Динамические стили и анимация

В соответствии со способами добавления стилей различают и способы динамического управления стилями. Использование таблиц стилей открывает широкие возможности для управления внешним видом и содержанием документа.

Управление стилями CSS

Напомним, что большинство свойств в таблицах стилей образованы ключевыми словами, разделенными дефисом (`font-size`, `text-align` и т.д.). Однако в языковых конструкциях JavaScript дефис может интерпретироваться как оператор. Поэтому в именах свойств объектов дефис не употребляется, а для согласованности их с именами свойств CSS применяется следующее правило. Первое ключевое слово свойства CSS набирается в нижнем регистре, а все последующие ключевые слова начинаются с прописной буквы и записываются без дефисов и пробелов. Например, если нужно обратиться к свойству `letter-spacing`, указывают имя `letterSpacing`, а полная запись этого свойства для объекта `element1` будет выглядеть как

```
element1.style.letterSpacing
```

При работе со свойствами стилей следует помнить, что все значения должны быть указаны в виде строк. В таблице стилей или атрибуте `style` можно написать:

```
position: absolute; font-family: sans-serif; background-
color: #ffffff;
```

Чтобы сделать то же самое для элемента e в JavaScript, необходимо поместить все эти значения в кавычки:

```
e.style.position = "absolute";
e.style.fontFamily = "sans-serif";
e.style.backgroundColor = "#ffffff";
```

Обратите внимание, что точки с запятыми остаются вне строк. Это обычные точки с запятой, употребляемые в синтаксисе языка JavaScript. Точки с запятой, используемые в таблицах CSS-стилей, не нужны в строковых значениях, устанавливаемых с помощью JavaScript.

Кроме того, помните, что во всех свойствах позиционирования должны быть указаны единицы измерения. Следовательно, нельзя устанавливать свойство `left` подобным образом:

```
e.style.left = 300; // Неправильно: это число, а не строка
e.style.left = "300"; // Неправильно: отсутствуют единицы
измерения
```

Единицы измерения обязательны при установке свойств стиля в JavaScript – так же, как при установке атрибутов стиля в таблицах стилей. Далее приводится правильный способ установки значения свойства `left` элемента e равным 300 пикселов:

```
e.style.left = "300px";
```

Чтобы установить свойство `left` равным вычисляемому значению, обязательно добавьте единицы измерения в конце вычислений:

```
e.style.left = (x0 + left_margin + left_border +
left_padding) + "px";
```

Как побочный эффект добавления единиц измерения, добавление строки преобразует вычисленное значение из числа в строку.

CSS-классы

Альтернативой использованию отдельных CSS-стилей через свойство `style` является применение значения атрибута `class` через свойство `className` любого HTML-элемента. Динамическое изменение класса элемента может приводить к существенным изменениям стилей, применяемых к этому элементу, при этом, конечно же, предполагается, что используемый класс соответствующим образом определен в таблице стилей.

Главное, что необходимо помнить об HTML-атрибуте `class` и соответствующем ему свойстве `className`, – они могут содержать более одного класса. Вообще, при работе со свойством `className` не принято просто устанавливать или читать его значение, как если бы это свойство содержало единственное имя класса. Вместо этого необходимо пользоваться функцией, позволяющей проверить принадлежность элемента классу, а также функциями добавления классов в свойство `className` элемента и их удаления из свойства `className`.

DHTML-анимация

Одной из наиболее мощных DHTML-технологий, которую можно реализовать с помощью JavaScript и CSS, является анимация. В DHTML-анимации нет ничего особенного – надо лишь периодически изменять одно или несколько свойств стиля одного или нескольких элементов. Чтобы, например, передвинуть изображение влево, надо постепенно увеличивать значение свойства `style.left` этого изображения, пока последнее не займет требуемое положение. Можно также постепенно изменять свойство `style.clip` для «открытия» изображения пиксел за пикселом.

Рассмотрим пример, который содержит простой HTML-файл, определяющий анимируемый элемент `div`, и короткий сценарий, изменяющий цвет фона элемента каждые 500 миллисекунд. Примечательно, что изменение цвета выполняется присваиванием значения свойства CSS-стиля. Анимация возникает за счет того, что изменение цвета выполняется периодически с помощью функции `setInterval()` объекта `window`. (Никакая DHTML-анимация не обходится без метода `setInterval()` или `setTimeout()`) И наконец, обратите внимание на использование оператора деления по модулю (получение остатка) % для перебора цветов.

```
<!-- Это анимируемый элемент div -->
<div id="urgent"><h1>Внимание!</h1>Веб-сервер
атакован!</div>
<script>
  var e = document.getElementById("urgent"); // Получаем
объект Element
  e.style.border = "solid black 5px"; // Рамка
  e.style.padding = "50px"; // И отступ
  var colors = ["white", "yellow", "orange", "red"] //
Перебираемые цвета
  var nextColor = 0; // Текущая позиция перебора
  // Вызывать следующую функцию с интервалом 500 миллисекунд
  // для изменения цвета рамки.
  setInterval(function() {
  e.style.borderColor=colors[nextColor++%colors.length];
 }, 500);
</script>
```

Данный пример реализует очень простую анимацию. На практике анимация с использованием CSS-стилей обычно подразумевает одновременную модификацию нескольких свойств стилей (таких как `top`, `left` и `clip`). Сложную анимацию с помощью технологии, показанной в примере выше, создать трудно. Кроме того, чтобы не надоедать пользователю, анимация должна выполняться в течение короткого периода времени и затем останавливаться, чего нельзя сказать об анимации из рассмотренного примера.

