

# ВВЕДЕНИЕ В JAVASCRIPT ДЛЯ МАГА

© 1996, 1997 Стефан Кох (Stefan Koch)

## Часть 12: Drag & Drop

### Что такое drag & drop?

С помощью новой модели событий в языке JavaScript, 1.2 и механизма слоев мы можем реализовать на нашей web-странице схему drag & drop ("перетащил и оставил"). Для этого Вам понадобится по крайней мере Netscape Navigator 4.0, поскольку мы будем пользоваться особенностями языка JavaScript 1.2.

Что такое drag & drop? Например, некоторые операционные системы (такие как Win95/NT или MacOS) позволяют Вам стирать файлы, просто перетаскивая их в мусорную корзину. Иными словами, Вы щелкаете клавишей мыши над изображением файла, перетаскиваете его (то есть держите клавишу нажатой и просто двигаете мышью) - drag - в мусорную корзину, а затем отпускаете - drop - его там.

Механизм drag & drop, который мы хотим здесь реализовать, ограничивается web-страницей. Поэтому Вы не можете использовать представленный здесь код, чтобы переносить объекты с HTML-страницы на жесткий диск вашего компьютера или другие подобные действия. (Начиная с версии 4.0 браузера Netscape Navigator ваш скрипт может реагировать на событие с названием DragDrop, событие, когда кто-либо перетаскивает файл на окно вашего браузера. Но это не совсем то, о чем мы здесь хотим поговорить)

*(Онлайновая версия руководства позволяет Вам незамедлительно проверить пример, описанный в этом уроке. В окне видны три кубика, которые можно передвигать с помощью мыши)*

Также можно рассмотреть пример, предоставленный компанией Netscape. Найти его Вы сможете по адресу:

[http://home.netscape.com/comprod/products/communicator/user\\_agent\\_vacation.html](http://home.netscape.com/comprod/products/communicator/user_agent_vacation.html)

Язык JavaScript не поддерживает напрямую механизм drag & drop. Это значит, что у нас нет возможности назначить объекту image свойство draggable (перемещаемый) или что-либо в этом роде. Поэтому мы должны сами писать необходимый для этого код. Впрочем, Вы увидите, что это не так сложно.

Итак, то же нам нужно? Нам нужны две вещи. Во-первых, мы должны регистрировать определенные события, связанные с работой мышью, то есть нужно понять, каким образом, мы сможем узнать, какой объект необходимо переместить и на какую позицию? Затем нам нужно подумать, каким именно образом мы сможем показывать перемещение объектов по экрану. Конечно же, мы будем пользоваться такой новой возможностью языка, как слои, при создании объектов и перемещении их по экрану. Каждый объект представлен собственным слоем.

## События при работе с мышью в JavaScript 1.2

Какие события, происходящие при работе с мышью, нам следует использовать? У нас нет такого события, как *MouseDown*, однако того же самого мы можем достичь, отслеживая события *MouseDown*, *MouseMove* и *MouseUp*. В версии 1.2 языка JavaScript используется новая модель событий. И без нее мы не смогли бы решить нашу задачу. Я уже говорил об этой новой модели на предыдущем уроке. Однако давайте взглянем на некоторые важные ее части еще раз.

Пользователь нажал клавишу мыши в каком-либо месте на окне браузера. Наш скрипт должен зафиксировать это событие и вычислить, с каким объектом (то есть слоем) это было связано. Нам необходимо знать координаты точки, где произошло это событие. В JavaScript 1.2 реализован новый объект *Event*, который сохраняет координаты этой точки (а также еще и другую информацию о событии).

Другой важный момент заключается в перехвате событий. Если пользователь, например, щелкает по клавише мыши, то сигнал о соответствующем событии посылается непосредственно объекту *button*. Однако в нашем примере необходимо, чтобы событие обрабатывалось объектом *window* (окно). Поэтому мы позволяем объекту окна перехватывать сигнал о событии, связанном с мышью, т.е. чтобы именно объект *window* фиксировал это событие и имел возможность на него реагировать. Это демонстрируется в следующем примере (на примере события *Click*). Вы можете щелкнуть в любом месте окна браузера. При этом возникнет окно сообщения, где будут показаны координаты точки, где это событие имело место.

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

Код этого примера:

```
<html>

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

    window.captureEvents(Event.CLICK);

    window.onclick= displayCoords;

    function displayCoords(e) {
        alert("x: " + e.pageX + " y: " + e.pageY);
    }

// -->
</script>

Щелкните клавишей мыши где-нибудь в окне браузера.

</html>
```

Сперва мы сообщаем, что объект window перехватывает сигнал о событии *Click*. Для этого мы пользуемся методом *captureEvent()*. Строка

```
window.onclick= displayCoords;
```

говорит о том, что должно происходить, когда случается событие *Click*. Конкретнее, здесь сообщается, что в качестве реакции на событие *Click* браузер должен вызвать процедуру *displayCoords()* (Заметим, что Вам при этом нельзя ставить скобки после слова *displayCoords*). В свою очередь, *displayCoords()* - это функция, которая определяется следующим образом:

```
function displayCoords(e) {  
    alert("x: " + e.pageX + " y: " + e.pageY);  
}
```

Как видите, эта функция имеет аргумент (мы назвали его e). На самом деле это объект Event, который передается на обработку функции *displayCoords()*. Объект Event имеет свойства *pageX* и *pageY* (наряду с другими), из которых можно получить координаты точки, где произошло событие. Окно с сообщением лишь показывает эти значения.

