Работа с объектами DOM - 2

10. Как связать объект и событие средствами языка JavaScript

До сих пор прослушивание событий в примерах было включено постоянно с помощью средств HTML. Так в примере 26 для этих целей в тег **<body>** включен атрибут

onmouseover="fover(event)"

Существуют задачи, в которых прослушивание события нужно включать и отключать в процессе исполнения программы. Это можно сделать средствами языка JavaScript.

На языке JavaScript включение прослушивания события объектом записывается в виде команды

имя_объекта.имя_события = имя_функции_реакции;

Например,

document.onmouseover = fover;

Заметим, что в команде имя функции не заключается в кавычки и не имеет скобок для аргумента. Выключение прослушивания объектом события на языке JavaScript записывают в виде команды

имя_объекта.имя_события = null;

Например,

document.onmouseover = null;

Пример 28.

В условиях примера 26 будем нажатием на первую кнопку включать прослушивание документом события **onmouseover** и отображение на закладке имени тега под указателем мыши, а нажатием на третью кнопку выключать прослушивание события **onmouseover** и включать отображение старого заголовка на закладке.

Используем для изменения файл **pos26.html**. Сразу меняем номер примера в тексте и сохраняем его под именем **pos28.html**.

Прежде всего, удаляем из тега **<body>** атрибут

onmouseover="fover(event)"

Далее в тело функции **func1()** дописываем команду включения прослушивания объектом **document** события **onmouseover**

```
document.onmouseover = fover;
```

В тело функции **func3()** дописываем команду выключения прослушивания

```
document.onmouseover = null;
```

и команду возврата исходного заголовка

```
document.title = "Пример 28";
```

Сохраняем файл **pos28.html** и запускаем его.

11. Перетаскивание объектов по окну браузера

Пример 29 (Перетаскивание объектов).

Разместим в окне браузера две картинки, текстовый абзац и закрашенный зеленым цветом прямоугольник, который можно построить при помощи тега

** **. Реализуем возможность перетаскивания по окну отображаемых объектов при помощи указателя мыши.

Воспользуемся файлом **pos25.html**. Сразу меняем название "Пример 29. Перетаскивание" и сохраняем файл под именем **pos29.html**.

Заметим, что для реализации перетаскивания у объекта обязательно должны быть заданы стилевые свойства: **position**, **top** и **left**.

Сначала займемся тегами. В теге **<body>** удаляем атрибуты прослушивания событий. Оставляем только

```
<body style="background:brown;" >
```

В теге **<p>** первое стилевое правило **text-align:center;** заменяем новыми.

Установим ширину

width:100px;

и положение

position:absolute;top:100px;left:500px;

Меняем и текст (содержимое текстового контейнера):

```
<p style="width:100px; position:absolute; top:100px; left:500px;+
font-family: Arial; color:white;">Перетаскивай!!!</p>
```

В теге **<img>** удаляем атрибут идентификатора и увеличиваем размеры

```
<img src="pic1.jpg" style="height:150px; position:absolute;+
top:200px; left:200px;">
```

Добавляем копированием такой же тег для **pic2.jpg**, в котором размер можно оставить, а вот положение лучше изменить, чтобы картинки не накладывались

```
<img src="pic2.jpg" style="height:150px; position:absolute;+
top:30px; left:30px;">
```

Добавляем тег-прямоугольник

```
<div style = "position:absolute;top:350px;left:450px;width:280px;+
height:70px; background-color:green;"></div>
```

Теперь займемся вопросами перетаскивания объектов.

Прежде всего, при перетаскивании нужно выключить автоматическое выделение, которое браузеры производят в окне, когда по нему перетаскивают курсор мыши. Для этого в тег **<body>** добавляем атрибуты

```
ondragstart="return false" onselectstart="return false"
```

При перетаскивании следует обрабатывать три события **onmousedown, onmousemove** и **onmouseup**. Для обработки будем использовать функции **dn()**, **mv()** и **up()** соответственно.

Если мы хотим, чтобы в окне перетаскивались любые объекты, устанавливаем в тег **<body>** атрибут прослушивания события:

```
onmousedown= "dn(event)".
```

События **onmousemove** и **onmouseup** сначала прослушиваться не должны (ничего не перетаскивается).

Очищаем внутреннюю область тега **<script></script>** для команд языка JavaScript. Функция **dn()** будет определять объект, на котором находится указатель мыши с нажатой кнопкой. Для хранения имени перетаскиваемого объекта задаем глобальную переменную **ob**

```
var ob;
```

Далее вводим три конструкции функции пользователя.

