УВАЖАЕМЫЕ СТУДЕНТЫ! Законспектируйте в своей рабочей тетради по дисциплине приведенную лекцию (объемом 4-5 страницы), ответьте *письменно* на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: igor-gricenko-95@mail.ru **в течении ТРЕХ дней.**

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)132-63-42

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция 24-25
События в JavaScript

События и обработчики событий являются очень важной частью для программирования на языке JavaScript. События, главным образом, инициируются теми или иными действиями пользователя. Если он щелкает по некоторой кнопке, происходит событие *"Click"*. Если указатель мыши пересекает какую-либо ссылку гипертекста - происходит событие *MouseOver*. Существует несколько различных типов событий.
Мы можем заставить нашу JavaScript-программу реагировать на некоторые из них. И это может быть выполнено с помощью специальных программ обработки событий. Так, в результате щелчка по кнопке может создаваться выпадающее окно. Это означает, что создание окна должно быть реакцией на событие щелка - *Click*. Программа - обработчик событий, которую мы должны использовать в данном случае, называется *onClick*. И она сообщает компьютеру, что нужно делать, если произойдет данное событие. Приведенный ниже код представляет простой пример программы обработки события *onClick*:

```
<form>
<input type="button" value="Нажми меня" onClick="alert('Ого')">
</form>
```

Функция *alert()* позволяет Вам создавать выпадающие окна. При ее вызове Вы должны в скобках задать некую строку. В нашем случае это *'Ого'*. И это как раз будет тот текст, что появится в выпадающем окне. Таким образом, когда читатель щелкает на кнопке, наш скрипт создает окно, содержащее текст *'Ого'*.

Иерархия объектов в JavaScript

В языке JavaScript все элементы на web-странице выстраиваются в иерархическую структуру. Каждый элемент предстает в виде объекта. И каждый такой объект может иметь определенные свойства и методы. В свою

очередь, язык JavaScript позволит Вам легко управлять объектами web-страницы, хотя для этого очень важно понимать иерархию объектов, на которые опирается разметка HTML. Как это все действует, Вы сможете быстро понять на следующем примере. Рассмотрим простую HTML-страницу:

```
<html>
<head>
<title>My homepage
</head>
<body bgcolor=#ffffff>

<center>
<img src="home.gif" name="pic1" width=200 height=100>
</center>

<p>
<form name="myForm">
<input type="text" name="name" value=""><br>
<input type="text" name="email" value=""><br><br>
<input type="button" value="Push me" name="myButton" onClick="alert('Yo')">
</form>
<p>

<center>
<img src="ruler.gif" name="pic4" width=300 height=15>
<p>

<a href="http://rummelplatz.uni-mannheim.de/~skoch/">My homepage</a>
</center>

</body>
</html>
```

Итак, мы имеем два рисунка, одну ссылку и некую форму с двумя полями для ввода текста и одной кнопкой. С точки зрения языка JavaScript окно браузера - это некий объект *window*. Этот объект также содержит в свою очередь некоторые элементы оформления, такие как строка состояния. Внутри окна мы можем разместить документ HTML (или файл какого-либо другого типа). Такая страница является ни чем иным, как объектом *document*. Это означает, что объект document представляет в языке JavaScript загруженный на настоящий момент документ HTML. Объект document является очень важным объектом в языке JavaScript и Вы будете пользоваться им многократно. К свойствам объекта document относятся, например, цвет фона для web-страницы. Однако для нас гораздо важнее то, что все без исключения объекты HTML являются свойствами объекта

document. Примерами объекта HTML являются, к примеру, ссылка или заполняемая форма.
Разумеется, мы должны иметь возможность получать информацию о различных объектах в этой иерархии и управлять ею. Для этого мы должны знать, как в языке JavaScript организован доступ к различным объектам. Как видно, каждый объект иерархической структуры имеет свое имя. Следовательно, если Вы хотите узнать, как можно обратиться к первому рисунку на нашей HTML-странице, то обязаны сориентироваться в иерархии объектов. И начать нужно с самой вершины. Первый объект такой структуры называется *document*. Первый рисунок на странице представлен как объект *images[0]*. Это означает, что отныне мы можем получать доступ к этому объекту, записав в JavaScript *document.images[0]*.
Если же, например, Вы хотите знать, какой текст ввел читатель в первый элемент формы, то сперва должны выяснить, как получить доступ к этому объекту. И снова начинаем мы с вершины нашей иерархии объектов. Затем прослеживаем путь к объекту с именем *elements[0]* и последовательно записываем названия всех объектов, которые минуем. В итоге выясняется, что доступ к первому полю для ввода текста можно получить, записав:
document.forms[0].elements[0]
А теперь как узнать текст, введенный читателем? Чтобы выяснять, которое из свойств и методов объекта позволят получить доступ к этой информации, необходимо обратиться к какому-либо справочнику по JavaScript (например, это может быть документация). Там Вы найдете, что элемент, соответствующий полю для ввода текста, имеет свойство *value*, которое как раз и соответствует введенному тексту. Итак, теперь мы имеем все необходимое, чтобы прочитать искомое значение. Для этого нужно написать на языке JavaScript строку:
name = document.forms[0].elements[0].value;