## MouseDown, MouseMove и MouseUp

Как я уже говорил, в языке JavaScript нет события *MouseDown*. Поэтому мы должны пользоваться событиями *MouseDown*, *MouseMove* и *MouseUp*, реализуя механизм drag & drop. В следующем примере демонстрируется применение *MouseMove* - текущие координаты курсора мыши отображаются в окне состояния.

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

Можно видеть, что код скрипта почти такой же, как и в предыдущем примере:

```
<html>  
  
<script language="JavaScript">  
<!--  
  
    window.captureEvents(Event.MOUSEMOVE);  
  
    window.onmousemove= displayCoords;  
  
    function displayCoords(e) {  
        status= "x: " + e.pageX + " y: " + e.pageY;  
    }  
  
// -->  
</script>
```

Координаты мыши показаны в строке состояния.

</html>

Заметьте, что Вам необходимо написать именно *Event.MOUSEMOVE*, где слово *MOUSEMOVE* обязательно должно быть написано заглавными буквами. А указывая, какая функция должна быть вызвана, когда произойдет событие *MouseMove*, Вы должны писать ее строчными буквами: *window.onmousemove=...*

Теперь мы можем объединить оба последних примера. Мы хотим, чтобы были представлены координаты указателя мыши, когда пользователь перемещает мышь, нажав на клавишу.

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

Код этого примера выглядит следующим образом:

<html>

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

*window.captureEvents(Event.MOUSEDOWN | Event.MOUSEUP);*

*window.onmousedown= startDrag;  
window.onmouseup= endDrag;  
window.onmousemove= moveIt;*

*function startDrag(e) {  
  window.captureEvents(Event.MOUSEMOVE);  
}*

*function moveIt(e) {  
  // показывать координаты  
  status= "x: " + e.pageX + " y: " + e.pageY;  
}*

*function endDrag(e) {  
  window.releaseEvents(Event.MOUSEMOVE);  
}*

*// -->  
</script>*

*Двигайте мышь, удерживая ее клавишу нажатой. В окне состояния при этом отображаются координаты курсора.*

</html>

Во-первых, мы заставляем объект *window* перехватывать сигналы о событиях *MouseDown* and *MouseUp*:

```
window.captureEvents(Event.MOUSEDOWN | Event.MOUSEUP);
```

Как видно, мы пользуемся символом | (или), чтобы сказать, что объект window должен перехватывать несколько указанных событий. Следующие две строки описывают, что именно должно происходить, когда указанные события имеют место:

```
window.onmousedown= startDrag;  
window.onmouseup= endDrag;
```

В следующей строке кода определяется, что происходит, когда объект window получает сигнал о событии *MouseMove*.

```
window.onmousemove= moveIt;
```

Однако постойте, мы же не определили *Event.MOUSEMOVE* в *window.captureEvents()*! Это означает, что данное событие не будет перехватываться объектом window. Тогда почему мы указываем объекту window вызывать *moveIt()*, раз сигнал об этом событии никогда не достигает объекта window? Ответ на этот вопрос можно найти в функции *startDrag()*, которая вызывается сразу после того, как произойдет событие *MouseDown*:

```
function startDrag(e) {  
    window.captureEvents(Event.MOUSEMOVE);  
}
```

Это означает, что объект window начнет перехватывать событие *MouseMove*, как только будет нажата клавиша кнопка мыши. И мы должны прекратить перехватывать событие *MouseMove*, если произойдет событие *MouseUp*. Это делается в функции *endDrag()* с помощью метода *releaseEvents()*:

```
function endDrag(e) {  
    window.releaseEvents(Event.MOUSEMOVE);  
}
```

Функция *moveIt()* записывает координаты мыши в окно состояния.

Теперь у нас есть все элементы скрипта, необходимые для регистрации событий, связанных с реализацией механизма drag & drop. И мы можем приступить к рисованию на экране наших объектов.