Первая функция начинается командами

```
function dn(e) {
     e = e|| event;
     ob = e.target || e.srcElement;     // определяется объект

}
```

Функция **dn()** кроме определения перетаскиваемого объекта должна включать прослушивание событий **onmousemove** (перетаскивание) и **onmouseup** (поднятие мыши и отключение от объекта).

Для объекта **document** и события **onmousemove** команда включения прослушивания имеет вид

```
document.onmousemove = mv;
```

а для объекта **document** и события **onmouseup** —

```
document.onmouseup = up;
```

Дописываем эти команды в тело функции **dn()**. Дописываем также в тело функции последнюю команду

```
     return false
```

которая выключает возможную дополнительную реакцию браузера на событие, вызвавшее функцию. Получаем

```
function dn(e) {
     e = e|| event;
     ob = e.target || e.srcElement; //определяется объект
     document.onmousemove = mv;     //включается прослушивание
     document.onmouseup = up;       //включается прослушивание
     return false
}
```

В теле функции **mv()**, если объект **ob** определен (имеет смысл), задаются значения смещения перетаскиваемого объекта в окне в соответствии с координатами указателя мыши. Последняя команда в теле функции должна выключить возможную дополнительную реакцию браузера на событие, вызвавшее функцию.

```
function mv(e){
     e = e|| event;
     if (ob) {                              // если объект определен
       ob.style.left = e.clientX-50+'px'; // x от курсора объекту
       ob.style.top = e.clientY-50+'px';  // y от курсора объекту
     }
     return false
}
```

Третья функция должна "обнулить" переменную, в которой был задан объект, а также выключить прослушивание событий **onmousemove** и **onmouseup**.

```
function up(){
     ob = null;                            //обнуляется переменная
```

```
    document.onmousemove = null;       //выключается прослушивание
    document.onmouseup = null;         //выключается прослушивание
}
```

Сохраняем файл **pos29.html** и запускаем его.

Если перетаскивать нужно только определенные теги, то из тега **<body>** атрибут **onmousedown="dn(event)"** удаляем и добавляем его в избранные теги. Именно эти объекты и будут перетаскиваться.

Задание.

1). В программе решения примера 29 включите перетаскивание только для картинок, только для прямоугольника.

2). В примере 29 в начале перетаскивания объект мгновенно перемещается ("дергается") в фиксированную позицию относительно указателя мыши. Эта позиция задана в функции **mv()** — левый верхний угол объекта располагается на **50px** выше и левее курсора. Измените текст программы так, чтобы объект при перетаскивании сохранял первоначальную позицию относительно курсора. Для этого нужно ввести дополнительно глобальные переменные для хранения первоначального сдвига курсора относительно верхнего левого угла объекта; сдвиг по **x** и по **y** можно вычислить в теле функции **dn()**, а использовать в теле функции **mv()**.

(!!! Задание 2 после выполнения показать преподавателю!!! 18)

Программа решения примера 29 позволяет перетаскивать по окну браузера любые объекты, которые не имеют дочерних. Объекты-контейнеры ведут себя по-иному.

Пример 30 (Перетаскивание контейнеров).

Изменим текст программы решения примера 29 так, чтобы объект ** ** включал обе картинки.

В тексте файла **pos28.html** переносим оба тега **<img>** во внутренность тегов ** **, меняем номер примера на 30, сохраняем текст в файле **pos30.html** и запускаем его. В результате все объекты все равно перемещаются по одному, за исключением объекта ** **, который перетаскивается сам и одновременно тащит за собой дочерние картинки.

Заметим, что смещение картинок, заключенных в теги ** **, определяется не относительно левого верхнего угла окна браузера, а относительно левого верхнего угла родительского объекта.

Продолжение примера 30.

Чтобы исключить обособленное перемещение картинок, можно из их таблиц стилей удалить стилевые правила, связанные с их положением: **position**, **top** и **left**. Так как менять значения свойств **top** и **left** в программе на языке JavaScript (перетаскивать) можно только объектам, у которых они заданы с самого начала.

Увеличим также размеры родительского прямоугольника, чтобы картинки вошли полностью.

```
width:350px; height:200px;
```

а смещение по вертикали уменьшим:

```
top:150px;
```

Сохраняем текст файла **pos30.html** и запускаем его.

Обособленное перетаскивание картинок прекратилось, но опускание указателя на картинку не воспринимается браузером как опускание на прямоугольник.

Продолжение примера 30.

