Язык JavaScript.

Занятие №1

Общие сведения

JavaScript (JS) был разработан совместно компаниями Sun Microsystems и Netscape. За синтаксическую основу нового языка был взят язык Java, в свое время разработанный компанией Sun Microsystems. Следует понимать, что программы, написанные на языке Java (так называемые Java-applet'ы) не имеют ничего общего с JS.

Программы Javascript, встраиваются непосредственно в HTML-страницы. Браузер (например, Internet Explorer), встречая такую программу, *интерпретируют* ее, т.е. в отличие, например, от *компилируемого* Turbo Pascal, двоичный код (exe) не создается. Программы, написанные на интерпретируемых языках, обычно называют *скриптами*. Скрипты могут находиться в любом месте HTML-документа. Главная часть JS программы может быть помещена в контейнер <head>... </head>, поскольку он считывается при загрузке HTML -документа одним из первых. JS программа помещается между тегами <script> ... </script>, исключение составляют обработчики событий. Встретив тег <script>, браузер построчно анализирует содержимое документа до тех пор, пока не будет достигнут тег </script>. После этого производится проверка скрипта на наличие ошибок и выполнение JS программы.

Программа на JS может не образовывать некого единого целого (как программа в Pascal), а состоит из отдельных фрагментов, решающих небольшие задачи.

JS – объектно-ориентированный язык. Это – новое понятие. Полный смысл его будет раскрыт далее. Основные объекты в JS – это window, document и navigator. Они отвечают за поведение окна браузера, документа и самого браузера соответственно.

Важным свойством, присущим последним версиям JS и HTML 4.0 (так называемый динамический HTML – DHTML), является возможность взаимодействия и динамического изменения таблиц стилей (CSS). Благодаря этому обстоятельству удается достигнуть различных динамических визуальных эффектов.

JS – событийно-ориентированный язык. С помощью специальных параметров, включаемых в тэги HTML (onMouseOver, onLoad и т.п.), задаются действия, которые необходимо выполнить при возникновении определенных событий: действий пользователя, открытия документа и т.п.

Итак, программа на JS включается в HTML-страницу следующим образом:

```
<script language="JavaScript">
    [текст программы]
</script>
```

Следует иметь в виду, что слово "JavaScript" записывается с соблюдением регистра символов. Кроме того, в отличие от HTML и Pascal, синтаксис самого JS чувствителен к регистру букв.

Пример №1:

Попробуйте в любом месте внутри тэга <body> вставить такую конструкцию:

```
<script language="JavaScript">
    document.write('<H1>Это строка генерируется JS</H1>');
</script>
```

Обратите внимание, что document – это имя объекта, соответствующего html-тексту, загруженному в браузер, а write – имя метода объекта document. Метод – это определенная подпрограмма, меняющая некоторые свойства объекта. В синтаксисе языка можно использовать кавычки или апострофы – они равнозначны, только должны быть парными.

Пример №2:

Попробуйте изменить предыдущий пример следующим образом:

```
<script language="JavaScript">
   for (i=0; i<10; i++)
    {
      document.write('<H1>Это строка генерируется JS</H1>');
    }
</script>
```

В этом примере используется цикл for, его синтаксис немного отличается от Pascal:

for (начальное значение переменной; условие продолжения; приращение переменной)
{
  тело цикла
}

Описание переменных в JavaScript является необязательным. Оператор присваивания обозначается просто знаком =. Вместо begin и end используются фигурные скобки { }. В этом примере их можно было бы и опустить, так как в теле цикла находится всего один оператор. Оператор i++ обозначает увеличение значения переменной i на единицу.

Пример №3:

Сделаем еще изменения:

```
<script language="JavaScript">
   for (i=6; i>0; i--)
    {
      document.writeln('<H',i,'>Это строка генерируется JS</H',i,'>');
    }
</script>
```

Посмотрите на результат самостоятельно, проанализируйте его и сопоставьте с текстом.