## Показ движущихся объектов

На предыдущих уроках мы видели, как с помощью слоев можно создать перемещающиеся объекты. Все, что мы должны теперь сделать - это определить, по какому именно слою пользователь щелкнул клавишей мыши. И затем этот объект должен двигаться вслед за мышью. Итак, код примера, показанного в начале этого урока:

```
<html>  
<head>
```

```

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

var dragObj= new Array();
var dx, dy;

window.captureEvents(Event.MOUSEDOWN | Event.MOUSEUP);

window.onmousedown= startDrag;
window.onmouseup= endDrag;
window.onmousemove= moveIt;

function startDrag(e) {
    currentObj= whichObj(e);
    window.captureEvents(Event.MOUSEMOVE);
}

function moveIt(e) {
    if (currentObj != null) {
        dragObj[currentObj].left= e.pageX - dx;
        dragObj[currentObj].top= e.pageY - dy;
    }
}

function endDrag(e) {
    currentObj= null;
    window.releaseEvents(Event.MOUSEMOVE);
}

function init() {
    // задать 'перемещаемые' слои
    dragObj[0]= document.layers["layer0"];
    dragObj[1]= document.layers["layer1"];
    dragObj[2]= document.layers["layer2"];
}

function whichObj(e) {

    // определить, по какому объекту был произведен щелчок

    var hit= null;
    for (var i= 0; i < dragObj.length; i++) {
        if ((dragObj[i].left < e.pageX) &&
            (dragObj[i].left + dragObj[i].clip.width > e.pageX) &&
            (dragObj[i].top < e.pageY) &&
            (dragObj[i].top + dragObj[i].clip.height > e.pageY)) {
            hit= i;
            dx= e.pageX- dragObj[i].left;
            dy= e.pageY- dragObj[i].top;
            break;
        }
    }
}

```

```

    }
    return hit;
}

// -->
</script>
</head>
<body onLoad="init()">

<layer name="layer0" left=100 top=200 clip="100,100" bgcolor="#0000ff">
<font size=+1>Object 0</font>
</layer>

<layer name="layer1" left=300 top=200 clip="100,100" bgcolor="#00ff00">
<font size=+1>Object 1</font>
</layer>

<layer name="layer2" left=500 top=200 clip="100,100" bgcolor="#ff0000">
<font size=+1>Object 2</font>
</layer>

</body>
</html>

```

Можно видеть, что в разделе `<body>` нашей HTML-страницы мы определяем три слоя. После того, как была загружена вся страница, при помощи программы обработки события `onLoad`, указанной в тэге `<body>`, вызывается функция `init()`:

```

function init() {
    // define the 'draggable' layers
    dragObj[0]= document.layers["layer0"];
    dragObj[1]= document.layers["layer1"];
    dragObj[2]= document.layers["layer2"];
}

```

Массив `dragObj` включает все слои, которые пользователь может перемещать. Каждый такой слой получает в множестве `dragObj` некий номер. Его мы рассмотрим попозже. Можно видеть, что мы используем тот же самый код, что использовался ранее для перехвата событий, связанных с мышью:

```

window.captureEvents(Event.MOUSEDOWN | Event.MOUSEUP);

window.onmousedown= startDrag;
window.onmouseup= endDrag;
window.onmousemove= moveIt;

```

К функции `startDrag()` я добавил следующую строку:

```

currentObj= whichObj(e);

```

Функция *whichObj()* определяет, по какому объекту был произведен щелчок. Возвращает она номер соответствующего слоя. Если ни один слой не был выделен, то возвращается значение *null*. Полученное значение хранится в переменной *currentObj*. Это означает, что из *currentObj* можно извлечь номер слоя, который в данный момент необходимо перемещать (либо это будет *null*, если никакого слоя перемещать не надо).

В функции *whichObj()* для каждого слоя мы проверяем свойства *left*, *top*, *width* и *height*. По этим значениям мы и можем проверять, по которому из объектов пользователь щелкнул клавишей.

## "Оставляемые" объекты

Теперь мы имеем все, что необходимо, чтобы реализовать механизм drag & drop. С помощью нашего скрипта пользователь может перемещать объекты по web-странице. Однако мы еще ничего не говорили об размещении перемещенных объектов. Предположим, Вы хотите создать онлайн-магазин. У нас есть несколько изделий, которые можно поместить в корзину. Пользователь должен переносить эти изделия в корзинку и оставлять их там. Это означает, что мы должны регистрировать моменты, когда пользователь опускает некий объект в корзину - иными словами, что он хочет купить его.

Какую часть кода мы должны изменить, чтобы сделать такое? Мы должны проверить, в какой месте оказался объект после того, как было зафиксировано событие *MouseUp* - то есть мы должны сделать некоторые добавления к функции *endDrag()*. Например мы могли бы проверять, попадает ли в этот момент курсор мыши в границы некого прямоугольника. Если это так, то Вы вызываете функцию, регистрирующую все изделия, которые необходимо купить (например, Вы можете поместить их в некий массив). Ну и после этого Вы можете показывать это изделие уже в корзинке.

## Реализации

Есть несколько путей для совершенствования нашего скрипта. Во-первых, мы могли бы изменять порядок следования слоев, как только пользователь щелкает клавишей мыши по какому-либо объекту. Иначе выглядело бы странным, если бы Вы перемещали объект, а он при этом прятался от Вас за окружающие предметы. Очевидно, что эту проблему можно решить, меняя лишь порядок следования слоев в функции *startDrag()*.

Я далек от мысли, что Вас устроит перемещение красных, зеленых и синих кубиков по Вашей web странице. Добавьте немного красивой графики, и читатели уже запомнят Вашу страницу. Вы можете поместить что-либо в объект слоя. Например, положите туда один тэг *<img>*, если хотите, чтобы ваш объект предстал в виде графического изображения.