Полученная строка заносится в переменную *name*. Следовательно, теперь мы можем работать с этой переменной, как нам необходимо. Например, мы можем создать выпадающее окно, воспользовавшись командой `alert ("Hi " + name)`. В результате, если читатель введет в это поле слово *'Stefan'*, то по команде `alert("Hi " + name)` будет открыто выпадающее окно с приветствием *'Hi Stefan'*.
Если Вы имеете дело с большими страницами, то процедура адресации к различным объектам по номеру может стать весьма запутанной. Например, придется решать, как следует обратиться к объекту
document.forms[3].elements[17] document.forms[2].elements[18]?
Во избежание подобной проблемы, Вы можете сами присваивать различным объектам уникальные имена. Как это делается, можно увидеть в примере:

```
<form name="myForm">
<input type="text" name="name" value=""><br>
...
```

Эта запись означает, что объект *forms[0]* получает теперь еще и второе имя - *myForm*. Точно так же вместо *elements[0]* Вы можете писать *name* (последнее было указано в атрибуте name тэга <input>). Таким образом, вместо

name= document.forms[0].elements[0].value;

Вы можете записать

name= document.myForm.name.value;

Это значительно упрощает программирование на JavaScript, особенно в случае с большими web-страницами, содержащими множество объектов. (Обратите внимание, что при написании имен Вы должны еще следить и за положением регистра - то есть Вы не имеете права написать *myform* вместо *myForm*)
В JavaScript многие свойства объектов доступны не только для чтения. Вы также имеете возможность записывать в них новые значения. Например, посредством JavaScript Вы можете записать в упоминавшееся поле новую строку.
Пример кода на JavaScript, иллюстрирующего такую возможность - интересующий нас фрагмент записан как свойство onClick второго тэга <input>:

```
<form name="myForm">
<input type="text" name="input" value="bla bla bla">
<input type="button" value="Write"
  onClick="document.myForm.input.value= 'Yo!'; ">
```

Рассмотрим следующий пример:
Исходный код скрипта:

```
<html>
<head>
<title>Objects</title>

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

function first() {

// создает выпадающее окно, где размещается
// текст, введенный в поле формы

  alert("The value of the textelement is: " +
    document.myForm.myText.value);
}
```

```
function second() {

// данная функция проверяет состояние переключателей

  var myString= "The checkbox is ";

// переключатель включен, или нет?
  if (document.myForm.myCheckbox.checked) myString+= "checked"
    else myString+= "not checked";

// вывод строки на экран
  alert(myString);
}

// -->
</script>
</head>
<body bgcolor=lightblue>

<form name="myForm">
<input type="text" name="myText" value="bla bla bla">
<input type="button" name="button1" value="Button 1"
  onClick="first()">
<br>
<input type="checkbox" name="myCheckbox" CHECKED>
<input type="button" name="button2" value="Button 2"
  onClick="second()">
</form>

<p><br><br>

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

document.write("The background color is: ");
document.write(document.bgColor + "<br>");

document.write("The text on the second button is: ");
document.write(document.myForm.button2.value);

// -->
</script>

</body>
```