Операции присваивания

В языке JS имеется несколько вариантов присваивания:

=	Прямое присваивание значения левому операнду
+=	Складывает значения левого и правого операндов и присваивает результат левому операнду
+	Складывает значения левого и правого операндов и присваивает результат левому операнду
++	Увеличивает значение левого операнда (правый может отсутствовать)
-=	Вычитает значения левого и правого операндов и присваивает результат левому операнду
-	Вычитает значения левого и правого операндов и присваивает результат левому операнду
--	Уменьшает значение левого операнда (правый может отсутствовать)
*	Умножает значения левого и правого операндов и присваивает результат левому операнду
*=	Умножает значения левого и правого операндов и присваивает результат левому операнду
/	Делит значения левого на правого операндов и присваивает результат левому операнду
/=	Делит значения левого на правого операндов и присваивает результат левому операнду

Так, например, можно записать вместо:

```
nval = nval * 10;
```

более компактно

```
nval *= 10;
```

т.е. переменная nval увеличивает значение в 10 раз.

Операторы комментариев и примечаний

Если надо поставить комментарий к какому-нибудь одному оператору, то это можно сделать так:

```
nval *= 10; // текст комментария
```

А если комментарий состоит из нескольких строк, то надо пользоваться такой конструкцией:

```
/*    Многострочный
      комментарий
*/
```

Многострочный комментарий удобен для временного выключения части JS-текста из программы. Это очень удобно при отладке скриптов.

Упражнение №1
Переделайте пример №3 так, чтобы заголовки выводились в возрастающем порядке.

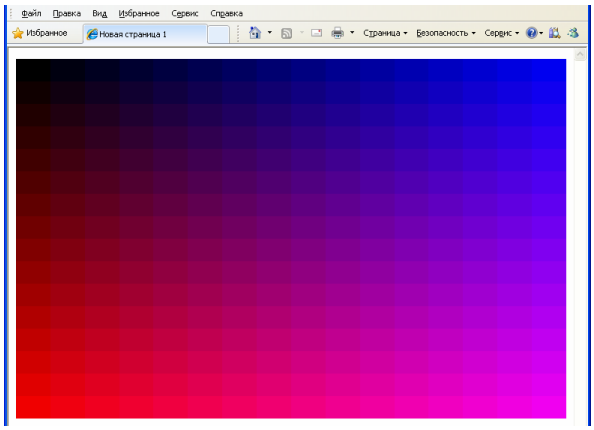
Упражнение №2
Напишите программу на JavaScript, выводящую в окне IE таблицу квадратов целых чисел от 1 до 10.

Упражнение №3*
Напишите программу на JavaScript, выводящую в окне IE таблицу умножения целых числе от 1 до 5 (здесь потребуется вложенный цикл for).

Упражнение №4**
Напишите программу на JavaScript, строящую таблицу 10×10 клеток. Информацию в клетках можно выбрать произвольно (например, та же таблица умножения).

Упражнение №5**
Напишите программу на JavaScript, строящую таблицу 16×16 клеток, такую чтобы она демонстрировала палитру смешения двух основных цветов в разных пропорциях. Для этого воспользуйтесь заданием фона ячейки вида `<TD BGCOLOR=rgb(255,0,0)> </TD>`. Ваша задача – изменять в циклах значения цветов в RGB-выражении, т.е. чтобы таблица строилась таким образом:

```
<TABLE>
 <TR><TD BGCOLOR=rgb(0, 0, 0)> </TD><TD BGCOLOR=rgb(16, 0, 0)> </TD> … </TR>
 <TR><TD BGCOLOR=rgb(0, 0,16)> </TD><TD BGCOLOR=rgb(16, 0,16)> </TD> … </TR>
 <TR><TD BGCOLOR=rgb(0, 0,32)> </TD><TD BGCOLOR=rgb(16, 0,32)> </TD> … </TR>
 …
</TABLE>
```



При правильной программе вы увидите что-то подобное. К сожалению, конструкцию RGB «понимает» только Internet Explorer. В данном примере выбран шаг 16 увеличения цветовой компоненты, зеленая компонента полагается равной нулю. Можно выбрать и другие параметры.

Занятие №2

Операторы циклов

Кроме цикла for, описанного в занятии №1, как и в Pascal существует цикл while

Цикл while

Синтаксис:

```
while (условие) {
программный блок
}
```

При помощи оператора **while** можно выполнять один или несколько операторов до тех пор, пока не будет удовлетворено условие. Если в теле цикла выполняется несколько операторов, их необходимо заключить в фигурные скобки.

