17. Программирование графики

Цель: обучиться программированию графики с использование JavaScript, применить на практике добавление такой графики к страницам, выделить основные преимущества использования JavaScript при программировании графики.

17.1. Теоретические сведения

17.1.1. Объект Image

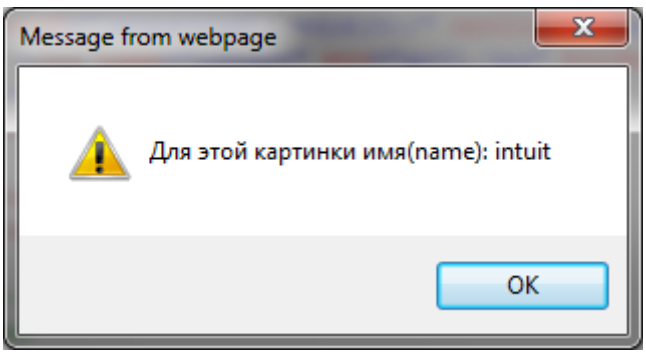
Наиболее зрелищные эффекты при программировании на JavaScript достигаются при работе с графикой. Программирование графики в JavaScript опирается на объект Image, который характеризуется следующими свойствами, методами и событиями:

Свойства	Методы	События
• border • complete • height • hspace • name • src • vspace • width • lowscr	• нет	• onAbort • onError • onLoad

Несмотря на такое обилие свойств, их абсолютное большинство можно только читать, но не изменять. Об этом свидетельствует, прежде всего, отсутствие методов. Но два свойства все же можно изменять: src и lowsrs. Этого оказывается достаточно для множества эффектов с картинками.

Все объекты класса Image можно разделить на встроенные и порожденные программистом. Встроенные объекты - это картинки контейнеров IMG. Если эти картинки проименовать, к ним можно обращаться по имени:

```
<form>
<a href="javascript: void(0);" onclick="window.alert('Для этой картинки имя(name): '+ document.images[0].name)" >
        <img name="intuit" src="gr01.jpg" border="0" alt="" />
</a>
</form>
```



К встроенному графическому объекту можно обратиться и по индексу:

```
document.images[0];
```

В данном случае images[0] — это первая картинка документа.

17.1.2. src и lowsrc

Свойства src и lowsrc определяют URL изображения, которое монтируется внутрь документа. При этом lowsrc определяет временное изображение, обычно маленькое, которое отображается, пока загружается основное изображение. Свойство src принимает значение атрибута SRC контейнера IMG. Рассмотрим пример с src:

```
document.i2.src = "images2.gif";
```

Как видно из этого примера, существует возможность модифицировать вмонтированную картинку за счет изменения значения свойства src встроенного объекта Image. Если вы в первый раз просматриваете данную страницу, то постепенное изменение картинки будет заметно.

17.1.3. Изменение картинки

Изменить картинку можно, только присвоив свойству src встроенного объекта Image новое значение. Очевидно, что медленная перезагрузка картинки с сервера не позволяет реализовать быстрое листание. Попробуем решить эту проблему.

Собственно, решение заключается в разведении по времени подкачки картинки и ее *отображения.* Для этой цели используют конструктор объекта Image:

```
<TABLE>
        <TD>
                <A HREF="javascript:void(0)"; onMouseover="document.m0.src=color[0].src; return true;
"onMouseout="document.m0.src=mono[0].src;return true; ">
                        <IMG NAME=m0 SRC="images0.gif" border=0>
                </A>
        </TD>
</TABLE>

<form>
<table>
        <td>
                <a href="javascript: void(0)"; onmousemove="document.m0.src='gr02.jpg'; return true; "
                   onmouseout="document.m0.src='gr01.jpg'; return true; ">
                        <img name="m0" src="gr01.jpg" border="0" alt="" />
                </a>
        </td>
</table>
</form>
```

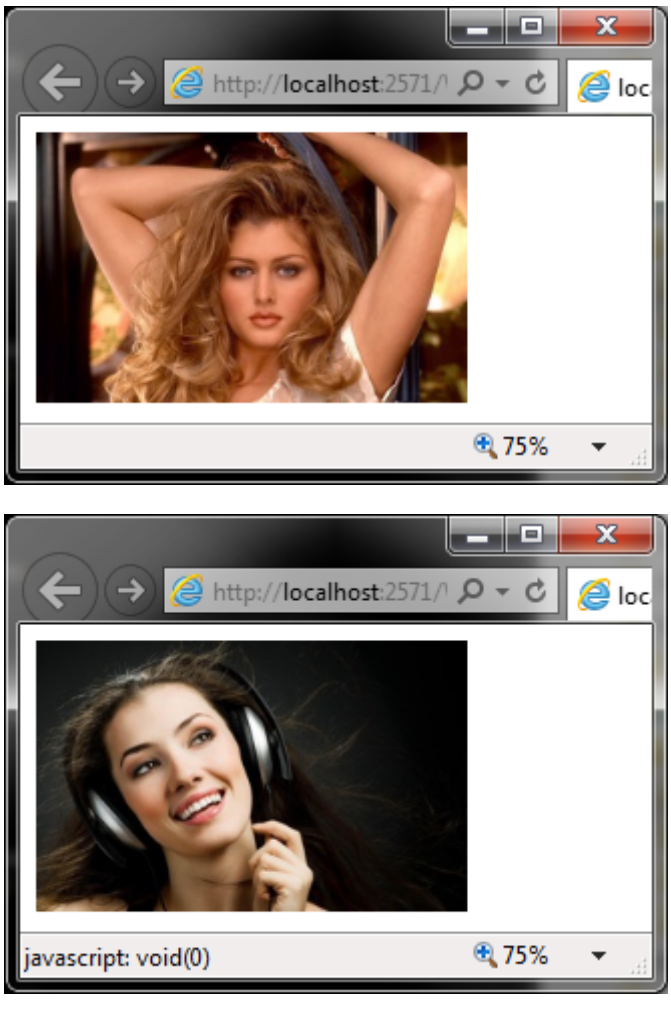


Рисунок 17.3. – Пример изменения картинки

Фрагмент кода показывает типовой прием замещения и восстановления картинки при проходе курсора мыши. Естественно, что менять можно не одну, а сразу несколько картинок.

```
<script language="javascript" type="text/javascript">
   var i = 1;
   //var image = document.getElementById("image");
   var image = document.getElementById("image");
    var imgs = new Array("gr00.jpg", "gr01.jpg", "gr02.jpg", "gr03.jpg"); //Добавьте свои картинки через запятую
   function imgsrc() {
      if (i > 3) i = 0;
      //image.src = imgs[i];
      document.images[0].src = imgs[i]
      i++;
   }
</script>

</head>
<body>

      <img id="image" src="gr00.jpg" onclick="imgsrc();" alt="" />

</body>
</html>
```