</html>

Демонстрационный пример. Самый продуктивный способ получить представление о языке - это разбор примеров. Если посмотреть на список этих примеров, который можно найти в соответствующей директории Yahoo, легко убедиться, что больше всего там различного сорта калькуляторов. Разберем программу такого же типа, только это будет не калькулятор, а программа обучения устному счету.
Приведенный пример содержит датчик случайных чисел (функции **init** и **rand**), таблицу, реализующую функции кнопок клавиатуры, и блок проверки результата вычислений. После загрузки программы пользователь должен выбрать тип вычислений (+,-), интервал вычислений (в пределах 10, 20, 100) и нажать кнопку "?" для генерации примера. После ввода числа с отображаемой клавиатуры пользователь нажимает на символ "=", что означает исполнить, и система проверяет правильность ответа. Если ответ правильный, то программа выдает "Молодец!", если нет - "Думай!". В системе Windows 3.x нет встроенного датчика случайных чисел, поэтому стандартная функция rand в этой версии JavaScript не реализована. Используемый в данной программе датчик был позаимствован из телеконференции по JavaScript. В скрипте, кроме этого, применяются объект типа "дата" и его методы, а также встроенные функции контроля вводимых данных. Как явствует из примера, обращение к полям HTML-формы представляет собой обращение к структуре, корнем которой является объект "окно", в котором определен объект "документ", внутри которого определена форма, а также ее поля и атрибуты полей. Не у всех полей можно менять значения атрибутов, так, например, атрибут **VALUE** в кнопке не меняет своего значения, если только не перезагрузить страницу.

```
<HTML>
<!- Author: Pavel Khramtsov
Date: February 23, 1996
URL: http://144.206.192.100/radleg/ mo_input.htm
->

<HEAD>
<TITLE>Проверка устного счета.</TITLE>

<SCRIPT LANGUGE="JavaScript">

var max_value = 0; var operand1 = 0
var operand2 = 0; var result = 0
var flag = 0; var sign = "+"
var input = ""; var v_date = new Date()
var number = v_date.getTime()

function init(factor){
```

```
var today = new Date()
return (factor*today.getTime()/1000)%30000
}

//Инициализировать датчик
//случайных чисел

IX = init(2); IY = init(3); IZ = init(4)

// 2, 3, и 4 были произвольно выбраны.
// Они устанавливают некоторое расстояние между
// начальные ценности IX, IY, и IZ

//Датчик случайных чисел.
function random(){
IX = Math.floor(171 * IX % 30269)
IY = Math.floor(172 * IY % 30307)
IZ = Math.floor(170 * IZ % 30323)
return ( IX/30269.0 + IY/30307.0+ IZ/30323.0 ) % 1.0 }

//Выбрать сложение или вычитание
function set_sign(x){
if( x == "+"){
flag = 0; sign = "+"
}
if( x == "-"){
flag = 1; sign = "-"
}
return true
}

// Определить первый операнд
function f_operand(){
operand1 = random()*max_value
return parseInt(operand1)
}

// Определить второй операнд
function s_operand(){
if(flag == 0){
operand2 = random()* (max_value-operand1)
}
else {
operand2 = random()*operand1
}
```

```
return parseInt(operand2)
}

// Проверить введенные данные
function input_sign(x){
if(x == 10){
if(flag == 0){
if(operand1+operand2 ==parseInt(input)){
window.document.test. r0.value = "Молодец!"
}
else {
window.document.test.r0.value = "Думай!?"
window.document.test.input = ""; input = ""
}
}
if(flag == 1){
if(operand1-operand2 ==parseInt(input)){
window.document.test.r0.value ="Li-ii-ip!"
}
else{
window.document.test.r0.value = "Думай!?"; window.document.test.input = "";
input = ""
}
}
return true
}
input += x
window.document.test.result.value = input
return true
}

// Генерация варианта
function main_calc(){
operand1 = f_operand()
window.document.test.op1.value = operand1
operand2 = s_operand()
window.document.test.op2.value = operand2
window.document.test.s_sign.value = sign
input = ""; window.document.test.input = ""
window.document.test.r0.value = " ???"
return true
}

//
</SCRIPT>
```

```
</HEAD>

<BODY>
<CENTER>
<H1>Математический тест</H1>
<HR>
<FORM NAME=test>
<TABLE BORDER=0>
<TR>
<TD><INPUT NAME=i10 TYPE=button VALUE="0-10"
onClick="max_value=10"></TD>
<TD><INPUT NAME=i20 TYPE=button VALUE="0-20"
onClick="max_value=20"></TD>
<TD><INPUT NAME=i100 TYPE=button VALUE="0-100"
onClick="max_value=100"></TD>
<TD><INPUT NAME=i+ TYPE=button VALUE="+"
onClick="set_sign('+')"></TD>
<TD><INPUT NAME=i- TYPE=button VALUE="-" onClick="set_sign('-
')"></TD>
</TR>
</TABLE>
<HR>
<TABLE BORDER=0>
<TR>
<TD><INPUT NAME=op1 SIZE=2 MAXLENGTH=2></TD>
<TD><INPUT NAME=s_sign SIZE=1 MAXLENGTH=1></TD>
<TD><INPUT NAME=op2 SIZE=2 MAXLENGTH=2></TD>
<TD>=</TD>
<TD><INPUT NAME=result SIZE=3 MAXLENGTH=3></TD>
<TD><INPUT NAME=award TYPE=button VALUE="?"
onClick="main_calc()">
<TD><INPUT NAME=r0 VALUE="???">
</TR>
</TABLE>
<HR>
<TABLE BORDER=2>
<TR>
<TD><INPUT NAME=b1 TYPE=button VALUE="1"
onClick="input_sign('1')"></TD>
<TD><INPUT NAME=b2 TYPE=button VALUE="2"
onClick="input_sign('2')"></TD>
<TD><INPUT NAME=b3 TYPE=button VALUE="3"
onClick="input_sign('3')"></TD>
</TR>
<TR>
```

```
<TD><INPUT NAME=b4 TYPE=button VALUE="4"
onClick="input_sign('4')"></TD>
<TD><INPUT NAME=b5 TYPE=button VALUE="5"
onClick="input_sign('5')"></TD>
<TD><INPUT NAME=b6 TYPE=button VALUE="6"
onClick="input_sign('6')"></TD>
</TR>
<TR>
<TD><INPUT NAME=b7 TYPE=button VALUE="7"
onClick="input_sign('7')"></TD>
<TD><INPUT NAME=b8 TYPE=button VALUE="8"
onClick="input_sign('8')"></TD>
<TD><INPUT NAME=b9 TYPE=button VALUE="9"
onClick="input_sign('9')"></TD>
</TR>
<TR>
<TD><INPUT NAME=b0 TYPE=button VALUE="0"
onClick="input_sign('0')"></TD>
<TD COLSPAN=2><INPUT NAME=bs TYPE=button VALUE="OK"
onClick="input_sign('10')"></TD>
</TR>
</TABLE>
</FORM>
</CENTER>
<HR>
</BODY>
</HTML>
```