Пример №4:

В данном примере можно было бы использовать цикл for, но в будущем мы встретимся с ситуациями, где без while не обойтись:

```
<script language="JavaScript">
   i=0;
   while (i<10)
   {
     document.writeln(i,'<SUP>2</SUP>=',i*i,'<BR>');
     i++;
   }
</script>
```

Обратите внимание на '
' в конце оператора writeln. Так как возникающий при исполнении скрипта html-текст будет интерпретироваться браузером, то необходимо включать все желаемые тэги форматирования. Уберите
 и посмотрите, что получиться.

Пример №5:

Циклы могут быть вложенными друг в друга. Проанализируйте следующий пример:

```
<TABLE WIDTH=100%>
  <script language="JavaScript">
     for (i=0; i<255; i+=16)
     { document.write('<TR>');
       for (j=0; j<255; j+=16)
         document.write('<TD BGCOLOR=rgb(0,',i,',',j,')> </TD>');
       document.writeln('</TR>');
     }
  </script>
</TABLE>
```

Условные операторы

Оператор - if . . . else

Синтаксис:
if (condition); {
Программный блок1
} [else { программный блок2 }]

Оператор **if . . . else** - это условный оператор, который обеспечивает выполнение одного или нескольких операторов, в зависимости от того, удовлетворяются ли условия. Часть *condition* оператора **if** является выражением, при истинности которого выполняются операторы языка в первом программном блоке. Программный блок должен быть заключен в фигурные скобки, однако если используется только один оператор, можно скобки не ставить. Необязательная часть **else** обеспечивает выполнение операторов второго блока, в случае, если условие *condition* оператора **if** является ложным. Операторы **if** можно вкладывать друг в друга.

Пример №6:

Преобразуем пример №2:

```
<script language="JavaScript">
   k=0
   for (i=0; i<10; i++)
     if (k==0)
        {document.write('<H1 STYLE="color: red">Красный</H1>'); k=1;}
      else
        {document.write('<H1 STYLE="color: green">Зеленый</H1>'); k=0;};
 </script>
```

Операции сравнения

= =	Равенство (равно)
!=	Не равно
!	Логическое отрицание
>=	Больше или равно
<=	Меньше или равно
>	Больше
<	Меньше

Логические операции

Для обозначения логической операции И в языке JS используют два символа амперсанта (&&), а для обозначения логической операции ИЛИ - два символа прямой черты (||). Эти операции применимы только к булевым значениям. Например:

```
if ( k==0 && i<5 )
```

Условные выражения

Условные выражения используются для сравнения одних переменных с другими, а также с константами или значениями, возвращаемыми различными выражениями. В языке JS есть оператор сравнения **?**, имеющий синтаксис:

(условное выражение) ? операторы_1 : операторы_2

Если условное выражение истинно - выполнются *операторы_1*, в противном случае *операторы_2*. Можно также присваивать значения переменным. Например, оператор:

```
type_time = (hour >= 12) ? "PM" : "AM"
```

присваивает строковое значение "PM" переменной type_time, если значение переменной hour больше или равно 12; в противном случае присваивается "AM".

Оператор **?** в действительности является сокращенным вариантом оператора *if . . . else*. Предыдущий пример может быть записан так:

```
if (hour >= 12)
  type_time="PM";
else
  type_time="AM";
```

Выход из цикла – оператор break

Оператор **break** используется для выхода из какого-либо цикла, например из цикла *for* или *while*. Выполнение цикла прекращается в той точке, в которой размещен этот оператор, а управление передается следующему оператору, находящемуся непосредственно после цикла.

Пример:

```
i=0;
while (true) // бесконечный цикл
 {
   if ( (i++) == 10 ) break;
 }
```

Продолжение цикла – оператор continue

Оператор **continue** используется для прерывания выполнения блока операторов, которые составляют тело цикла и продолжения цикла в следующей итерации. В отличие от оператора *break*, оператор *continue* не останавливает выполнение цикла, а наоборот запускает новую итерацию. Если в цикле *while* идет просто запуск новой итерации, то в циклах *for* запускает с обновленным шагом.

Упражнение № 5

Попробуйте модифицировать пример №5 (в строну сильного упрощения), так чтобы на экране появилась шахматная доска, т.е. клетки были бы белые и черные. Высоту строки таблицы можно задать с помощью параметра HEIGHT в тэге <TR>.Понадобится оператор *if* внутри цикла *for*.

Упражнение № 6

Напишите программу, которая выводила бы 80 строчек таким образом, чтобы в первая строчка состояла из одной буквы A, вторая из двух букв A, и т.д.