Главное, не то, что картинки замещаются, а то, с какой скоростью они это делают. Для достижения нужного результата в начале страницы создаются массивы картинок, в которые перед отображением перекачивается *графика* (обратите внимание на строку статуса при загрузке страницы):

```
color = new Array(32);
mono = new Array(32);
for(i=0; i<32; i++)
      {
      mono[i] = new Image(); color[i] = new Image(); if(i. toString(). length==2)
            {
                  mono[i]. src = "images0"+i+". gif"; color[i]. src = "images0"+i+". gif";
            }
            else
            {
                  mono[i]. src = "images00"+i+". gif"; color[i]. src = "images00"+i+". gif";
            }
      }

<head>
   <title></title>
<script language="javascript" type="text/javascript">
   var ImgNum = 0;
   var image = document.getElementById("slide_show");
   var all_images = new Array(); //
   var ImgLength = 12;

   for (i = 0; i < ImgLength; i++) {
      all_images[i] = new Image();
      if (i.toString().length >= 2) {
         all_images[i] = "gr" + i + ".jpg";
      }
      else {
         all_images[i] = "gr0" + i + ".jpg";
      }
   }

   function chgImg(direction) {
      if (document.images) {
         ImgNum = ImgNum + direction;
         if (ImgNum >= ImgLength) { ImgNum = 0; }
         if (ImgNum < 0) { ImgNum = ImgLength; }
```

```
            document.image.src = all_images[ImgNum];
            //document.images[0].src = all_images[ImgNum];
        }
    }
</script>

</head>
<body>
        <img id="slide_show"
            src="gr01.jpg" alt=""
            onclick="chgImg(1);"
             />
</body>
```

Еще один характерный прием — применение функции отложенного исполнения JavaScript-кода (eval()).

В данном случае eval() избавляет нас от необходимости набирать операции присваивания.

А теперь будем использовать временную задержку:

```
<script type="text/javascript">
// Предварительная загрузка изображений
 numimg=0
 imgslide=new Array()
 imgslide[0]=new Image()
 imgslide[1]=new Image()
 imgslide[2]=new Image()
 imgslide[3]=new Image()
 imgslide[0].src="gr00.jpg"
 imgslide[1].src="gr01.jpg"
 imgslide[2].src="gr02.jpg"
 imgslide[3].src="gr03.jpg"
 //чередование изображений
 function demoslides(){
   document.images[0].src=imgslide[numimg].src
   numimg++;
   if(numimg == 4) numimg=0;
   setTimeout("demoslides()", 1000)
 }
</script>

</head>
<body bgcolor="#FFFFFF" onload="demoslides()">
   <img src="gr00.jpg" width="200" height="150" alt="" />

</body>
```

17.1.4. Мультипликация

Естественным продолжением идеи замещения значения атрибута SRC в контейнере IMG является мультипликация, т. е. последовательное изменение значения этого атрибута во времени. Для реализации мультипликации используют метод объекта Window - setTimeout().

Собственно, существует два способа запуска мультипликации:

onLoad();
onClick(), onChange()...

Наиболее популярный — setTimeout() при onLoad().

17.1.4.1. Событие onLoad()

Событие onLoad() наступает в момент окончания загрузки документа браузером. Обработчик события указывается в контейнере BODY:

Рассмотрим пример по анимации. В нашем случае при загрузке документа должен начать выполняться цикл изменения картинки:

```
<html>
<head>
 <title>JavaScript Animation</title>
```

```
 <script type="text/javascript">
    var i = 0;
    function movie() {
        eval("document.images[0].src='gr0" + i + ".jpg';");
        i++;
        if (i > 6) i = 0;
        setTimeout("movie();", 300);
    }
 </script>

</head>

<body onload="movie();">

    <img id="myImage" src="" alt="" />

</body>
</html>
```

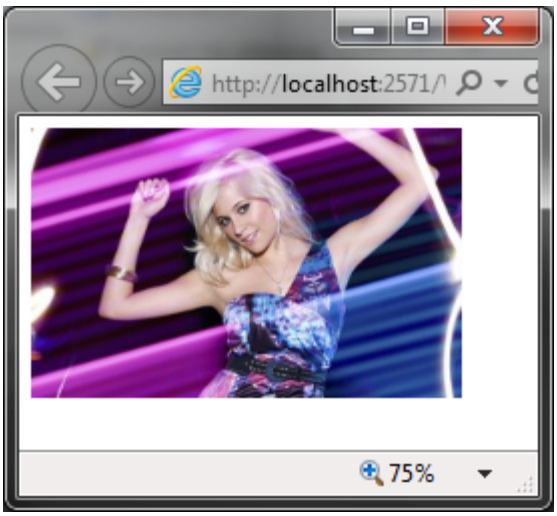


Рисунок 17.2. – Пример использования бесконечного цикла в анимации

Можно реализовать конечное число подмен:

```
<html>
<head>
 <title>JavaScript Animation</title>
 <script type="text/javascript">
    var i = 0;
    function movie() {
        eval("document.images[0].src='gr0" + i + ".jpg';");
        i++;
        if (i < 7) {
            setTimeout("movie();", 500);
        }
    }

 </script>

</head>

<body onload="movie();">

    <img id="myImage" src="" alt="" />

</body>
</html>
```

В обоих примерах следует обратить внимание на использование метода setTimeout(). На первый взгляд, это просто рекурсия. Но в действительности все несколько сложнее. JavaScript разрабатывался для многопоточных операционных систем, поэтому правильнее будет представлять себе исполнение скриптов следующим образом:

1. Скрипт получает управление при событии onLoad().
2. Заменяет картинку.
3. Порождает новый скрипт и откладывает его исполнение на 500 миллисекунд.
4. Текущий скрипт уничтожается JavaScript-интерпретатором.

После окончания срока задержки исполнения все повторяется. В первом примере (бесконечное повторение) функция порождает саму себя и, тем самым, поддерживает непрерывность своего выполнения. Во втором примере (конечное число итераций) после шести повторов функция не порождается. Это приводит к завершению процесса отображения новых картинок.

17.1.4.2. Запуск и остановка мультипликации

Перманентная мультипликация может быть достигнута и другими средствами, например многокадровыми *графическими* файлами. Однако движение на странице — не всегда благо. Часто возникает желание реализовать запуск и остановить движения по требованию пользователя. Удовлетворим это желание, используя предыдущие примеры (запустить или остановить мультипликацию):

```
<html>
<head>
  <title>JavaScript Animation</title>
  <script type="text/javascript">
    var i = 0;
    var flag1=0;
    function movie() {
              if(flag1 == 0) {
                 eval("document.images[0].src='gr0" + i + ".jpg';");
                       i++;
                       if(i>6) i=0;
          }
          setTimeout("movie();", 500);
}

  </script>

</head>

<body onload="movie();">
    <img id="myImage" src="" alt="" /><br /><br />
    <input type="button" value="Start/Stop" onclick="if(flag1==0) flag1=1; else flag1=0; " />
</body>
</html>
```

В данном случае мы просто обходим изменение картинки, но при этом не прекращаем порождение потока. Если мы поместим setTimeout() внутрь конструкции if(), то после нажатия на кнопку Start/Stop поток порождаться не будет, и запустить движение будет нельзя.

