

ВВЕДЕНИЕ В JAVASCRIPT ДЛЯ МАГА

© 1996, 1997 Стефан Кох (Stefan Koch)

Часть 10: Слои II

Мы уже обсудили основные понятия новой технологии слоев. В этой же части будут рассмотрены следующие темы:

- Вырезка из слоя
- Вложенные слои
- Различные эффекты с прозрачными слоями

Вырезка из слоя

Можно постулировать, что какая-то (прямоугольная) часть слоя будет нам видима. Все же, что лежит за ее пределами, показано на экране не будет. Такой прием называется вырезанием. Например, в разметке HTML можно задать следующую функцию вырезания:

```
<ilayer left=0 top=0 clip="20,50,110,120">  
  
</ilayer>
```

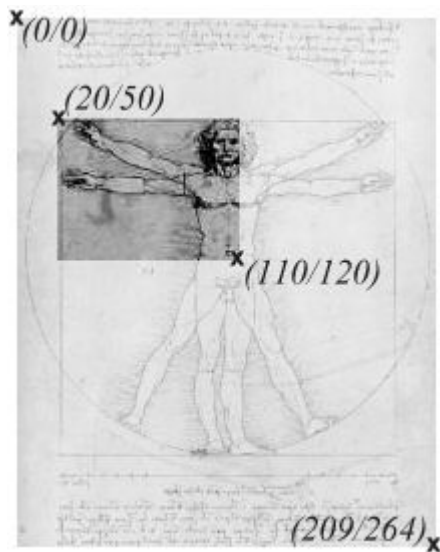
(Здесь я приписал параметры *left=0* и *top=0*, поскольку в противном случае, если этого не сделать, то с моей версией Netscape (PR3 on WinNT) возникают некоторые проблемы)

Хотя само изображение и имеет размеры 209x264 пикселей, мы можем видеть лишь его малую часть:



Данный фрагмент изображения имеет размер 90x70 (пикселей). Первые два значения, указанные в атрибуте *clip* (атрибуте HTML-тэга `<layer>` или `<ilayer>`), указывают

верхний левый угол вырезаемой части. Следующие два значения указывают нижний правый угол. Сказанное можно проиллюстрировать следующим рисунком:



Еще более интересных результатов можно добиться, управляя вырезанной частью с помощью языка JavaScript. Точнее, Вы можете изменять значения свойств `clip.left`, `clip.top`, `clip.right` и `clip.bottom` объекта `Layer`. Достаточно всего лишь занести в одно из этих свойств новое значение, как фрагмент тут же будет кадрирован соответствующим образом. В следующем примере параметры вырезанной части изображения меняются динамически, и в результате у пользователя создается впечатление, будто изображение медленно "растет":

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

Код соответствующего скрипта:

```
<html>
<head>

<script language="JavaScript">
<!-- hide

var middleX, middleY, pos;

function start() {
    // получить размер изображения
    var width= document.layers["imgLayer"].document.davinci.width;
    var height= document.layers["imgLayer"].document.davinci.height;

    // определить, какой пиксел находится в центре изображения
    middleX= Math.round(width/2);
    middleY= Math.round(height/2);

    // начальная позиция
```

```

    pos= 0;

    // запуск!
    show();
}

function show() {

    // увеличить размер вырезаемой области
    pos+= 2; // величина шага
    document.layers["imgLayer"].clip.left= middleX- pos;
    document.layers["imgLayer"].clip.top= middleY- pos;
    document.layers["imgLayer"].clip.right= middleX+ pos;
    document.layers["imgLayer"].clip.bottom= middleY+ pos;

    // проверить, не высвечено ли все изображение
    if (!(pos > middleX) && (pos > middleY))
        setTimeout("show()", 20);

}

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

<body>

<ilayer name="imgLayer" clip="0,0,0,0">

</ilayer>

<form>
<input type=button value="Start" onClick="start()">
</form>

</body>
</html>

```

Кнопка, представленная в разделе `<body>`, вызывает функцию `start()`. Сначала мы должны определить точку, с которой нам следует начать работу - фактически это будет некий пиксел в центре нашего изображения. Значения координат *x* и *y* этого пиксела мы помещаем в переменные `middleX` и `middleY`. После этого мы вызываем функцию `show()`, которая задает размеры вырезаемой части изображения в зависимости от значений переменных `middleX`, `middleY` и параметра `pos`. При этом значение переменной `pos` автоматически увеличивается при каждом вызове функции `show()`. То есть размер вырезаемой части изображения с каждым разом становится все больше и больше. В самом конце процедуры `show()` мы устанавливаем таймер с помощью вызова `setTimeout()` - и благодаря этому функция `show()` вызывается вновь и вновь. И этот процесс остановится только тогда, когда изображение будет показано целиком. Заметим, что размер изображения мы получаем в самом начале функции `start()`:

```
var width= document.layers["imgLayer"].document.davinci.width;  
var height= document.layers["imgLayer"].document.davinci.height;
```