Занятие №3

Функции

В Javascript, как и в Pascal, существуют подпрограммы, и только один вид – функции. Использование функций – широко распространенный прием. Для того, чтобы к моменту использования функции, она была уже загружена из Internet, описание функций помещают в заголовок документа (в секцию <HEAD> ... </HEAD>), хотя это и не является строго обязательным.

Аргументы в функцию передаются только по значению, т.е. информацию из функции можно получить только через ее значение или через глобальные переменные, но не через параметры. Через параметры можно только передать информацию в функцию. Функция может вообще не возвращать значение (тогда она будет аналогом подпрограммы-процедуры в Pascal).

Пример №7. (полный текст HTML-страницы)

```
<HTML>
<HEAD>
   <SCRIPT>
    function h(s)
    {
      return '<H1 ALIGN=CENTER>'+s+'</H1>';
    }
   </SCRIPT>
</HEAD>

<BODY>
<H1 ALIGN=CENTER>Пример №7</H1>
 <SCRIPT>
  document.writeln( h("Эта строка преобразуется в заголовок") );
 </SCRIPT>
</BODY>
</HTML>
```

Обратите внимание, что в отличие от Pascal возвращаемое значение указывается в операторе *return*. Заметьте, что операция сложения (конкатенации) строк записывается точно также как в Pascal. В качестве ограничителя текстовых строк можно использовать либо апострофы, либо кавычки.

Функция может иметь свои собственные переменные, называемые локальными переменными.

Пример №8.

```
<HTML>
<HEAD>
   <SCRIPT Language="Javascript">
   function z(s,n)
   { var i;
     var q='';
     for (i=0; i<n; i++) q=q+s;
     return q;
   }
  </SCRIPT>
 </HEAD>

     <BODY>
     <H1 ALIGN=CENTER>Пример №8</H1>
     <HR>
      <SCRIPT>
       document.writeln( z("Javascript ",10) );
      </SCRIPT>
     <HR>
     </BODY>
</HTML>
```

В этом примере оператор *var i* вводит локальную переменную i, не задавая никакой дополнительной информации. Ее тип и значение определяются автоматически в цикле for, в котором ей присваивается целое значение. Оператор *var q=''* представляет символьную переменную с пустым значением (после знака равенства не кавычки, а два апострофа). Так можно и желательно делать. Тело цикла for состоит всего лишь из одного оператора, поэтому фигурные скобки опущены.

Важное замечание: В принципе можно вообще было бы не описывать переменные i и q, а ввести их просто операторами присваивания. В этом случае браузер завел бы две **глобальных** (т.е. определенных во всем тексте программы) переменных. Это может иногда привести к нежелательным последствиям, если бы, например, переменная i использовалась в другом месте. Т.е. функция изменила бы значение этой переменной. Однако, в некоторых случаях использование глобальных переменных –

единственная возможность передачи информации (особенно большой по объему) из функции в вызывающую программу.

Пример №9.

```
<HTML>
<HEAD>
   <SCRIPT Language="Javascript">

    var A="Функция не вызвана";

   function d()
   {
    A="Функция вызвана";
   }
   </SCRIPT>
</HEAD>
    <BODY>

    <H1 ALIGN=CENTER>Пример №9</H1>
    <HR>
     <SCRIPT>
      document.writeln(A,'<BR>');
      d();
      document.writeln(A);
     </SCRIPT>
    <HR>

    </BODY>
</HTML>
```

В самом начале определяется глобальная переменная A явным образом. В принципе, можно *var* опустить, т.к. глобальная переменная заводится автоматически. Функция d не имеет параметров, тем не менее, надо обязательно указывать круглые скобки, а список параметров оставить пустым. Если функция не возвращает никаких параметров, то оператор *return* можно также опустить. В тексте программы сначала выводится значение переменной A, затем вызывается функция d с пустым списком параметров, и повторно выводится значение переменной A.

Упражнение № 7

Напишите функцию в секции <HEAD></HEAD>, возводящую число в квадрат, постройте с помощью нее таблицу квадратов чисел от 1 до 10 в любом виде. (Можно сделать двумерную таблицу в которой будут квадраты числе от 0 до 99).