Существует еще один способ решения проблемы остановки и старта мультипликации. Он основан на применении метода clearTimeout(). Внешне все выглядит по-прежнему, но процесс идет совсем по-другому:

```
<html>
<head>
  <title>JavaScript Animation</title>
  <script type="text/javascript">
    var i = 0;
          var flag1;
          var id1;
          function movie() {
             eval("document.images[0].src='gr0" + i + ".jpg';");
             i++;
             if (i > 6) i = 0;
             id1 = setTimeout("movie(); ", 500);
          }

  </script>
```

```
</head>

<body onload="movie();">
   <img id="myImage" src="" alt="" /><br /><br />
   <input type="button" value="Start/Stop"
      onclick="if (flag1 == 0)
               { id1 = setTimeout('movie(); ', 500); flag1 = 1; }
            else
               { clearTimeout(id1); flag1 = 0; }; " />

</body>
</html>
```

Обратите внимание на два изменения. Во-первых, объявлен и используется идентификатор потока (id1); во-вторых, применяется метод clearTimeout(), которому, собственно, идентификатор потока и передается в качестве аргумента. Чтобы остановить воспроизведение функции movie () достаточно «убить» поток.

17.1.5. Оптимизация отображения

Обычная дилемма оптимизации программ — скорость или размер занимаемой памяти — решается только путем увеличения скорости. О размере памяти при программировании на JavaScript думать не принято.

Из всех способов оптимизации отображения картинок мы остановимся только на нескольких:

- оптимизация отображения при загрузке;
- оптимизация отображения за счет предварительной загрузки;
- оптимизация отображения за счет нарезки изображения.

Если первые две позиции относятся в равной степени как к отображению статических картинок, так и к мультипликации, то третий пункт характерен главным образом для мультипликации.

17.1.5.1. Оптимизация при загрузке

Практически в любом руководстве по разработке HTML-страниц отмечается, что при использовании контейнера IMG в теле HTML-страницы следует указывать атрибуты WIDTH и HEIGHT. Это продиктовано порядком загрузки компонентов страницы с сервера и алгоритмом работы HTML-парсера.

Первым загружается текст разметки. После этого парсер разбирает текст и начинает загрузку дополнительных компонентов, в том числе *графики.* При этом *загрузка картинок,* в зависимости от типа HTTP-протокола, может идти последовательно или параллельно.

Также параллельно с загрузкой парсер продолжает свою работу. Если для картинок заданы параметры ширины и высоты, то можно отформатировать текст и отобразить его в окне браузера. До тех пор, пока эти параметры не определены, отображения текста не происходит.

Таким образом, указание высоты и ширины картинки позволит отобразить документ раньше, чем картинки будут получены с сервера. Это дает пользователю возможность читать документ или задействовать его гипертекстовые ссылки до момента полной загрузки (событие load).

С точки зрения JavaScript, указание размеров картинки задает начальные параметры окна отображения *графики* внутри документа. Это позволяет воспользоваться маленьким прозрачным образом, для того, чтобы заменить его полноценной картинкой. Идея состоит в передаче маленького объекта для замещения его по требованию большим объектом.

17.1.5.2. Предварительная загрузка

Замена одного образа другим часто бывает оправдана только в том случае, когда это происходит достаточно быстро. Если перезагрузка длится долго, то эффект теряется. Для быстрой подмены используют возможность предварительной загрузки документа в специально созданный объект класса Image.

Реальный эффект можно почувствовать только при отключении кэширования страниц на стороне клиента (браузера). Кэширование часто используют для ускорения работы со страницами Web-узла. Как правило, загрузка первой страницы - это достаточно длительный процесс. Самое главное, чтобы пользователь в этот момент был готов немного подождать. Поэтому, кроме *графики,* необходимой только на первой странице, ему можно передать и *графику,* которая на ней не отображается. Но зато при переходе к другим страницам узла она будет отображаться без задержки на передачу с сервера.

Описанный выше прием неоднозначен. Его оправдывает только то, что если пользователь нетерпелив, то он вообще отключит передачу *графики.*

17.1.6. Нарезка картинок

Нарезка картинок применяется довольно часто. Она позволяет достигать эффекта частичного изменения отображаемой картинки. Чаще всего он применяется при создании меню.

Кроме подобного эффекта нарезка позволяет реализовать мультипликацию на больших картинках. При этом изменяется не весь образ, а только отдельные его части.

17.1.7. Графика и таблицы

Одним из наиболее популярных приемов дизайна страниц Web-узла является техника нарезки картинок на составные части. Можно выделить следующие способы применения этой техники для организации навигационных компонентов страницы:

- горизонтальные и вертикальные меню;
- вложенные меню;
- навигационные графические блоки.

Главной проблемой при использовании нарезанной *графики* является защита ее от контекстного форматирования страницы HTML-парсером. Дело в том, что он автоматически переносит элементы разметки на новую строку, если они не помещаются в одной. Составные части нарезанной картинки должны быть расположены определенным образом, поэтому простое их перечисление в ряд не дает желаемого эффекта.

Элементы переносятся на новую строку, так как ширина раздела меньше общей ширины всех картинок. Проблема решается, если применить защиту от парсера – <PRE>.

Использование такого меню требует определения на нем гипертекстовых ссылок:

```
<PRE>
        <A HREF="javascript: void(0); ">
                <IMG SRC="image1.gif" BORDER="0"></A>
        <A HREF="javascript: void(O); ">
                <IMG SRC="image2.gif" BORDER="0"></A>
        <A HREF="javascript: void(0); ">
                <IMG SRC="image3.gif" BORDER="0"></A>
        <A HREF="javascript: void(0); ">
                <IMG SRC="image4.gif" BORDER="0"></A>
</PRE>
```

```
<pre>
        <a href="javascript: void(0) "><img id="im01g" src="im01g.gif" style="border-style: solid;" alt="" /></a><a href="javascript: void(0); "><img id="im02g" style="border-style: solid;" src="im02g.gif" alt="" /></a><a href="javascript: void(0); "><img id="im03g" style="border-style: solid;" src="im03g.gif" alt="" /></a><a href="javascript: void(0); "><img id="Img04" style="border-style: solid;" src="im04g.gif" alt="" /></a>
</pre>
```