С помощью конструкции `document.layers["imgLayer"]` мы можем обратиться к слою с именем `imgLayer`. Однако почему после `document.layers["imgLayer"]` мы ставим `document`? Дело в том, что каждый слой имеет свою собственную HTML-страницу - то есть, **каждый слой получает имеет объект document**. Чтобы получить доступ к изображению внутри слоя `imgLayer`, нам необходимо получить доступ к этому объекту `document`. В приведенном выше примере такое изображение носило название `davinci`. Все остальное поле листа должно быть чистым.

Вложенные слои

Как мы уже видели, слой может содержать несколько различных объектов. Он могут даже включать в себя другие слои. Конечно, может возникнуть вопрос, для чего это нужно. На самом деле есть несколько причин, чтобы пользоваться вложенными слоями. Рассмотрим несколько примеров, демонстрирующих применение вложенных слоев.

В первом примере используется слой (называемый `parentLayer`), в который вложено еще два других слоя (`layer1` и `layer2`).

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

После открытия мы видим три кнопки. Эти кнопки могут запускать и останавливать движение слоев. Также можно видеть, что перемещение слоя `parentLayer` сопровождается перемещением и двух других слоев, тогда как перемещение слоя `layer1` (или `layer2`) ни на что другое не влияет. Этот пример демонстрирует возможность объединения группы объектов с помощью механизма вложенных слоев.

Рассмотрим теперь исходный код скрипта:

```
<html>  
<head>  
  
<script language="JavaScript">  
<!-- hide  
  
// начальна\я позици\я  
var pos0= 0;  
var pos1= -10;  
var pos2= -10;  
  
// движение?  
var move0= true;  
var move1= false;  
var move2= false;  
  
// направление?  
var dir0= false;  
var dir1= false;
```

```

var dir2= true;

function startStop(which) {
  if (which == 0) move0= !move0;
  if (which == 1) move1= !move1;
  if (which == 2) move2= !move2;
}

function move() {

  if (move0) {
    // move parentLayer
    if (dir0) pos0--
    else pos0++;

    if (pos0 < -100) dir0= false;

    if (pos0 > 100) dir0= true;

    document.layers["parentLayer"].left= 100 + pos0;
  }

  if (move1) {
    // перемещение parentLayer
    if (dir1) pos1--
    else pos1++;

    if (pos1 < -20) dir1= false;

    if (pos1 > 20) dir1= true;

    document.layers["parentLayer"].layers["layer1"].top= 10 + pos1;
  }

  if (move2) {
    // перемещение parentLayer
    if (dir2) pos2--
    else pos2++;

    if (pos2 < -20) dir2= false;

    if (pos2 > 20) dir2= true;

    document.layers["parentLayer"].layers["layer2"].top= 10 + pos2;
  }

}

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

```

```

<body onLoad="setInterval('move()', 20)">

<ilayer name=parentLayer left=100 top=0>
  <layer name=layer1 z-index=10 left=0 top=-10>
    Это первый слой
  </layer>

  <layer name=layer2 z-index=20 left=200 top=-10>
    Это второй слой
  </layer>

  <br><br>
  Это главный (родительский) слой
</ilayer>

<form>
<input type="button" value="Move/Stop parentLayer" onClick="startStop(0);">
<input type="button" value="Move/Stop layer1" onClick="startStop(1);">
<input type="button" value="Move/Stop layer2" onClick="startStop(2);">
</form>

</body>
</html>

```

Можно видеть, что внутри `parentLayer` мы определили два слоя. Это как раз и есть вложенные слои. Как получить к этим слоям доступ в языке JavaScript? Как это делается, можно посмотреть в функции `move()`:

```

document.layers["parentLayer"].left= 100 + pos0;
...
document.layers["parentLayer"].layers["layer1"].top= 10 + pos1;
...
document.layers["parentLayer"].layers["layer2"].top= 10 + pos2;

```

Чтобы получить доступ к вложенным слоям, Вам недостаточно будет просто написать `document.layers["layer1"]` или `document.layers["layer2"]`, поскольку слои `layer1` и `layer2` лежат внутри `parentLayer`.

Посмотрим теперь, как можно задать выделяемую область. В следующем примере используется механизм вырезания и перемещение изображения. Чего этим мы хотим достичь - чтобы вырезаемая часть была зафиксирована, т.е. чтобы при перемещении всего изображения не происходила смена видимого на экране фрагмента.