Упражнение № 8

Напишите функцию table с двумя параметрами M и N, которая строит таблицу M×N ячеек. Можно сделать, чтобы функция не использовала операторы document.write, а возвращала текстовую строку, которая выводилась бы уже из скрипта, размещенного в тэге <BODY>, т.е. таким образом:

```
<SCRIPT>
  document.writeln(table(10,10));
  document.writeln(table(5,5));
</SCRIPT>
```

Для более осмысленного результата, пусть таблица будет раскрашена как шахматная доска.

Из функций можно составить библиотеку и пометить в отдельный файл с расширением .js. Такое размещение похоже на модель языка Pascal. Подключается такой модуль к HTML-файлу следующей конструкцией, которую обычно стараются пометить в секцию <HEAD>

```
<SCRIPT SRC="biblio.js"></SCRIPT>
```

Обратите внимание, что тег надо закрывать, хоть его тело и пустое.

Упражнение № 9

Переместите написанные функции из упражнений 7, 8 в отдельный файл и подключите их к HTML-файлу.

Занятие №4

Объектно-ориентированное программирование (ООП)

Javascript является объектно-ориентированным языком программирования. Это относительно новое направление в программировании. Хотя первый язык ООП – Small Talk – появился в 1972 г., широкое распространение ООП получило в 1990-х годах, когда появились объектно-ориентированные расширения языка C – C++ и Pascal – Object Pascal. Существует большое разнообразие моделей ООП. Общее во всех них, что в качестве данных, над которыми оперируют конструкции языка используются составные структуры, состоящие из *свойств* (собственно данных) и *методов* их обработки.

Работы с объектами в Javascript очень проста, существует набор стандартных объектов, и вы их просто используете. Имеется возможность создавать свои объекты, но она используется не часто. К сожалению, такие важные черты ООП, как наследование и полиморфизм реализованы в очень ограниченной форме.

Объекты в Javascript можно разделить на две группы: внутренние и внешние. Первые существуют всегда и являются скорее частью языка. Набор вторых – зависит о среды выполнения. Если это браузер, то это одни объекты, если это web-сервер, то другие и т.п. Дело в том, что Javascript стал настолько популярен, что он применяется не только для создания динамических HTML-страниц.

Внутренние объекты Javascript

Внутренние объекты не относятся к браузеру или загруженному в настоящее время HTML-документу. Эти объекты могутсоздаваться и обрабатываться в любой JavaScript-прграмме. Они включают в себя простые типы, такие как строки, а также более сложные объекты, в частности даты.

Имя объекта	**Описание**
Array	Массив.
Date	Дата и время
Math	Поддержка математических функций
Object	Обобщенный объект.
String	Текстовая строка. Не поддерживается в старых версиях

Объект Date

Объект содержит информацию о дате и времени. Этот объект имеет множество методов, предназначенных для получения такой информации. Кроме того, объекты **Date** можно создавать и изменять, например, путем сложения или вычитания значений дат получать новую дату. Для создания объекта **Date** применяется синтаксис:

dateObj = new Date(parameters)

где dateObj - переменная, в которую будет записан новый объект Date. Аргумент parameters может принимать следующие значения:

- пустой параметр, например date() в данном случае дата и время - системные.
- строку, представляющую дату и время в виде: "месяц, день, год, время", например "March 1, 2000, 17:00:00" Время представлено в 24-часовом формате;
- значения года, месяца, дня, часа, минут, секунд. Например, строка "00,4,1,12,30,0" означает 1 апреля 2000 года, 12:30.
- целочисленные значения только для года, месяца и дня, например "00,5,1" означает 1 мая 2000 года, сразу после полночи, так, как значения времени равны нулю.

Как уже говорилось ранее, данный объект имеет множество методов, свойств объект **Date** не имеет.

Методы

Методов много. Кратко опишем методы объекта **Date** в таблице. Наиболее важные выделены жирным шрифтом