Чтобы опускание указателя мыши на картинку (дочерний объект), воспринималось браузером как опускание указателя на прямоугольник (родительский объект), в тело функции **dn()** после второй команды (определения объекта) вставляем команду

```
if (ob.parentNode.tagName=='div')  ob=ob.parentNode;
```

которая в случае, когда объект под указателем мыши является дочерним для объекта ** **, выбирает для перетаскивания именно родительский объект ** ** (**parentNode** — родительский объект)

Сохраняем текст файла **pos30.html** и запускаем его.

Теперь программа будет работать, даже если задавать положение картинок внутри объекта с помощью стилевых свойств **position**, **top** и **left**.

Задание.

В примере 30 увеличьте вертикальный размер прямоугольника вдвое и задайте положение второй картинки в правом нижнем углу прямоугольника.

12. Автоматическое прямолинейное движение объекта

Пример 31.

В левом верхнем углу окна браузера заданы две кнопки Пуск и Стоп, а ниже и правее — голубой прямоугольник, внутри которого находится объект-картинка. Нажатие кнопки Пуск запускает горизонтальное движение объекта вправо. Доходя до края прямоугольника, объект меняет движение на противоположное. Нажатие кнопки Стоп останавливает движение.

Изменим текст файла **pos7.html**. Меняем номер примера и сохраняем файл с именем **pos31.html**.

Полностью удаляем из текста теги ** ** и **<input>** третьей кнопки.

На первой кнопке меняем текст на "Пуск", на второй — на "Стоп".

Задаем тег прямоугольника с описанием цвета фона, положения, размера, толщины границы и ее цвета:

```
<div style = "background-color:lightblue;position:absolute;+
top:150px; left:50px; width:400px; height:300px;">
</div>
```

Внутри тега прямоугольника задаем тег изображения для файла ball.gif с размером **40px**.

Поместим середину картинки строго посередине прямоугольника. Напоминаем, что для картинки можно задать положение только левого верхнего угла. Тогда

рассчитываем "координаты" средней точки прямоугольника (**top:150px; left:200px**) и угол картинки размером 40x40 смещаем на 20px левее и выше — **top:130px; left:180px**. Теперь середина картинки на месте.

```
<img id="ball" src="ball.gif" style="position:absolute;top:130px;+
left:180px; height:40px;">
```

Идентификатор нужен для организации сдвига картинки.

Теперь займемся функциями. Запускать движение будет функция **func1()**, останавливать — **func2()**, а сдвиг объекта на некоторое расстояние по окну будет проводить функция **func3()**.

Значение элементарного сдвига **10(px)** задаем как значение глобальной переменной

```
var dx=10;
```

Программы на языке JavaScript исполняются на компьютере практически мгновенно. Чтобы задать видимое перемещение объекта, функцию сдвига запускают в режиме повторения через определенный промежуток времени.

Запуск функции **func3()** через каждые 20 миллисекунд (20/1000 сек) проводит команда

```
setInterval("func3()",20);
```

Чтобы иметь возможность остановить движение, для функции **setInterval()** нужно задать переменную-идентификатор (любое имя, пусть **sf3**) командой вида

```
sf3 = setInterval("func3()",20);
```

Именно эту команду и записываем в тело функции **func1()**.

Остановка движения в теле функции **func2()** задается командой

```
clearInterval(sf3);
```

Отметим, что аргумент функции **clearInterval()** записывается без кавычек.

Будем строить функцию **func3()**. Значение свойства

document.getElementById('ball').style.left

есть строковое значение левого отступа картинки (ее левого верхнего угла) от левого края прямоугольника. Тогда новое, но уже числовое значение отступа (со сдвигом dx) вычисляем как значение вспомогательной переменной nleft:

```
var nleft=parseInt(document.getElementById('ball').style.left)+dx;
```

Присваиваем левому отступу объекта это новое значение

```
document.getElementById('ball').style.left = nleft;
```

Далее записываем команды изменения знака сдвига при достижении объектом края прямоугольника

```
if (nleft>=360) dx=-dx;  //у правого края
if (nleft<=0) dx=-dx;    //у левого края
```

В условии первой из команд 360 — это значение отступа слева левого угла картинки в момент, когда правый край картинки достигает правого края прямоугольника (360=400 – 40, т.е. ширина прямоугольника минус ширина картинки).

Сохраняем текст файла pos30.htm и запускаем его. Требуемое движение воспроизводится. Однако программа некорректно отрабатывает повторные нажатия на кнопку Пуск — движение объекта после нажатия кнопки ускоряется вдвое.

Продолжение примера 31.