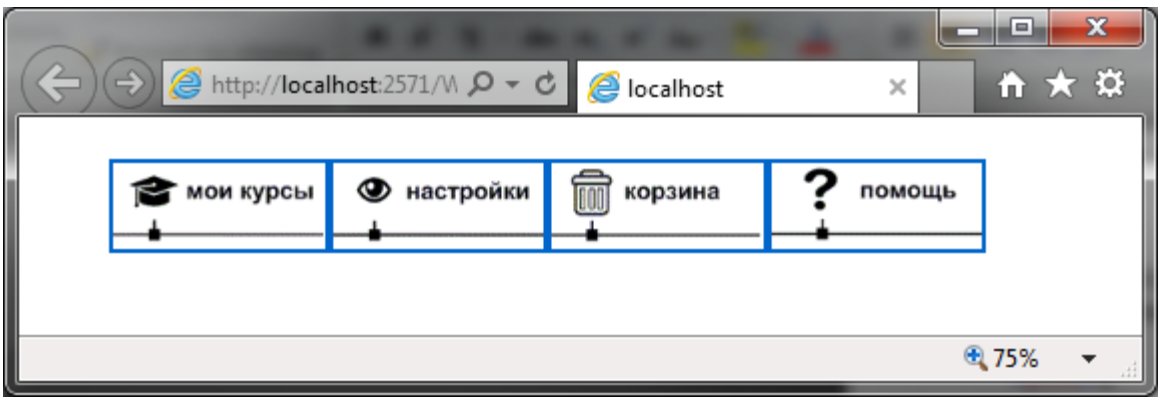


Рисунок 17.3. – Расположение элементов изображений в одну строку

Если мы попробуем тем же способом реализовать многострочное меню, то сплошной картинки при этом не получается, так как высота строки не равна высоте картинки. Подогнать эти параметры практически невозможно. Каждый пользователь настраивает браузер по своему вкусу. Решение заключается в использовании таблицы:

Листинг 16.2

```
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 ALIGN=center>
        <TR>
                <TD><IMG SRC=tree. gif WIDTH=27 HEIGHT=21 BORDER=0></TD>
                <TD><A HREF="javascript: void(0); "><IMG SRC="image1.gif" WIDTH=103 HEIGHT=21 BORDER=0></A></TD>
        </TR>
        <TR>
```

```
                    <TD><IMG SRC=tree. gif WIDTH=27 HEIGHT=21 BORDER=0></TD>
                    <TD><A HREF="javascript: void(O); "><IMG SRC="image2.gif" WIDTH=103 HEIGHT=21
BORDER=0></A></TD>
                </TR>
                <TR>
                    <TD><IMG SRC=tree. gif WIDTH=27 HEIGHT=21 BORDER=0></TD>
                    <TD><A HREF="javascript: void(0); "><IMG SRC="image3.gif" WIDTH=103 HEIGHT=21
B0RDER=0></A></TD>
                </TR>
                <TR>
                    <TD><IMG SRC=tree. gif WIDTH=27 HEIGHT=21 BORDER=0></TD>
                    <TD><A HREF="javascript: void(O); "><IMG SRC="image14.gif" WIDTH=103 HEIGHT=21
BORDER=0></A></TD>
                </TR>
        </TABLE>

        <table border="0" cellpadding="0" cellspacing="0" align="center" >
          <tr>
            <td><a href="javascript: void(0); "><img id="im01h" src="im01h.gif" style=" width:27; height:21 " alt=""
/></a></td>
          </tr>
          <tr>
            <td><a href="javascript: void(0); "><img id="im02h" src="im02h.gif" style=" width:27; height:21 " alt=""
/></a></td>
          </tr>
          <tr>
            <td><a href="javascript: void(0); "><img id="im03h" src="im03h.gif" style=" width:27; height:21 " alt=""
/></a></td>
          </tr>
          <tr>
            <td><a href="javascript: void(0); "><img id="im04h" src="im04h.gif" style=" width:27; height:21 " alt=""
/></a></td>
          </tr>
        </table>
```

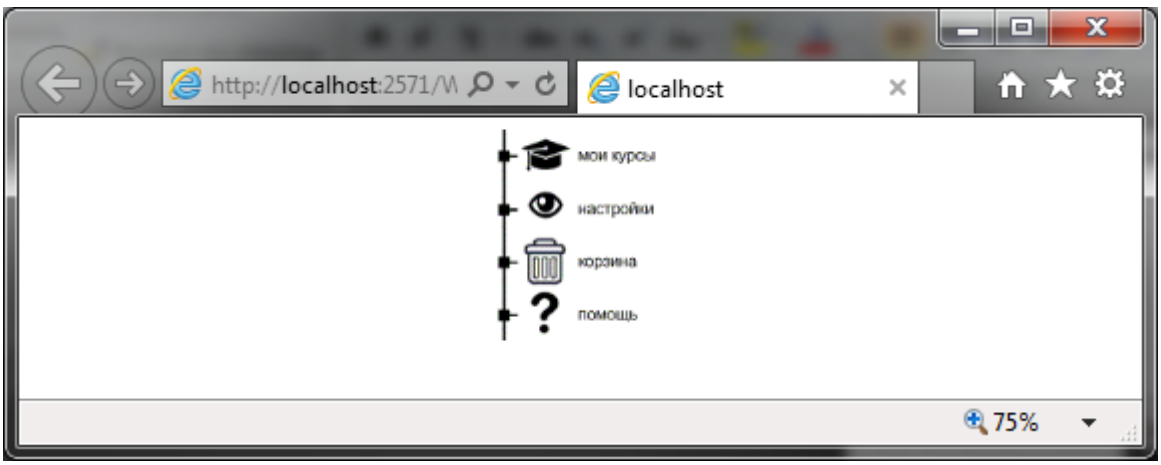


Рисунок 17. 4. – Многострочное меню

В данном случае все картинки удается сшить без пропусков и тем самым достичь непрерывности навигационного дерева. Пропуски устраняются путем применения атрибутов BORDER, CELLSPACING и CELLPADDING. Первый устраняет границы между ячейками, второй устанавливает расстояние между ячейками равным 0 пикселов, третий устанавливает отступ между границей ячейки и элементом, помещенным в нее, в 0 пикселов.

17.1.8. Графика и обработка событий

В данном разделе мы не будем рассматривать обработчики событий контейнера IMG. Мы остановимся на наиболее типичном способе комбинирования обработчиков событий и изменения *графических* образов. Собственно, не имело бы смысла применять нарезанную *графику,* если бы не возможность использования обработчиков событий для изменения отдельных частей изображения. Продолжая обсуждение примера с навигационным деревом, покажем его развитие с обработкой событий, вызванных наведением мыши на объект, и изменением картинок:

Листинг 16.3

```
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 ALIGN=center>
        <TR>
                <TD><IMG SRC=image.gif WIDTH=20 HEIGHT=20 BORDER=0></TD>
                <TD><IMG SRC=image1.gif WIDTH=20 HEIGHT=20 BORDER=0></TD>
        </TR>
        <TR>
                <TD><IMG SRC=image2.gif WIDTH=20 HEIGHT=20 BORDER=0></TD>
                <TD><A  HREF="javascript;  void(0);  "  onMouseover="document.manual.src='image3.gif';
return true; " onMouseout="document.manual.src='image4.gif'; return true; ">
                        <IMG SRC=image5.gif BORDER=0 WIDTH=20 HEIGHT=20x/A></TD>
        </TR>
        <TR>
                <TD><IMG SRC=image6.gif WIDTH=20 HEIGHT=20 BORDER=0></TD>
                <TD><A HREF="javascript; void(0); "
                        onMouseover="document.desk.src='image7.gif'; return true; "
                        onMouseover="document.desk.src='image8.gif'; return true; "
                        <IMG SRC=image9.gif BORDER=0 WIDTH=20 HEIGHT=20></A></TD>
        </TR>
</TABLE>

<table border="0" cellpadding="0" cellspacing="0" align="center" >
  <tr>
        <td><a  href="javascript:  void(0);  "  onclick="document.myImg.src='gr08.jpg';  return  true;  "><img
id="im01h" src="im01h.gif" style=" width:27; height:21 " alt="" /></a></td>
  </tr>
  <tr>
        <td><a  href="javascript:  void(0);  "  onclick="document.myImg.src='gr09.jpg';  return  true;  "><img
id="im02h" src="im02h.gif" style=" width:27; height:21 " alt="" /></a></td>
  </tr>
  <tr>
        <td><a  href="javascript:  void(0);  "  onclick="document.myImg.src='gr10.jpg';  return  true;  "><img
id="im03h" src="im03h.gif" style=" width:27; height:21 " alt="" /></a></td>
  </tr>
  <tr>
        <td><a  href="javascript:  void(0);  "  onclick="document.myImg.src='gr11.jpg';  return  true;  "><img
id="im04h" src="im04h.gif" style=" width:27; height:21 " alt="" /></a></td>
  </tr>
</table>
<br /><br />
<div align="center">
  <img name="myImg" src="" alt="" />
</div>
```

В данном примере при проходе курсор мышки через картинки меню последние изменяются. Этот эффект достигается за счет применения двух событий; onMouseover и onMouseout. По первому событию картинка меняется с позитива на негатив, по второму событию восстанавливается первоначальный вариант. Следует заметить, что события определены и контейнере якоря (A), а не в контейнере IMG. Это наиболее устойчивый с точки зрения совместимости браузеров вариант.

17.1.9. Вертикальные и горизонтальные меню

Практически все, что изложено в разделах «Графика и таблицы» и «Графика и обработка событий» касается вопросов построения одноуровневых меню. Поэтому в данном разделе мы постараемся привести более или менее реальные примеры таких меню. Графическое меню удобно тем, что автор может всегда достаточно точно расположить его компоненты на экране. Это, в свою очередь, позволяет другие элементы страницы точнее располагать относительно элементов меню.

Посмотрим на реализацию вертикального меню, построенного на основе *графических* блоков текста, как сейчас это принято делать:

Листинг 16.5

```
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" ALIGN="center">
        <TR>
                <TD>
                        <A  HREF="javascript:  void(0);  "onMouseover="document.evente1.src='corner.gif';
"onMouseout="document.evente1.src='clear.gif'; ">
```

```
                                        <IMG SRC="image1. gif" border="0"></A>
                                </TD>
                                <TD>
                                        <A HREF="javascript: void(0); "onMouseover="document.evente1.src='corner.gif';
"onMouseout="document.evente1.src='clear.gif';
                                        <IMG NAME="evente1" SRC="clear.gif" border="0"></A>
                                </TD>
                        </TR>
                        <TR>
                                <TD>
                                        <A HREF="javascript: void(0); "onMouseover="document.evente2.src='corner.gif';"
onMouseout="document.evente2.src='clear.gif';">
                                        <IMG SRC="image2. gif" border=”0"></A>
                                </TD>
                                <TD>
                                        <A       HREF="javascript:       void(0);       "onMouseover="document.evente2.src='
corner.gif';"onMouseout="document.evente2.src=' clear.gif';">
                                        <IMG NAME="evente2" SRC="clear.gif" border="0"></A>
                                </TD>
                        </TR>
                        <TR>
                                <TD>
                                        <A HREF="javascript: void(0); "onMouseover="document.evente3.src='corner.gif';"
onMouseout="document.evente3.src='clear.gif ';">
                                        <IMG SRC="image3.gif" border="0"></A>
                                </TD>
                                <TD>
                                        <A    HREF="javascript:    void(0);    "OnMouseover="document.evente3.src='corner
gif';"onMouseout="document.evente3.src='clear.gif ';">
                                        <IMG NAME="evente3" SRC="clear.gif" border="0"></A>
                                </TD>
                        </TR>
                        <TR>
                                <TD>
                                        <A HREF="javascript: void(0); "onMouseover="document.evente4.src='corner.gif';"
onMouseout="document.evente4.src='clear.gif';">
                                        <IMG SRC="image4.gif" border="0"></A>
                                </TD>
                                <TD>
                                        <A HREF="javascript: void(0); "onMouseover="document.evente4.src='corner.gif';"
onMouseout="document.evente4.src='clear.gif'; ">
                                        <IMG NAME="evente4" SRC="clear.gif" border="0"> </A>
                                </TD>
                        </TR>
                </TABLE>

                <form>
                <table border="0" cellpadding="0" cellspacing="0" align="center" >
                        <tr>
                                <td><a href="javascript: void(0); " onmousedown="document.myImg.src='gr01.jpg'; return true; "
onmousemove="document.im01g.src='im0102g.png';                  return                  true;                  "
onmouseout="document.im01g.src='im0101g.png'; return true; " ><img name="im01g" src="im0101g.png" style="
width:15; height:8 " title="Мои курсы" alt="" /></a></td>
                        </tr>
                        <tr>
                                <td><a href="javascript: void(0); " onmousedown="document.myImg.src='gr03.jpg'; return true; "
onmousemove="document.im02g.src='im0202g.png';                  return                  true;                  "
onmouseout="document.im02g.src='im0201g.png'; return true; " ><img name="im02g" src="im0201g.png" style="
width:15; height:8 " title="Настройки" alt="" /></a></td>
                        </tr>
                        <tr>
                                <td><a href="javascript: void(0); " onmousedown="document.myImg.src='gr05.jpg'; return true; "
onmousemove="document.im03g.src='im0302g.png';                  return                  true;                  "
onmouseout="document.im03g.src='im0301g.png'; return true; " ><img name="im03g" src="im0301g.png" style="
width:15; height:8 " title="Корзина" alt="" /></a></td>
                        </tr>
```

```
        <tr>
            <td><a href="javascript: void(0); " onmousedown="document.myImg.src='gr07.jpg'; return true; "
onmousemove="document.im04g.src='im0402g.png'; return true; "
onmouseout="document.im04g.src='im0401g.png'; return true; " ><img name="im04g" src="im0401g.png" style="
width:15; height:8 " title="Помощь" alt="" /></a></td>
        </tr>
    </table>
    <br /><br />
    <div align="center">
        <img name="myImg" src="" alt="" />
    </div>
    </form>
```

