ЛАБОРАТОРНАЯ РАБОТА 2
Язык программирования JavaScript. События в модели документа

Цель работы – научиться использовать имеющиеся в модели документа события для внесения изменений в страницу.

Наиболее часто в сценариях используется рассмотренное выше событие *onclick*. Для того чтобы обратить внимание пользователя на определённый элемент HTML-документа, можно менять свойства этого элемент при попадании на него курсора мышки, а при снятии курсора восстанавливать прежние значения свойств. Например, можно менять цвет или размер элемента. Попадание курсора мышки на элемент фиксируется событием *onMouseOver*. Парное для него событие *onMouseOut* происходит при снятии курсора мышки с элемента.

Другая пара событий *onMouseDown* и *OnMouseUp* происходит при нажатии и отпускании левой кнопки мышки. Эту пару событий удобно применять для изменения свойств элементов или замены элементов на время удержания кнопки мышки нажатой.

Реакция на событие в отдельном элементе

Так как в объектной модели объекты могут быть вложены друг в друга, то событие, происходящее в дочернем объекте, одновременно происходит и в родительском объекте. JavaScript предоставляет различные способы локализации влияния события на иерархию объектов. Простейшей способ локализации (пример 2.1) заключается в размещении сценария в теге, на который должно воздействовать событие.

Пример 2.1.

```
<HTML>
<BODY>
<P align=right ID='alfa'
onMouseOver="document.all.alfa.align='center'
"onMouseOut="this.align='left'"">
События onMouseOver и onMouseOut </p>
</body>
</html>
```

Страница в примере 2.1 состоит из одной строки, заключённой в контейнер <*P*> …</*p*>. В объектной модели страницы событие, происходящее с объектом *P*, происходит также и с родительским объектом BODY. Чтобы локализовать реакцию на событие только пределами строки, то есть объекта *P*, сценарий реакции на события помещен в тег <*P* >.

В результате исполнения сценария изменяется положение текста на строке. Первоначально строка прижата к правому краю окна. При попадании на неё

курсора она выравнивается по центру, а после снятия курсора прижимается к левому краю окна. Для обращения к объекту используется коллекция *all*, которая правильно воспринимается браузерами Internet Explorer 6.0 и Mozilla Firefox 2.0 . Ключевое слово *this* означает ссылку на текущий объект.

Если при наступлении события нужно произвести много действий, то удобно сценарий написать в виде функции и поместить её отдельно от элемента в специально предназначенный для сценариев контейнер <*SCRIPT*> …</*script*>. В примере 2.2 каждое из событий*onMouseOver* и *onMouseOut* вызывает два действия: выравнивание и изменение цвета текста в строке.

Пример 2.2

```
<HTML>
<P align=right ID='alfa' onMouseOver="M_Over()"
onMouseOut="M_Out()">
Событие onMouseOver</p>
<SCRIPT>
function M_Over()
{ document.all.alfa.align='center'
  document.all.alfa.style.color='FF00FF'
}
function M_Out()
{ document.all.alfa.align='left'
  document.all.alfa.style.color='0000FF'
}
</script>
</html>
```

Задача 2.1. Напишите HTML-документ, отображающийся в окне браузера в виде следующих четырёх строк:

- Пять событий с мышкой
- Щёлкните по мне мышкой
- На этом тексте нажмите, подержите и отпустите левую кнопку мышки
- Медленно проведите курсором мышки по этой надписи

Первая строка – заголовок страницы. Вторая строка меняется при щелчке мышкой следующим образом:

- шрифт увеличивается до 48pt;
- цвет шрифта меняется на белый;
- цвет фона меняется на голубой.

Повторный щелчок мышкой возвращает вторую строку к первоначальному виду.

Фон третьей строки меняется, когда курсор мышки находится на ней и нажимается или отпускается левая кнопка мышки. При нажатии фон становится зелёным, а при отпускании – жёлтым.

При попадании курсора мышки на четвёртую строку её фон становится красным, а при снятии – голубым.

Фиксация события в родительском элементе

Если реакцию на какое-либо событие требуют несколько элементов, расположенных на странице, то можно вызвать функцию для обработки этого события только в родительском элементе. В функции определяется, на каком элементе произошло событие, и выполняются соответствующие действия. Удобство такого подхода состоит в том, что весь алгоритм преобразований находится в одном месте, а недостаток – в сложности самой функции. Рассмотрим сценарий (пример 2.3), в котором для изменения свойств любого из трёх объектов, находящихся в окне браузера служит одна функция.

