

# ВВЕДЕНИЕ В JAVASCRIPT ДЛЯ МАГА

© 1996, 1997 Стефан Кох (Stefan Koch)

## Part 11: Модель событий в JavaScript 1.2

### Новые события

Наступило время, рассмотреть одну из новых особенностей Netscape Navigator 4.x - модель событий JavaScript 1.2. Приведенные здесь примеры будут работать только в Netscape Navigator 4.x (хотя большинство из них работают также и в предварительных версиях этого браузера).

В JavaScript 1.2 поддерживается обработка следующих событий (если Вы хотите узнать побольше об этих событиях, обратитесь к документации JS 1.2 от фирмы Netscape - [http://developer.netscape.com/library/documentation/communicator/jsguide/js1\\_2.htm](http://developer.netscape.com/library/documentation/communicator/jsguide/js1_2.htm)):

Abort	Focus	MouseOut	Submit
Blur	KeyDown	MouseOver	Unload
Click	KeyPress	MouseUp	
Change	KeyUp	Move	
DblClick	Load	Reset	
DragDrop	MouseDown	Resize	
Error	MouseMove	Select	

Изучая таблицу, можете увидеть, что была реализована обработка некоторых новых событий. На этом уроке мы и рассмотрим некоторые из них.

Сперва давайте рассмотрим событие `Resize`. С помощью этого события Вы можете определить, был бы размер окна изменен читателем. Следующий скрипт демонстрирует, как это делается:

```
<html>
<head>
<script language="JavaScript">
```

```
    window.onresize= message;
```

```
function message() {
    alert("The window has been resized!");
}
```

```
</script>
</head>
<body>
```

*Пожалуйста, измените размер этого окна.*

```
</body>
</html>
```

В строке

```
window.onresize= message;
```

мы задаем процедуру обработки такого события. Точнее, функция `message()` будет вызываться всякий раз, как только пользователь изменит размер окна. Возможно, Вы не знакомы с таким способом назначения программ, обрабатывающих события. Однако JavaScript 1.2 ничего нового здесь не привносит. Например, если у Вас есть объект `button`, то Вы можете определить процедуру обработки события следующим образом:

```
<form name="myForm">
<input type="button" name="myButton" onClick="alert('Click event occured!')">
</form>
```

Однако Вы можете написать это и по другому:

```
<form name="myForm">
<input type="button" name="myButton">
</form>
```

...

```
<script language="JavaScript">
```

```
document.myForm.myButton.onclick= message;
```

```
function message() {
  alert('Click event occured!');
}
```

```
</script>
```

Можно подумать, что вторая альтернатива немного сложнее. Однако почему тогда именно ее мы используем в первом скрипте? Причина состоит в том, что объект `window` нельзя определить через какой-либо определенный тэг - поэтому нам и приходится использовать второй вариант.

Два важных замечания: Во-первых, Вам не следует писать `window.onResize` - я имею в виду, что Вы должны писать все прописными буквами. Во-вторых, Вы не должны ставить после сообщения никаких скобок. Если Вы напишете `window.onresize= message()`, то браузер интерпретирует `message()` как вызов функции. Однако в нашем случае мы не хотим напрямую вызывать эту функцию - мы лишь хотим определить обработчик события.

## Объект Event

В язык JavaScript 1.2 добавлен новый объект Event. Он содержит свойства, описывающие некое событие. Каждый раз, когда происходит какое-либо событие, объект Event передается соответствующей программе обработки. В следующем примере на экран выводится некое изображение. Вы можете щелкнуть где-нибудь над ним клавишей мыши. В результате появится окошко сообщений, где будут показаны координаты той точки, где в этот момент находилась мышь.

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

Код скрипта:

```
<layer>
<a href="#" onClick="alert('x: ' + event.x + 'y: ' + event.y); return false;">
</a>
</layer>
```

Как видите, в тэг <a> мы поместили программу обработки событий onClick, как это мы уже делали в предшествующих версиях JavaScript. Новое здесь заключается в том, что для создания окошка с сообщением мы используем event.x и event.y. А это как раз и есть объект Event, который здесь нам нужен, чтобы узнать координаты мыши.