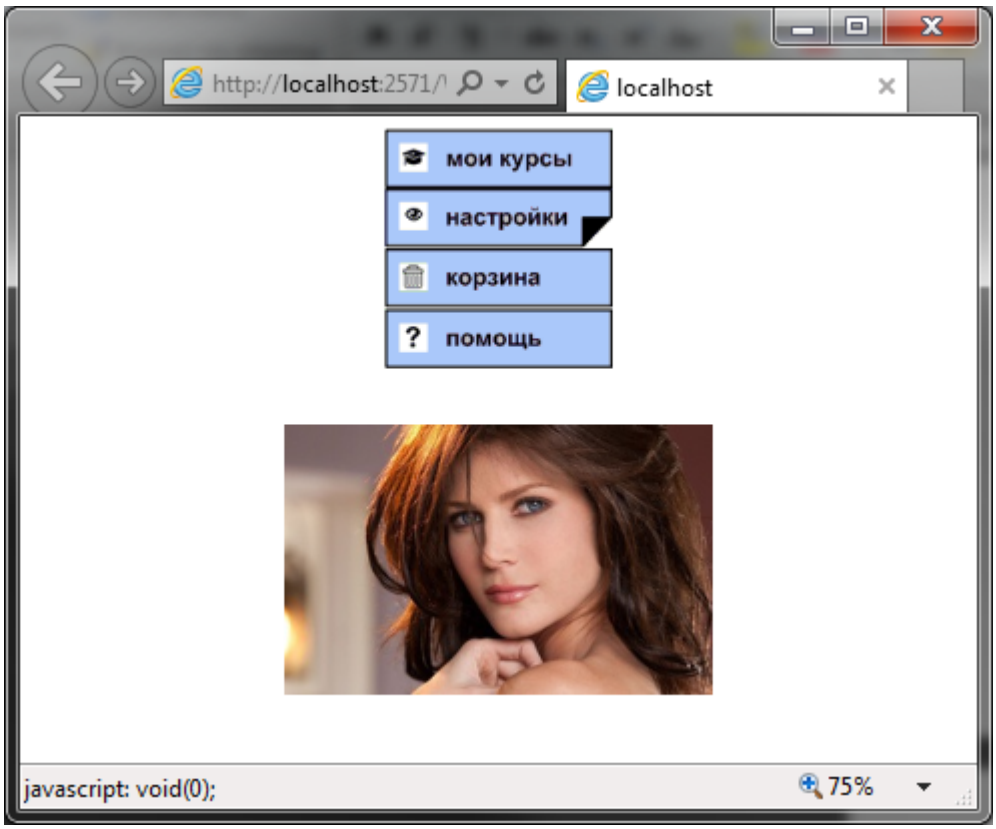


Рисунок 17. 5. – Реализация вертикального одноуровневого меню

При движении мыши у соответствующего компонента, попавшего в фокус мыши, «отгибается уголок». В данном случае «уголок» — это самостоятельная картинка. Все уголки реализованы в правой колонке таблицы. Для того чтобы гипертекстовая ссылка срабатывала по обеим картинкам (тексту и «уголку»), применяются одинаковые контейнеры A, охватывающие *графические* образы. В этом решении есть один недостаток: при переходе от текста к «уголку» последний «подмигивает». Картинки можно разместить и в одной ячейке таблицы, но тогда нужно задать ее ширину, иначе при изменении размеров окна браузера картинки могут «съехать». Чтобы убрать «подмигивание», необходимо сделать полноценные картинки замены.

«Подмигивание» происходит при переходе с одного элемента разметки контейнера на другой. При этом заново определяются свойства отображения элемента.

17.1.10. Вложенные меню

При обсуждении программирования форм отмечено, что в HTML нет стандартного способа реализации вложенных меню. Тем не менее за счет *графики* можно создать их подобие. При этом следует понимать, что место, на которое ложится *графика,* нельзя заполнить текстом.

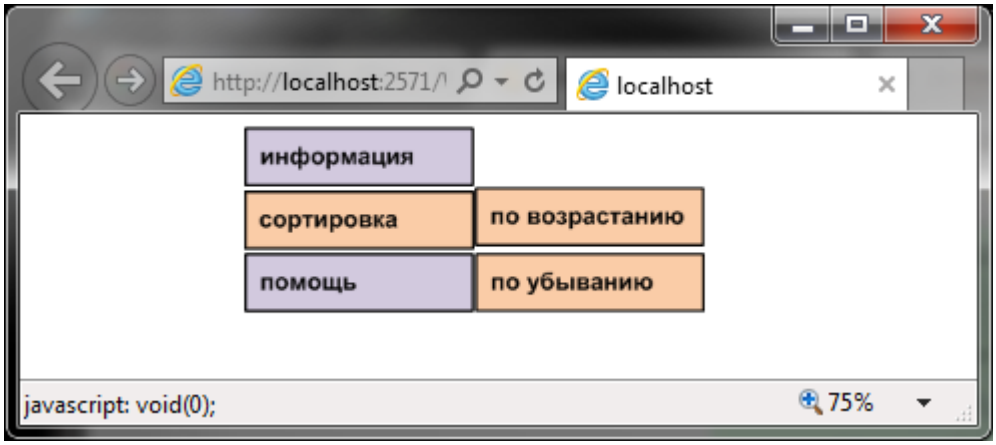


Рис. 17. 6. – Вложенное HTML-меню

В этом примере вложенное меню расположено справа от основного Эффект вложенности достигается за счет изменения цвета. Подчиненность меню можно подчеркнуть изменением его положения относительно основного меню.

При использовании слоев можно создать настоящее выпадающее меню.

Простое меню с использованием JavaScript.

При создании сайта лучше всего использовать меню, которое будет генерироваться на JavaScript. И при добавлении нового пункта не придётся править все страницы сайта. Код меню будет находиться в отдельном файле, в нём будут прописаны названия элементов меню и ссылки на страницы сайта.

Код файла my_menu.js:

```
var menu = new Array();
var link = new Array();

menu[0] = "Главная"; // Название страницы
link[0] = "main.htm"; // Имя страницы
menu[1] = "Новости";
link[1] = "news.htm";
menu[2] = "Скачать";
link[2] = "download.htm";

document.write('<div style="background-color:#cccccc;text-align:right;padding:5px;">');
/* Задаём цвет фона и выравнивание */
for (var i = 0; i < menu.length; i++) { // Выводим все пункты меню
   document.write('<a href="' + link[i] + '">' + menu[i] + '</a> ');
   if (i + 1 < menu.length) document.write('| ');
   /* В качестве разделителя используем вертикальную черту */
}
document.write('</div>')
```

HTMLPage3.htm

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title></title>
</head>
<body>
<!-- Этот код должен быть на каждой странице.
        Он отвечает за вывод меню. -->
<script language="javascript"
   type="text/javascript"
   src="my_menu.js">
</script>