Пример 2.3

```
<HTML>
<HEAD>
<TITLE>Реакция на событие</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
<STYLE>H1 {color:FF00FF} #k1{position:absolute; left:50;top:200;
width:300; height:100;background-color:blue}
#k2{position:relative;
left:50;top:25; width:200; height:50;}
</STYLE></HEAD> <BODY ID="B" bgcolor="AAAAAA"
onclick="rodEl()">
<H1 ID="HH" >ЦВЕТ</H1> <DIV ID="k1" >
<DIV ID="k2">
</div>
</div>
</BODY>
<SCRIPT> /*
Функция запускается при щелчке мышкой по любой точке документа */
function rodEl()
{ var e=window.event
  id1=e.srcElement.id
  switch(id1)
  { case "k1":
//Изменение цвета внешнего прямоугольника
z=document.getElementById(id1).style.backgroundColor
if(z!="red")z="red"
else
  z="green" document.getElementById(id1).style.backgroundColor=z
break
    case "k2":
//Изменение цвета внутреннего прямоугольника
z=document.getElementById(id1).style.backgroundColor
if(z!="#00ffff"){z="#00ffff"} else{z="FF00FF"}
```

```
  document.getElementById(id1).style.backgroundColor=z
break
    case "B": //Изменение цвета заднего плана документа
        document.getElementById(id1).bgColor="777777"
        break
    case "HH": //Изменение цвета слова "Цвет"
        document.getElementById(id1).style.color= "AA00AA"
}
}
</SCRIPT>
</HTML>
```

В примере 2.3 родительским по отношению к элементу *H1* и двум элементам *DIV* является элемент *BODY*. Поэтому в теге <*BODY*> вызывается функция *rodEl()*, служащая для обработки события *onclick*.

В момент наступления события вся информация о нём запоминается в объекте *event*, дочернем по отношению к объекту *window*. В примере используется свойство *srcElement*, хранящее в качестве значения объект на котором произошло событие. Поэтому выражение*srcElement.Id* читается так: **Id** объекта, на котором произошло событие. К сожалению, свойство *srcElement* имеется только в объектной модели, поддерживаемой браузером Internet Explorer, поэтому сценарий данного примера в других браузерах работать не будет.

В сценарии сначала определяется *Id* элемента, по которому пользователь щёлкнул мышкой, а затем с помощью оператора *Switch* делается переход к изменению свойств указанного элемента. Цвет слова *ЦВЕТ* и фона документа меняется только один раз, а цвет прямоугольников – при каждом щелчке по ним мышкой.

В примере 2.3 для того, чтобы хорошо была видна структура документа, громоздкие описания параметров вынесены в CSS.

Задача 2.2. Создайте страницу с изображением и подписью под ним. При щелчке по подписи, она должна менять свой цвет. Щелчок по изображению должен вызывать замену изображения и подписи. Функция для обработки события должна вызываться из родительского по отношению к изображению и подписи объекта.

Предотвращение всплывания события. Свойство cancelBubble

В примере 2.3 рассматривалась простая страница с небольшим количеством элементов, но даже в таком простом случае функция реакции на событие получилась сложной. Проще для каждого элемента написать свою функцию обработки события, а распространение события вверх по дереву иерархической структуры от "детей" к "родителям" (в этом случае говорят о всплывании события) блокировать с помощь специально для этого предназначенного свойства *cancelBubble* объекта event. Изменим пример 2.3 так, чтобы для реакции на щелчок по каждому из четырёх элементов страницы служила своя функция:

Пример 2.4

```
<HTML>
<HEAD><TITLE>Реакция насобытие</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
<STYLE>H1 {color:FF00FF} #k1{position:absolute;
left:50;top:200; width:300; height:100;background-color:blue}
#k2{position:relative; left:50;top:25; width:200;
height:50;</STYLE></HEAD>
<BODY ID="B" bgcolor= "AAAAAA" onclick="rodEl()">
<H1 ID="HH"onclick="H_l()">
ЦВЕТ</H1>
<DIV ID="k1" onclick="D_l()">
<DIV ID="k2" style=" background-color:yellow"
onclick="D_2(this)"></div>
</div>
</BODY>
<SCRIPT>
/* Функция запускается при щелчке мышкой по
любой точке документа */
function rodEl()
{ //Изменение цвета заднего плана документа
z=document.all.B.bgColor
if(z!="#777777"){z="#777777"}
else{z="AAAAAA"}
document.all.B.bgColor =z
}
function D_l()
{ //Изменение цвета внешнего прямоугольника
z= document.all.k1.style.backgroundColor
if(z!="red")z="red"
else z="green"
document.all.k1.style.backgroundColor=z
}
function D_2(thi)
{ //Изменение цвета внутреннего прямоугольника
z=thi.style.backgroundColor
if(z!="#00ffff"){z="#00ffff"}
else{z="FF00FF"}
thi.style.backgroundColor=z
}
function H_l()
{ //Изменение цвета слова "Цвет"
z=document.all.HH.style.colorif(z!="#aa00aa"){z="#aa00aa"}
else{z="00FFFF"}
document.all.HH.style.color= z
}
</SCRIPT>
</HTML>
```

Для лучшего проявления эффекта всплывания событий в пример 2.4 добавлено изменение фона всего документа и цвета надписи при каждом щелчке мышкой по ним. Скопируйте в свой каталог и просмотрите пример 2.4 в браузере Internet Explorer. Щёлкните по слову*ЦВЕТ*. Изменится не только цвет слова, но и цвет фона документа, так как после щелчка сначала выполнится

функция H_1(), а затем событие всплывёт к родительскому элементу BODY и выполнится функция rodE1(). При щелчке по внутреннему прямоугольнику будут меняться цвета обоих прямоугольников и фона документа. Щелчок по внешнему прямоугольнику изменит цвет этого прямоугольника и цвет фона документа.

Задача 2.3. Чтобы предотвратить всплывание события в примере 2.4, вставьте в начало всех функций, кроме первой, оператор

```
window.event.cancelBubble=true
```