Корректную отработку нажатия на кнопку Пуск организуем с помощью глобальной переменной-флага flag. Добавляем переменную

```
var flag=true;
```

Тело функции **func1()** записываем в виде

```
if (flag) sf3=setInterval("func3()",20);
flag=false;
```

Теперь при запуске движения флаг выключается (значение **false**), и повторно сделать запуск невозможно. А в конец тела функции **func2()** дописываем команду

```
flag=true;
```

чтобы восстановить значение флага, когда движения выключается.

Сохраняем текст файла **pos30.html** и запускаем его.

Скорость движения объекта регулируется размером сдвига dx и временем задержки повторения (второй аргумент функции **setInterval()**). Чтобы движение объекта было плавным, величина задержки повторения не должна превышать 40 миллисекунд (40 = 1/25 сек или 25 кадров в сек, как в современном кинематографе).

Задания:

1). Измените текст программы решения примера 31 так, чтобы объект двигался по вертикали (ввести **dy=10**, все команды функции **func3()** перевести в комментарии и построить аналогичные команды для **ntop** и **...style.top**).

2). Измените текст программы решения задания 1 так, чтобы объект начинал двигаться по прямой **dx=dy** (вправо-вниз под углом 45°), а отражался от краев прямоугольника по схеме "угол падения равен углу отражения" (достаточно убрать знаки комментария).

(!!! Задание 2 после выполнения показать преподавателю!!! 19)

3). Измените текст программы решения задания 2 так, чтобы объект начинал двигаться по прямой **dx=2dy**.

13. Автоматическое криволинейное движение объектов

Пример 32.

В левом верхнем углу окна браузера зеленого цвета заданы две кнопки Пуск и Стоп, а ближе к центру — объект-картинка. Нажатие кнопки Пуск запускает движение объекта по замкнутой кривой. Нажатие кнопки Стоп останавливает движение.

Изменим текст файла **pos31.html**. Меняем номер примера и сохраняем файл с именем **pos32.html**. В тексте файла удаляем блок ** **, а движущуюся картинку **<img>**, что внутри него, оставляем.

Движение картинки по экрану обычно организуется с помощью уравнений, которые вычисляют координаты картинки по значениям некоторого параметра. Параметр меняется, и, соответственно, меняются координаты.

В предыдущем примере картинка двигалась по горизонтали, и фактически уравнениями движения были уравнения $x = t$ и $y = 130$, где t — отступ левого края картинки от левого края прямоугольника в пикселах, а 130 тоже, конечно, пикселов.

В разбираемом примере движение картинки организуем по замкнутой кривой, которая называется эллипсом и задается параметрическими уравнениями:

$$x(t)=Xc + Rx \cos(t),$$
$$y(t)=Yc + Ry \sin(t).$$

где t — параметр;

(Xc, Yc) — координаты центра вращения картинки;

Rx — наибольшее удаление по оси OX;

Ry — наибольшее удаление по оси OY.

Заметим, что если задать $Rx = Ry$, получается окружность.

В разделе программы на языке JavaScript удаляем объявления переменных

```
var dx=10;
var dy=10;
```

и вводим переменные, необходимые для нашего примера.

```
var Xc=400;
var Yc=250;
var Rx=200;
var Ry=100;
```

Изменяемый параметр t в формулах является аргументом тригонометрических функций, которые имеют период 360° или 2π (в радианах). При расчетах на языке JavaScript параметр удобно задавать в радианах, а для значений параметра использовать отрезок [0, 2π]. Начальное значение переменной-параметра **t** также зададим этих же пределах, например:

```
var t=1;
```

Параметр будет каждый раз изменяться на некоторую величину *dt* (приращение, сдвиг). Задаем и эту величину.

```
var dt=0.05;
```

Переменную flag и первые две функции в тексте оставляем — они связаны с запуском и остановкой движения. Как задано в теле функции **func1()**, параметр **t** будет меняться каждые 20 миллисекунд.

А вот тело функции **func3()** очищаем. В ней надо задать положение картинки. Значения свойств

```
document.getElementById('ball').style.left
document.getElementById('ball').style.top
```

по определению задают положение левого верхнего угла картинки.

Чтобы они задавали положение центра картинки, вычисленные по формулам значения нужно уменьшить на 20px (1/2 от 40px – фактической высоты и ширины картинки):

```
document.getElementById('ball').style.left=Xc+Rx*Math.cos(t)-20;
document.getElementById('ball').style.top=Yc+Ry*Math.sin(t)-20;
```

Далее организуем изменение параметра **t** на величину **dt**

```
t+=dt;
```

Приращение **dt** может быть и отрицательным (движение в обратную сторону), поэтому, чтобы параметр **t** постоянно принадлежал промежутку [0, 2π], дописываем двойную конструкцию, которая все значения **t**, выходящие за пределы [0, 2π], возвращает в этот промежуток.

```
if (t>2*Math.PI) t=0;
if (t<0) t=2*Math.PI;
```

Чтобы заданное начальное значение параметра **t** сразу вступило в силу, функцию **func3()** нужно запустить один раз после загрузки текста. Для этого в теге **<body>** дописываем команду для атрибута

```
onload="func3()"
```

Сохраняем текст файла **pos32.html** и запускаем его.