<h4>Основная часть страницы</h4>
</body>
</html>
```

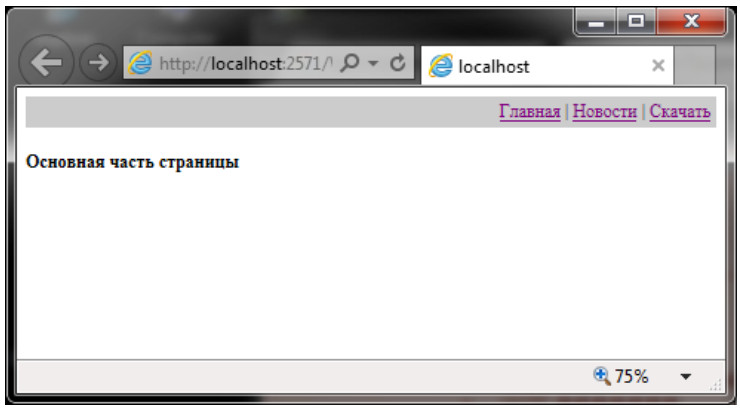


Рисунок 17.7. – Простое меню с использованием JavaScript

17.1.11. Выпадающее меню на HTML / CSS без использования JavaScript

Очень часто необходимо сделать на сайте выпадающее при наведении меню. Большинство веб-девелоперов используют для этого JavaScript / Jquery. Это конечно хорошо, но такого рода задачи легко решаются с помощью HTML / CSS.

Для построения меню был использован список с классом menu. Подменю сделаны списками с классом submenu.

Здесь используется минимальное оформление нашего меню стилями.

Для класса submenu было установлено свойство 'display: none'. Это позволило нам спрятать наше выпадающее меню.

В CSS (в конце) добавлен код. Он позволяет показывать подменю при наведении.

Вот простой пример HTML-кода меню:

HTMLCSSMenu.htm

```
<html xmlns="http://www.w3.org/1999/xhtml">
<link type="text/css" rel="stylesheet" href="MyStyleMeny.css">
<head>
   <title></title>
</head>
<body>
<ul class="menu">
   <li><a href=#>Menu 1</a>
      <ul class="submenu">
         <li><a href=#>Sudmenu 1</a></li>
         <li><a href=#>Sudmenu 1</a></li>
         <li><a href=#>Sudmenu 1</a></li>
      </ul>
   </li>
   <li><a href=#>Menu 2</a>
      <ul class="submenu">
         <li><a href=#>Sudmenu 2</a></li>
         <li><a href=#>Sudmenu 2</a></li>
         <li><a href=#>Sudmenu 2</a></li>
      </ul>
   </li>
   <li><a href=#>Menu 3</a>
      <ul class="submenu">
         <li><a href=#>Sudmenu 3</a></li>
         <li><a href=#>Sudmenu 3</a></li>
         <li><a href=#>Sudmenu 3</a></li>
      </ul>
   </li>
   <li><a href=#>Menu 4</a>
      <ul class="submenu">
         <li><a href=#>Sudmenu 4</a></li>
         <li><a href=#>Sudmenu 4</a></li>
         <li><a href=#>Sudmenu 4</a></li>
      </ul>
   </li>
   <li><a href=#>Menu 5</a>
      <ul class="submenu">
```

```
            <li><a href=#>Sudmenu 5</a></li>
            <li><a href=#>Sudmenu 5</a></li>
            <li><a href=#>Sudmenu 5</a></li>
        </ul>
    </li>
</ul></body>
</html>
```

MyStyleMeny.css

```
body {
    font: 14px 'Verdana';
    margin: 0;
    padding: 0;
}
ul {
    display: block;
    margin: 0;
    padding: 0;
    list-style: none;
}
ul:after {
    display: block;
    content: ' ';
    clear: both;
    float: none;
}
ul.menu > li {
    float: left;
    position: relative;
}
ul.menu > li > a {
    display: block;
    padding: 10px;
    color: white;
    background-color: red;
    text-decoration: none;
}
ul.menu > li > a:hover {
    background-color: black;
}
ul.submenu {
    display: none;
    position: absolute;
    width: 120px;
    top: 37px;
    left: 0;
    background-color: white;
    border: 1px solid red;
}
ul.submenu > li {
    display: block;
}
ul.submenu > li > a {
    display: block;
    padding: 10px;
    color: white;
    background-color: red;
    text-decoration: none;
}
ul.submenu > li > a:hover {
    text-decoration: underline;
}

ul.menu > li:hover > ul.submenu {
    display: block;
}
```

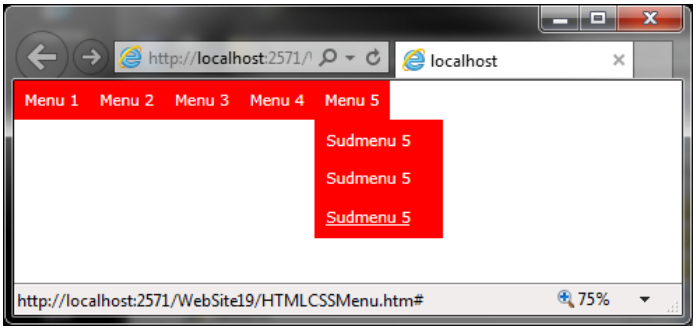


Рисунок 17.8. – Выпадающее меню, созданное с использованием CSS/HTML

Часто в верстке нужно сделать выпадающий список, обычно для главного меню. Как решается такой вопрос? Конечно, с помощью javascript. Тем не менее, можно значительно упростить создание выпадающего меню.

Мы создадим список со ссылками, который будет отображаться при наведении курсора. И я думаю, что многие так уже делают, поэтому просто покажу решение, которым пользуюсь я.

Сначала создадим список с основными пунктами и добавим по одному списку внутрь каждого из пунктов. Вложенные списки будут отображаться только при наведении курсора.

Чтобы все работало, нам потребуются стили.

Как видите, решение очень простое и полностью кроссбраузерное.

17.1.12. Создание меню при помощи CSS и JavaScript

Здесь мы рассмотрим примеры создания наиболее распространенных типов меню для web-сайтов. Для этого мы будем использовать семантику HTML и возможности CSS и JavaScript (DHTML).

17.1.12.1. Простое горизонтальное меню

Цель: Простое горизонтальное меню, с очень коротким и простым кодом.

Возможности: Простая, удобная и красивая навигация на сайте с небольшим количеством разделов.

Принцип работы: Берем маркированный список со ссылками, выстраиваем его в линию и делаем выделение той ссылки, на которую наводят курсор.

HTMLPage4.htm

```
<html xmlns="http://www.w3.org/1999/xhtml">

<link type="text/css" rel="stylesheet"
          href="MyStyleMeny02.css">
<head>
   <title></title>
</head>
<body>
<!-- Этот код должен быть на каждой странице.
         Он отвечает за вывод меню.
<script language="javascript"
   type="text/javascript"
   src="my_menu02.js">
</script>
-->

<div>

<!--маркированный список со ссылками внутри-->
<ul>
   <li><a href="#">Home</a></li>
   <li><a href="#">About</a></li>
   <li><a href="#">Works</a></li>
   <li><a href="#">Links</a></li>
   <li><a href="#">Contacts</a></li>
</ul>