Метод	**Описание метода**
getDate()	Возвращает день месяца из объекта в пределах от 1 до 31
getDay()	Возвращает день недели из объекта: 0 - вс, 1 - пн, 2 - вт, 3 - ср, 4 - чт, 5 - пт, 6 - сб.
getHours()	Возвращает время из объекта в пределах от 0 до 23
getMinutes()	Возвращает значение минут из объекта в пределах от 0 до 59
getMonth()	Возвращает значение месяца из объекта в пределах от 0 до 11
getSeconds()	Возвращает значение секунд из объекта в пределах от 0 до 59
getTime()	Возвращает количество миллисекунд, прошедшее с 00:00:00 1 января 1970 года.
getTimeZoneoffset()	Возвращает значение, соответствующее разности во времени (в минутах)
getYear()	Возвращает значение года из объекта
Date.parse(arg)	Возвращает количество миллисекунд, прошедшее с 00:00:00 1 января 1970 года. Arg - строковый аргумент.
setDate(day)	С помощью данного метода устанавливается день месяца в объекте от 1 до 31
setHours(hours)	С помощью данного метода устанавливается часы в объекте от 0 до 23
setMinutes(minutes)	С помощью данного метода устанавливаются минуты в объекте от 0 до 59
setMonth(month)	С помощью данного метода устанавливается месяц в объекте от 1 до 12
setSeconds(seconds)	С помощью данного метода устанавливаются секунды в объекте от 0 до 59
setTime(timestring)	С помощью данного метода устанавливается значение времени в объекте.
setYear(year)	С помощью данного метода устанавливается год в объекте year должно быть больше 1900.
toGMTString()	Преобразует дату в строковый объект в формате GMT.
toString()	Преобразует содержимое объекта Date в строковый объект.
toLocaleString()	Преобразует содержимое объекта Date в строку в соответствии с местным временем.
Date.UTC(year, month, day [,hours][,mins][,secs])	Возвращает количество миллисекунд в объекте Date, прошедших с 00:00:00 1 января 1970 года по среднему гринвичскому времени.

Пример №10.

```
<html>
<head>
<script language "JavaScript">
function showh() {
  var theDate = new Date();
  document.writeln(theDate.getDay()+"."+theDate.getMonth()+"."+theDate.getYear());
 }
 </script>
</head>
<body>
      <script language "JavaScript">
            showh();
      </script>
</html>
```

Разберем еще один пример. Подобный мы уже разбирали, когда рассматривали условные операторы, просто вспомним его и немного изменим: пусть меняется фон страницы в зависимости от времени суток.

Пример №11

```
<html>
<H1 ALIGN=CENTER>Пример №11</H1>

<script language "JavaScript">
 theTime = new Date();
 theHour = theTime.getHours();
 if (18 > theHour)
  document.writeln("<body bgcolor='White' text='Black'>");
 else
  document.writeln("<body bgcolor='Black' text='White'>");
</script>

</body>
</html>
```

Вероятно, вы успели заметить, что тег <body> создается в JavaScript-программе, а закрывается уже в статическом тексте HTML. Это вполне допустимо, так, как все теги расположены в правильном порядке.

Объект Math

Объект Math является встроенным объектом языка JavaScript и содержит свойства и методы, используемые для выполнения математических операций. Объект Math включает также некоторые широко применяемые математические константы. Синтаксис:

Math.propertyName
Math.methodName(parameters)

Свойства

Свойствами объекта Math являются математические константы:

E	Константа Эйлера. Приближенное значение 2.718 . . .
LN2	Значение натурального логарифма числа два. Приближенное значение 0.693 . . .
LN10	Значение натурального логарифма числа десять. Приближенное значение 2.302 . . .
LOG2_E	Логарифм *e* по основанию 2 (не вижу смысла в этой константе - это же корень из двух.) Приближенное значение 1.442 . . .)
LOG10_E	Десятичный логарифм *e*. Приближенное значение 0.434 . . .
PI	Число ПИ. Приближенное значение 3.1415 . . .
SQRT2	Корень из двух

Методы

Методы объекта Math представляют собой математические функции.

abs()	Возвращает абсолютное значение аргумента.
acos()	Возвращает арккосинус аргумента
asin()	Возвращает арксинус аргумента
atan()	Возвращает арктангенс аргумента
ceil()	Возвращает большее целое число аргумента, округление в большую сторону. Math.ceil(3.14) вернет 4
cos()	Возвращает косинус аргумента
exp()	Возвращает экспоненту аргумента
floor()	Возвращает наибольшее целое число аргумента, отбрасывает десятичную часть

log()	Возвращает натуральный логарифм аргумента
max()	Возвращает больший из 2-х числовых аргументов. Math.max(3,5) вернет число 5
min()	Возвращает меньший из 2-х числовых аргументов.
pow()	Возвращает результат возведения в степень первого аргумента вторым. Math.pow(5,3) вернет 125
random()	Возвращает псевдослучайное число между нулем и единицей.
round()	Округление аргумента до ближайшего целого числа.
sin()	Возвращает синус аргумента
sqrt()	Возвращает квадратный корень аргумента
tan()	Возвращает тангенс аргумента

Синтаксис очень прост, вызывается метод как любая функция, но это всеже метод и не забывайте указывать префикс **Math** перед методом: **var mpi = Math.Pi.** В данном случае переменной mpi присвоится значение Пи. Или, например, **var myvar = Math.sin(Math.Pi/4)**.

