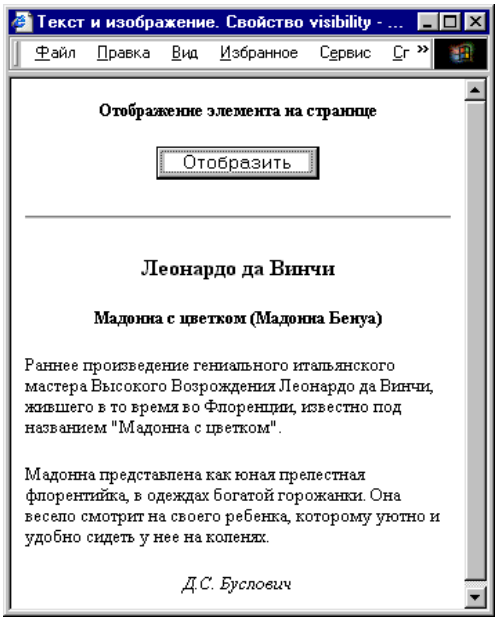


Рис. 5. Документ со скрытым изображением, обладающим свойством **`display`**.

Листинг 6. Объект со свойством display

```
<HTML>
<HEAD>
<TITLE>Текст и изображение. Свойство visibility</TITLE>
<SCRIPT LANGUAGE=»JavaScript»>
<!-- //
var d=document
function showsl ()
{ if (d.form1.but1.value == «Отобразить»)
 {d.all[«mypict»].style.display='block'
 d.form1.but1.value=»Скрыть»
 }
 else
 { d.all[«mypict»].style.display='none'
 d.form1.but1.value=»Отобразить»
 }
}
//—>
</SCRIPT>
</HEAD>
<BODY>
<center>
<h4>Отображение элемента на странице</h4>
<FORM name=»form1">
 <input type=»button» value=Скрыть name =but1 onClick=»showsl()»>
</FORM>
<center>
<hr>
<h3 align=center>Леонардо да Винчи</h3>
<h4 align=center>Мадонна с цветком (Мадонна Бенуа)</h4>
<img id=»mypict» src=pict.jpg height=200 STYLE=»display='block'»
align=center>
<p align=left>Раннее произведение гениального итальянского мастера
Высокого Возрождения Леонардо да Винчи, жившего в то время во Флоренции,
известно под названием «Мадонна с цветком».</p>
<p align=left>Мадонна представлена как юная прелестная флорентийка,
в одеждах богатой горожанки. Она весело смотрит на своего ребенка,
которому уютно и удобно сидеть у нее на коленях.</p>
<i>Д.С. Буслович</i>
</BODY></HTML>
```

отображением изображения со свойством **display**.

На рисунке 5 документ со свойством **display** после того, как нажата кнопка **Скрыть**. На странице не оставлено пустого блока под изображение, иными словами, элемент не только не отображается, а исключается из потока отображения в отличие от случая, рассмотренного ранее.

Используя свойства объекта **visibility** и **display**, можно создавать простые анимационные эффекты. Ранее для простой анимации использовался прием: загружалось изображение, которое затем заменялось пустым. В свою очередь, одно из пустых изображений заменялось реальным. Для создания простого анимационного эффекта можно управлять свойством **visibility** объекта. Например, для создания эффекта движения слева направо можно лишь изменять значения свойства **visibility** рядом расположенных

элементов. Для создания более сложных эффектов, например, светящейся гирлянды на новогодней елке, можно изменять свойства отображения случайно выбранных элементов и т. п. Свойство **`display`** часто используется для организации меню. Для того чтобы «раскрыть» пункт меню, следует «раздвинуть» пункты основного меню, вставив подпункты следующего уровня.

РЕКОМЕНДАЦИИ ПО ПОДГОТОВКЕ ТЕКСТА ДЛЯ ПУБЛИКАЦИИ В ИНТЕРНЕТ

Большая часть информации, которая размещена в сети Интернет, представлена в виде текста. Текст должен помогать восприятию информации. При подготовке текста к публикации в Интернет следует обратить внимание на особенности, которые отличают текст в Интернет от текста, например, печатного издания.