</div>
</body>
```

```
</html>
```

MyStyleMeny02.css

```
/*маркированный список со ссылками внутри*/
/*выстроить его в линию и заодно убрать маркер*/
ul li{
        list-style-type:none;
        display:inline
        }
/*Поскольку мы не можем использовать блочные свойства
стилей для inline элементов, сделаем из тега a блочный элемент,
а чтобы они выстроились в одну строку,
добавим свойство float:left*/
li a{
        display:block;
        float:left
        }
/*описываем стили для текста ссылок*/
li  a{
        font-family:Tahoma,Helvetica,sans-serif;
        text-decoration:none;
        color:#fff
        }
/*добавляем внутренние отступы и фон для ссылки*/
li a{
        display:block;
        float:left;
        padding:4px 10px;
        background:#193D6A
        }
/*описываем псевдокласс hover*/
li a:hover{
        background:#408BE8;
        padding:7px 14px;
        position:relative;
        top:-3px
        }
```

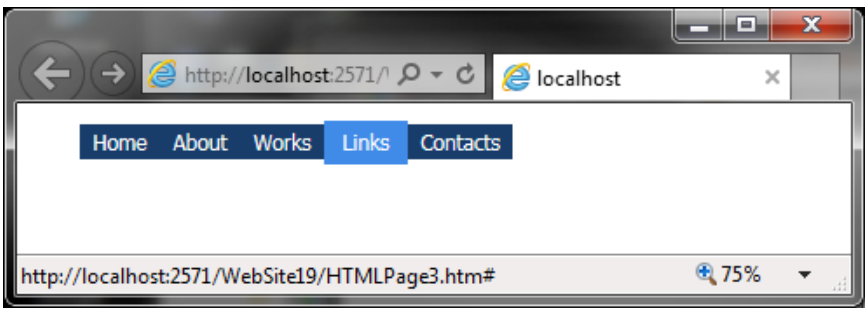


Рисунок 17.9. – Простое горизонтальное меню

Не забудьте следующему блочному элементу после меню прописать в стилях свойство clear:both, чтобы прекратить обтекание.

17.1.12.2. Вертикальное выпадающее меню

Описание: Это вертикальное выпадающее меню, которое позволяет разместить большое количество ссылок на ограниченной площади. Идею этого меню я позаимствовал у господина Nick Rigby, доработав меню для своих целей.

Возможности: Простая, удобная и красивая навигация на сайте с большим количеством разделов.

Принцип работы: Идея этого меню проста, как и все гениальное, и основана на псевдоклассах CSS. Используя псевдокласс hover, мы меняем свойство выпадающего блока с display:none на display:block. Т. е., при наведении мышки невидимый контейнер со ссылками становится видимым.

HTMLPage3.htm

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<link type="text/css" rel="stylesheet"
          href="MyStyleMeny02.css">
<head>
   <title></title>
</head>
<body>
<!-- -->
<script language="javascript"
   type="text/javascript"
   src="my_menu02.js">
</script>

<!--тело меню-->
<div id="menu_body">
  <ul id="ul1">
       <li><a href="#">menu 1</a>
           <ul>
              <li><a href="#">sub menu 1</a></li>
              <li><a href="#">sub menu 2</a></li>
              <li><a href="#">sub menu 3</a></li>
              <li><a href="#">sub menu 4</a></li>
           </ul>
       </li>

.........

  </ul>
</div>
</body>
</html>
```

MyStyleMeny02.css

```
/*уберем подменю*/
#menu_body li ul{display:none}

/*Используя псевдокласс hover,
сделаем подменю видимым при наведении мыши*/
#menu_body li:hover ul, #menu_body li.over ul{
          display: block
          }

#menu_body{
          background:#81A192;
          width:200px
          }

#menu_body ul li{
          list-style-type:none;
          border-bottom:1px solid #fff;
          margin-left:-40px;
          padding-left:7px
          }

#menu_body ul li a{
          color:#fff;
          font-family:verdana,arial,sans-serif;
          text-decoration:none
          }

#menu_body ul li ul li{
          border:0;
          list-style-type:square;
          color:#fff;
          list-style-position:inside
          }
```

```
#menu_body ul li ul{
        border-top:1px solid #fff;
        margin-left:-7px;
        padding-left:50px
        }
```

my_menu02.js

```
/*Поскольку Internet Explorer считает,
что псевдоклассом hover обладает только тег a,
добавим небольшой код на JavaScript, который это исправит*/
startList = function() {
   if (document.all && document.getElementById) {
      navRoot = document.getElementById("ul1");
      for (i = 0; i < navRoot.childNodes.length; i++) {
         node = navRoot.childNodes[i];
         if (node.nodeName == "LI") {
            node.onmouseover = function() {
               this.className += " over";
            }
            node.onmouseout = function() {
               this.className = this.className.replace(" over", "");
            }
         }
      }
   }
}
window.onload = startList;
```

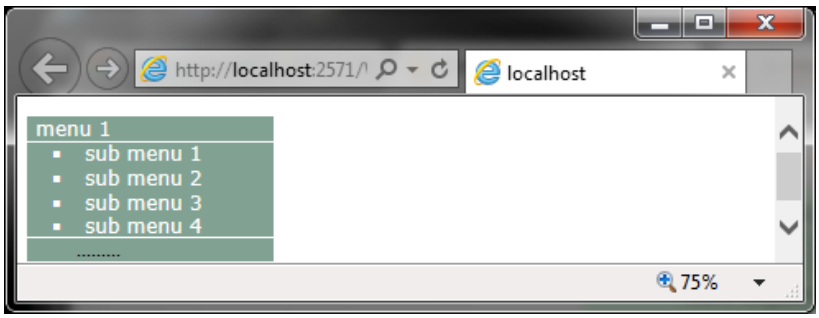


Рисунок 17.10. – Вертикальное выпадающее меню

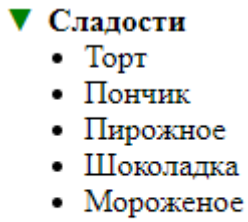
17.2. Контрольные вопросы

1. На какие группы делятся объекты класса Image?
2. Какие способы применения техники нарезки картинок на составные части для организации навигационных компонентов Вы можете назвать?
3. Какие способы обращения к графическому объекту вы знаете?
4. За что отвечают свойства src и lowsrc?
5. Что такое мультипликация? Какой метод используется для ее реализации?
6. Как избежать переноса нарезанной графики на следующую страницу?

17.3. Задания

1. Создайте массив картинок, в который перед отображением будет перекачиваться графика, используйте временную задержку и функцию eval(). (R17001)
2. Создайте анимацию, в которую встроен цикл изменения картинки с конечным количеством подмен, реализуйте для неё запуск и остановку движения по требованию пользователя. (R17002)
3. Создайте горизонтальное выпадающее меню, используя семантику HTML и возможности CSS и JavaScript (DHTML). (R17003)

▶ **Сладости**



4. Создайте выпадающее меню(рис.17.11). (R17004)

Рис. 17.11