Текст с меняющимся цветом

Составим документ, в котором реализован эффект меняющегося цвета текста. Для текста определяется объект стиля, цвет текста задается значением свойства `color` этого объекта. Периодическое изменение значения color осуществляется с помощью функции таймера `setInterval()`. Код документа, содержащего такой текст, выглядит следующим образом:

```
<html>
  <head>
    <title>Текст переменного цвета</title>
    <style type="text/css">
    #changeColor{font-size:14pt; color:gray}
    </style>
    <script language="JavaScript">
    //Функция, изменяющая цвет объекта
    function flashColor(){
      //Определение объекта стиля
      var elem=document.all.changeColor.style;
      //Изменение свойства цвета
      if (elem.color=="gray")
      elem.color="red"
      else
      elem.color="gray"
    }
    //Периодический вызов функции flashColor()
```

```
    setInterval("flashColor()",500)
    </script>
  </head>
  <body>
    <span id="changeColor">
      Текст меняющегося цвета
    </span>
  </body>
</html>
```

На данной странице отображаются слова «Текст меняющегося цвета», цвет которых периодически становится то серым, то красным. Период изменения цвета, выраженный в миллисекундах, определяется значением второго аргумента функции `setInterval()`.

Мигающий текст

Аналогично можно составить код сценария, который реализует мигание текста. Для этого вместо свойства color нужно использовать свойство `visibility` со значениями `visible` и `hidden`.

Замена изображения

Задача замены изображения решается с помощью обработчиков событий `onmouseover` и `onmouseout`. Так, при наведении курсора мыши на одну картинку, первоначально присутствовавшую на странице, она заменяется на другую картинку. После того, как указатель мыши убран с картинки, прежнее изображение возвращается. При этом код записывается в следующем виде:

```
<img src="picture1.gif"
onmouseover="this.src='picture2.gif'"
onmouseout="this.src='picture1.gif'">
```

Еще один способ замены изображения – наведение указателя мыши на гиперссылку. Предположим, на странице имеется элемент `img`, который выводит изображение `start.gif`. При наведении указателя мыши на ссылку «Фото» изображение `start.gif` должно заменяться на `fotosmall.gif` (маленькая фотография). Если кликнуть мышью по ссылке, то в отдельное окно должен быть загружен файл с изображения `foto.jpg` (большая фотография). Код, реализующий такую замену, приводится ниже:

```
<a href="foto.jpg"
onmouseover="document.mypic.src='forosmall.gif';"
onmouseout="document.mypic.src='start.gif';">
Фото</a>
<br>
<img name="mypic" border=0 src="start.gif">
```

Создание движущихся объектов

Рассмотрим примеры документов со сценариями, которые реализуют движение текста.

Движение текста слева направо

```
<html>
<head>
<title>Движение теста слева направо</title>
<script language="JavaScript">
function moveTxt()
{
if (anil.style.pixelLeft < 500)
{
anil.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Движение текста по диагонали

```
<html>
<head>
<title>Движение теста по диагонали</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelTop <500)
{
anim.style.pixelTop +=2;
anim.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Динамическое управление слоями

Сценарии JavaScript позволяют динамически управлять параметрами установленных слоев, поэтому можно получить такие эффекты, как скрытие и отображение слоя, изменение порядка отображения, перемещение слоя в окне браузера. Все эти эффекты достигаются с помощью изменения соответствующих стилевых параметров установленных слоев.

Для обращения к слоям из сценариев JavaScript, удобнее всего каждому слою дать собственное имя при помощи параметра `id`. Например:

```
<div id="div1">
...
</div>
```

Для того чтобы скрыть отображение слоя `div1`, можно использовать

```
div1.style.visibility='hidden';
```

Для повторного отображения слоя следует выполнить следующее присвоение:

```
div1.style.visibility='visible';
```

Рассмотрим пример динамической смены слоев. В данном примере для отображения некоторого слоя следует нажать на соответствующую ссылку. Эту идею можно применить и для организации выпадающих меню.

```
<html>
<head>
<style type="text/css">
div {
position: absolute; top: 20;left: 160;visibility: hidden;
}
</style>
<script language="JavaScript">
function show_d(d)
{
div1.style.visibility='hidden';
div2.style.visibility='hidden';
div3.style.visibility='hidden';
div4.style.visibility='hidden';
div5.style.visibility='hidden';
d.style.visibility='visible';
}
</script>
</head>
<body>
<a href="javascript:void(0)" onClick="show_d(div1);">
показать слой 1
</a><br>
<a href="javascript:void(0)" onClick="show_d(div2);">
показать слой 2
</a><br>
```

```
<a href="javascript:void(0)" onClick="show_d(div3);">
показать слой 3
</a><br>
<a href="javascript:void(0)" onClick="show_d(div4);">
показать слой 4
</a><br>
<a href="javascript:void(0)" onClick="show_d(div5);">
показать слой 5
</a><br>
<div id="div1">
<h3>Слой номер один</h3>
Некоторый текст, расположенный на слое. Его можно скрыть и показать. Текст может содержать несколько строк.
</div>
<div id="div2">
<h3>Слой номер два</h3>
Содержит свой текст. Если показывается, то текст на других слоях не виден.
</div>
<div id="div3">
<h3>Слой номер три</h3>
Тоже текст. При работе со слоями надо следить, чтобы текст одного слоя не "выглядывал" из-под другого слоя при самых различных размерах окна браузера и используемых шрифтах.
</div>
<div id="div4">
<h3>Слой номер четыре</h3>
Здесь нет текста.
</div>
<div id="div5">
<h3>Слой номер пять</h3>
И тут тем более нет.
</div>
</body>
</html>
```