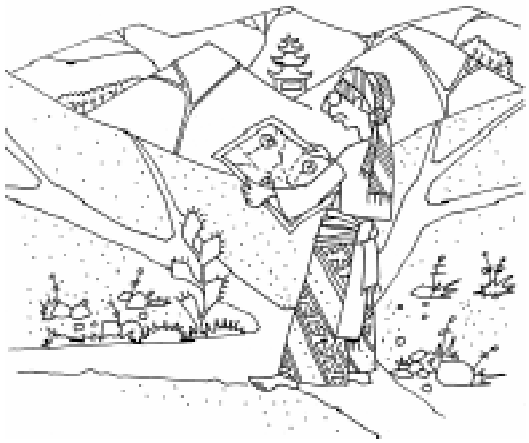
Если текст большого объема, то рекомендуется его разбить на части, причем каждой части дать заголовок. Заголовки

должны быть короткими. Пользователь сможет лишь просмотреть заголовки и перейти сразу к нужному разделу. Иногда разумно привести аннотацию каждой части.

В текстах для Интернет при составлении оглавления или просто при перечислении полезно пользоваться списками как маркированными, так и нумерованными. Информация, представленная списком, как правило, легче воспринимается пользователем.

Каждый раздел текста состоит из абзацев. Рекомендуется использовать небольшие абзацы, которые содержат, пять, шесть предложений. Предпочтение отдается коротким предложениям. Рекомендуется основную мысль изложить в первых предложениях, так как многочисленные исследования показали, что пользователи Интернет часто просто бегло просматривают текст.

При организации сайта следует тщательно продумать навигацию по документам, то есть способы представления документов и переходы по гиперссылкам. Навигация должна быть удобной и понятной пользователю, чтобы он не заблудился в дебрях сайта. При разработке сайта рекомендуется иметь перед глазами схему переходов, и надо стараться, чтобы крат-



чайший переход от одного документа к другому состоял не более, чем из двух шагов. Хорошо продуманная система навигации позволяет пользователю быстро и правильно ориентироваться в представленной информации.

ЗАКЛЮЧЕНИЕ

Начиная работу над Web-сайтом, надо четко представлять себе цели его создания, для кого он предназначен, какая информация важна для посетителей сайта, как посетители могут этой информацией воспользоваться. Цели создания Web-сайта должны определять его содержание, тематику разделов, иллюстративный материал, средства оформления. Образ любой страницы должен соответствовать ее назначению. На стадии разработки следует представлять себе аудиторию, для которой предназначено содержание

сайта. Чем шире предполагаемая аудитория, тем сложнее представить информацию, которая заинтересует многих.

Рассмотренных средств достаточно для создания собственного Web-приложения. Разбиение материала на разделы, Web-сайта на страницы, обеспечение навигации по страницам сайта, выбор вида и организация навигационных панелей, подбор иллюстративного материала, создание анимационных эффектов – все это зависит от профессиональных возможностей, художественных способностей и фантазии автора.

Разработчики JavaScript предоставляют информацию, полезную для авторов, создающих Web-приложения, на сайте http://developer.netscape.com/library/documentation/.

Задания.

1. В документе в одной строке располагаются несколько изображений. Напишите сценарий, который разрешает пользователю вводить номер изображения, которое должно стать невидимым. Все остальные изображения должны сдвинуться влево на освободившееся место.

2. Напишите сценарии, обеспечивающие визуальные эффекты на основе свойств отображения элементов, например, при реализации вертикальных и горизонтальных графических меню.

3. Напишите сценарий, обеспечивающий анимационный эффект движения изображений на основе свойств отображения элементов.

4. Создайте сценарий, при выполнении которого все изображения, хранящиеся в документе, выравниваются либо по левому, либо по правому краю. Управление выравниванием изображений осуществляется с помощью кнопок. Выбор способа выравнивания осуществляется переключателем.

5. Создайте сценарий, обеспечивающий изменение цвета шрифта пунктов текстового меню при попадании курсора мыши на соответствующий пункт.

Дмитриева Марина Валерьевна, доцент кафедры информатики математико-механического факультета Санкт-Петербургского государственного университета.