К тому же я поместил все команды в тэг <layer>. Благодаря этому мы получаем в сообщении координаты относительно данного слоя, т.е. в нашем случае относительно самого изображения. В противном же случае мы получили бы координаты относительно окна браузера. (инструкция *return false;* используется здесь для того, чтобы браузер обрабатывал далее данную ссылку)

Объект Event получил следующие свойства (их мы рассмотрим в следующих примерах):

Property	Description
<i>Data</i>	Массив адресов URL оставленных объектов, когда происходит событие <i>DragDrop</i> .
<i>LayerX</i>	Горизонтальное положение курсора (в пикселах) относительно слоя. В комбинации с событием <i>Resize</i> это свойство представляет ширину окна браузера.
<i>LayerY</i>	Вертикальное положение курсора (в пикселах) относительно слоя. В комбинации с событием <i>Resize</i> это свойство представляет высоту окна браузера.
<i>modifiers</i>	Строка, задающая ключи модификатора - <i>ALT_MASK</i> , <i>CONTROL_MASK</i> , <i>META_MASK</i> or <i>SHIFT_MASK</i>
<i>pageX</i>	Горизонтальное положение курсора (в пикселах) относительно окна браузера.
<i>pageY</i>	Вертикальное положение курсора (в пикселах) относительно окна браузера.
<i>screenX</i>	Горизонтальное положение курсора (в пикселах) относительно экрана.
<i>screenY</i>	Вертикальное положение курсора (в пикселах) относительно экрана.
<i>target</i>	Строка, представляющая объект, которому исходно было послано событие.
<i>type</i>	Строка, указывающая тип события.

<i>which</i>	ASCII-значение нажатой клавиши или номер клавиши мыши.
<i>x</i>	Синоним <i>layerX</i> .
<i>y</i>	Синоним <i>layerY</i> .

## Перехват события

Одна из важных особенностей языка - перехват события. Если кто-то, к примеру, щелкает на кнопке, то вызывается программа обработки события `onClick`, соответствующая этой кнопке. С помощью обработки событий Вы можете добиться того, чтобы объект, соответствующий вашему окну, документу или слою, перехватывал и обрабатывал событие еще до того, как для этой цели объектом указанной кнопки будет вызван обработчик событий. Точно так же объект вашего окна, документа или слоя может обрабатывать сигнал о событии еще до того, как он достигает своего обычного адресата.

Чтобы увидеть, для чего это может пригодиться, давайте рассмотрим следующий пример:

```
<html>
<head>
<script language="JavaScript">

window.captureEvents(Event.CLICK);

window.onclick= handle;

function handle(e) {
    alert("Объект window перехватывает это событие!");
    return true; // т.е. проследить ссылку
}

</script>
</head>
<body>
<a href="test.htm">Click on this link</a>
</body>
</html>
```

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

Как видно, мы не указываем программы обработки событий в тэге `<a>`. Вместо этого мы пишем

```
window.captureEvents(Event.CLICK);
```

с тем, чтобы перехватить событие *Click* объектом `window`. Обычно объект `window` не работает с событием *Click*. Однако, перехватив, мы затем его переадресуем в объект `window`. Заметим, что в `Event.CLICK` фрагмент `CLICK` должен писаться заглавными буквами. Если же Вы хотите перехватывать несколько событий, то Вам следует отделить их друг от друга символами `|`. Например:

```
window.captureEvents(Event.CLICK | Event.MOVE);
```

Помимо этого в функции *handle()*, назначенной нами на роль обработчика событий, мы пользуемся инструкцией *return true*;. В действительности это означает, что браузер должен обработать и саму ссылку, после того, как завершится выполнение функции *handle()*. Если же Вы напишете вместо этого *return false*;, то на этом все и закончится.

Если теперь в тэге *<a>* Вы зададите программу обработки события *onClick*, то поймете, что данная программа при возникновении данного события вызвана уже не будет. И это не удивительно, поскольку объект *window* перехватывает сигнал о событии еще до того, как он достигает объекта *link*. Если же Вы определите функцию *handle()* как

```
function handle(e) {  
    alert("The window object captured this event!");  
    window.routeEvent(e);  
    return true;  
}
```

то компьютер будет проверять, определены ли другие программы обработки событий для данного объекта. Переменная *e* - это наш объект *Event*, передаваемый функции обработки событий в виде аргумента.

Кроме того, Вы можете непосредственно послать сигнал о событии какому-либо объекту. Для этого Вы можете воспользоваться методом *handleEvent()*. Это выглядит следующим образом:

```
<html>  
<script language="JavaScript">  
  
window.captureEvents(Event.CLICK);  
  
window.onclick= handle;  
  
function handle(e) {  
    document.links[1].handleEvent(e);  
}  
  
</script>  
<a href="test.htm"> "Кликните" по этой ссылке</a><br>  
<a href="test.htm"  
    onClick="alert('Обработчик событий для второй ссылки!');">Вторая ссылка</a>  
</html>
```

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

Все сигналы о событиях *Click*, посылаются на обработку по второй ссылке - даже если Вы вовсе и не щелкнули ни по одной из ссылок!

Следующий скрипт демонстрирует, как Ваш скрипт может реагировать на сигналы о нажатии клавиш. Нажмите на какую-либо клавишу и посмотрите, как работает этот скрипт.

```
<html>
<script language="JavaScript">

window.captureEvents(Event.KEYPRESS);

window.onkeypress= pressed;

function pressed(e) {
    alert("Key pressed! ASCII-value: " + e.which);
}

</script>
</html>
```

©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>