Упражнение № 10

Напишите программу, тестирующую работу функции **`toLocaleString().`**

Упражнение № 11

С помощью функции **`pow`** постройте таблицу степеней двойки от 1 до 20.

Упражнение № 12*

Ознакомьтесь с синтаксисом оператора **switch** – аналога оператора case в Pascal. Обратите внимание, что ветку выбора надо заканчивать оператором break. Секция **default** может отсутствовать.

```
switch(s)
      { case 1: document.writeln("Первый выбор");
                break;
        case 2: document.writeln("Второй выбор");
               document.writeln("Второй оператор");
                break;
        case 3: document.writeln("Третий выбор");
                break;
        default: document.writeln("Необязательный выбор по умолчанию");
      }
```

Напишите с помощью этого оператора программу, которая выводила бы текущий день недели.

Упражнение № 13**

Напишите с помощью функции random() скрипт, который случайным образом разбрасывал бы надписи по веб-странице. Воспользуйтесь абсолютным позиционированием. Ключевой оператор выглядел бы так:

```
document.writeln("<H1 STYLE='position: absolute; top: " + round(random()*500)+
"px; left: " + round(random()*500)+ "px '>Текст</H1>");
```

такой оператор, например, сгенерирует строку вида:

```
document.writeln("<H1 STYLE='position: absolute; top: 380px; left: 150px'> Текст</H1>");
```

которая разместит заголовок в странице произвольным образом. Повторите этот оператор в цикле for несколько раз и получите желаемый эффект.

Занятие №5

Объект Window

Объект *window* обычно соответствует главному окну браузера и является объектом верхнего уровня в языке JavaScript, поскольку документы, собственно, и открываются в окне. Предполагается, что такое окно существует, но при помощи метода window.open() можно открывать и другие окна и обращаться к ним посредством свойств объекта *window*.

Потребность в работе с объектом window возникает очень часто, поэтому префикс [window.] можно опускать. Например,

```
window.alert(”Добро пожаловать на сайт”);
```

можно записать просто как

```
alert(”Добро пожаловать на сайт”);
```

Объект window содержит следующие методы:

- **alert()** – вывод на экран текстового сообщения во всплывающем окне с кнопкой «ОК».
- **confirm()** – вывод на экран окна сообщения с кнопками «Yes» и «No», возвращает логическое значение true или false, в зависимости от нажатой кнопки.
- **prompt()** – вывод диалогового окна с полем для ввода текстовой строки, которая и будет результатом функции.
- **open()** – создает новое окно браузера.
- **close()** – закрывает окно браузера.
- **setTimeout()** – устанавливает в текущем окне обработку событий, связанных с таймером
- **clearTimeout()** – отменяет обработку таких событий.

и свойства:

- **defaultStatus** – текстовое сообщение, которое по умолчанию выводится в строке состояния браузера.
- **name** - заголовок окна, который задается с помощью аргумента windowName метода open().
- **status** - текст временного сообщения в строке состояния окна браузера.

События

Методы объекта window часто применяются в связи с какими-либо событиями, происходящими с браузером. Например, напишем html-страницу, которая открывает всплывающее окно с приветствием:

Пример №12

```
<HTML>
 <HEAD>

  <script language "JavaScript">
    function Hello()
    {
      alert("Приветствуем Вас на нашей странице!");
    }
  </script>

  </HEAD>

<BODY onLoad="Hello();”>
<H1 ALIGN=CENTER>Наша страница</H1>
</BODY>
</HTML>
```

В этом примере мы заводим функцию `Hello`, которая вызывает метод объекта `window` – `alert()`. Имя объекта window, как уже говорилось, можно опустить. В тэге `<BODY>` используется *обработчик события* `OnLoad`. Событие `OnLoad` возникает в момент окончания загрузки документа. Конструкция

```
onLoad="Hello();"
```

сообщает браузеру о том, что нужно выполнить оператор языка Javascript, в данном случае – вызов функции `Hello()`.