Если поменять знак приращения **dt**, то поменяется направление движения объекта.

Пример 33.

В условиях примера 32 запустить на экране два объекта по замкнутой кривой одного и того же типа (эллипс), но с разными исходными данными.

Изменим текст файла **pos32.html**. Меняем номер примера и сохраняем файл с именем **pos33.html**. В тексте копированием повторяем тег **<img>** и в новом теге меняем идентификатор на "**ball2**".

В разделе программы на языке JavaScript первым шести переменным (с исходными данными) в имя добавляем символ "**1**". Затем копированием повторяем эти шесть строк, в новых строках меняем в именах "**1**" на "**2**", а значения уменьшаем ровно в 2 раза. В значении **dt2** меняем и знак.

Функцию **func3()** оставляем для организации движения обоих объектов. Изменим ее позднее.

А теперь функцию **func3()** копированием повторяем два раза, называем копии **func4()** и **func5()**. Будем использовать эти функции для формул движения картинок.

В теле функции **func4()** изменяем имена переменных на имена для первой картинки, а в теле функции **func5()** — для второй

Внимание! Изменяя имена, не пропустите какое-то имя!

Теперь тело функции **func3()** очищаем и записываем команды вызова наших функций:

```
func4();
func5();
```

Сохраняем текст файла **pos33.html** и запускаем его. Аналогично можно запускать любое число движущихся картинок, причем по разным кривым (каждому объекту своя функция).

Пример 34.

Поскольку объекты в примере 33 движутся по кривой одного и того же типа, при построении программы можно пойти другим, более экономным путем.

Опять изменим текст файла **pos32.html**. Меняем номер примера и сохраняем файл с именем **pos34.html**. Опять в тексте копированием повторяем тег **<img>** и в

новом теге меняем идентификатор на **"ball2"**. Исходные данные (по 6 на каждый объект) можно скопировать из файла **pos33.html**.

Функцию **func3()** копированием повторим, копию переименуем в **func4()**. С функцией **func3()** разберемся потом.

Функцию **func4()** сделаем универсальной для вычислений по любым исходным данным для кривых этого типа.

Аргументами функции **func4()** будут 6 исходных данных, которые в ней и используются, плюс идентификатор картинки, пусть **ide**. Т.е. первую строку функции запишем в виде

```
function func4(ide,Xc,Yc,Rx,Ry,t,dt) {
```

Далее изменения коснутся только первых двух команд тела функции **func4()**, в которых надо конкретное значение идентификатора **"ball"** поменять на имя переменной **ide** (без кавычек).

Чтобы сохранить вычисленные значения параметров **t1** и **t2**, в конце тела функции **func4()** запишем команду

```
return t;
```

которая присвоит вычисленное значение параметра имени функции **func4()**.

Теперь очистим тело функции **func3()** и запишем туда две команды вызова функции **func4()** с разными наборами входных данных. Команды одновременно сохраняют новые вычисленные значения параметров:

```
t1=func4("ball",Xc1,Yc1,Rx1,Ry1,t1,dt1);
t2=func4("ball2",Xc2,Yc2,Rx2,Ry2,t2,dt2);
```

Сохраняем текст файла **pos34.html** и запускаем его.

Задания:

1). В условиях примера 32 в центр вращения подвижной картинки установите центр еще одной картинки (неподвижной).

2). В условиях примера 32 запустите картинку по кардиоиде:

$x = Xc + Rx(1+\cos t)\cos t;\ \ y = Yc + Rx(1+\cos t)\sin t.$

(!!! Задание 2 после выполнения показать преподавателю!!! 20)

3). В условиях примера 33 запустите одну картинку по кардиоиде, а вторую — по астроиде:

$x = Xc + Rx\cos^3 t;\ \ y = Yc + Ry\sin^3 t.$

Одноименные исходные данные для объектов должны совпадать, а $Rx = Ry$.

4). Сделайте в тексте программы примера 34 исходные данные у объектов равными, кроме **dt1** и **dt2**, которые должны различаться только знаками.

5). Измените текст программы примера 34 так, чтобы вращались три объекта с равными исходными данными, за исключением **Xc1=200**, **Xc2=400**, **Xc3=600**.