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

Исходный код скрипта:

```

<html>
<head>

```

```

<script language="JavaScript">
<!-- hide

var pos= 0; // начальное положение
var direction=false;

function moveNclip() {

    if (pos<-180) direction= true;
    if (pos>40) direction= false;

    if (direction) pos+= 2
    else pos-= 2;

    document.layers["clippingLayer"].layers["imgLayer"].top= 100 + pos;

}

// -->
</script>

</head>
<body onLoad="setInterval('moveNclip()', 20);">

<ilayer name="clippingLayer" z-index=0 clip="20,100,200,160" top=0 left=0>
  <ilayer name="imgLayer" top=0 left=0>
    
  </ilayer>
</ilayer>

</body>
</html>

```

И снова, можно видеть пример обращения к вложенному слою:

```
document.layers["clippingLayer"].layers["imgLayer"].top= 100 + pos;
```

С остальными элементами этого скрипта Вы уже должны быть знакомы.

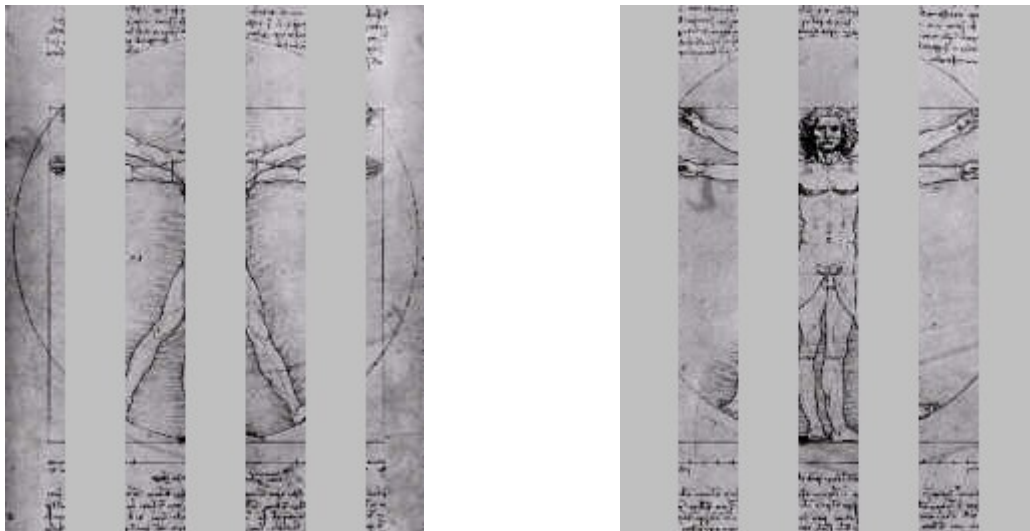
Различные эффекты с вложенными слоями

Интересные эффекты могут быть созданы с помощью (частично) прозрачных слоев. Сочетание специально подобранных изображений с прозрачными областями может создавать совершенно потрясающий результат. Не все форматы изображений поддерживают работу с прозрачными частями. В настоящее время лучший из отвечающих этому условию форматов - gif89a. Большинство новых графических программ поддерживает этот формат. Помимо этого, в Internet доступны некоторые свободно распространяемые инструменты для работы с графикой.

Новый формат изображений PNG также поддерживает эффект прозрачных частей изображения. Я полагаю, что в ближайшем будущем мы увидим множество страниц, использующих этот формат (точнее, как только большинство браузеров смогут его поддерживать). По сравнению с gif этот формат имеет множество преимуществ.

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

В данном примере используются два изображения (сплошные серые зоны здесь на самом деле являются прозрачными):



Сам скрипт несильно отличается от других примеров - так что я не буду здесь его распечатывать (впрочем, Вы конечно можете увидеть его, выбрав в меню Вашего браузера пункт '*View document source*').

В Сети можно найти множество замечательных страниц, основанных на сочетании слоев с прозрачными частями. Некоторые из таких примеров Вы можете найти на моей странице с примерами JavaScript (она является частью home page моей книги о JavaScript и находится по адресу <http://www.dpunkt.de/javascript/>) - сама страница доступна как в английском, так и в немецком варианте.

Я надеюсь, что с помощью этого описания Вы получили представление об основных приемах использования слоев. Поэтому в будущем я надеюсь увидеть действительно прекрасные эффекты, созданные на основе JavaScript...

©1996,1997 by Stefan Koch
e-mail: skoch@rumms.uni-mannheim.de
<http://rummelplatz.uni-mannheim.de/~skoch/>
Моя книга по JavaScript: <http://www.dpunkt.de/javascript>