Каждый тэг HTML поддерживает различные события. Например, тэг `<BODY>` поддерживает события `onLoad`, `onUnload`, `onClick`, `onKeypress` и многие другие. Подробный список есть в любом справочнике по HTML.

Рассмотрим работу других методов объекта *window*. Например, иногда возникает необходимость открыть новое окно при посещении страницы. Это можно сделать следующим образом:

Пример №13

```
<HTML>
 <HEAD>

  <script language="JavaScript">

    function OpenHelloWnd() // эта функция откроет новое окно.
    { var NewWindow;        // необязательная переменная
      NewWindow=window.open("n14a.html","",
            "toolbar=no,menubar=0,scrollbar=no,width=250,height=160")
    }

  </script>

 </HEAD>

<BODY onLoad="OpenHelloWnd();">

<H1 ALIGN=CENTER>Наша страница</H1>
 При открытии этой страницы (n14.html)
 автоматически откроется страница n14a.html

</BODY>

</HTML>
```

Проанализируем этот пример. В заголовке документа описана функция `OpenHelloWnd`. В ней вызывается метод объекта `window` – `open`.

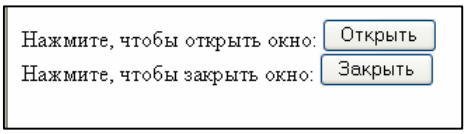
У него три параметра:

- первый – имя открываемого файла;
- второй – необязательное имя окна, если этот параметр – пустая строка, то в качестве имени используется имя, указанное в тэге `<TITLE>` открываемого файла;
- третий параметр – символьная строка, в которой указываются параметры открытия нового окна; в данном случае создается окно без панели кнопок, без меню, без полос скроллинга, задается также ширина и высота окна.

Очень часто программы на Javascript используются совместно с формами[1] в HTML-страницах. Формы могут состоять из кнопок, переключателей, элементов для ввода текста, списков. Подробнее формы будут рассмотрены в следующем занятии. Сейчас мы ограничимся формами, в которых есть только нажимающиеся кнопки. Такая форма может быть задана следующей конструкцией:

```
<FORM>
 Нажмите, чтобы открыть окно:
   <INPUT TYPE="Button" VALUE="Открыть" OnClick="OpenWnd();"><BR>
 Нажмите, чтобы закрыть окно:
   <INPUT TYPE="Button" VALUE="Закрыть" OnClick="CloseWnd();"><BR>
</FORM>
```

Вид этой формы приведен на рисунке. Обратите внимание, что тэги HTML и текст могут использоваться для оформления формы.



Любая форма заключается в тэг <FORM>, внутри которого могут использоваться любые конструкции HTML, в том числе и специальный тэг <INPUT>, в данном случае создающий кнопку. Параметры тэга определяют вид управляющего элемента (TYPE="Button" – кнопка), надпись на кнопке (VALUE) и действие по нажатию на кнопку. Пример №15 иллюстрирует ее работу.

Пример №14

```
<HTML>
 <HEAD>
   <TITLE>Кнопки</TITLE>
   <SCRIPT>
     var GlobalWindow; // глобальная переменная, содержащее объект "окно"

     function OpenWnd()
     {
        GlobalWindow=window.open("js14a.html","",
           "toolbar=no,menubar=0,scrollbar=no,width=250,height=160");
     }

     function CloseWnd()
     {
       GlobalWindow.close(); // вызывается метода close
     }

   </SCRIPT>
 </HEAD>

<BODY>
<FORM>
 Нажмите, чтобы открыть окно:
  <INPUT TYPE="Button" VALUE="Открыть" OnClick="OpenWnd();"><BR>
 Нажмите, чтобы закрыть окно:
   <INPUT TYPE="Button" VALUE="Закрыть" OnClick="CloseWnd();"><BR>
</FORM>

</BODY>
</HTML>
```

Прокомментируем пример. В заголовке HTML-страницы определяются две функции и глобальная переменная. Так как переменная описана раньше всех функций, то ее имя будет известно в функциях. Метод `window.open` открывает новое окно, и присваивает переменной `GlobalWindow` дескриптор[2] этого окна, с которым можно работать так же, как с объектом `window`. Это и происходит в функции `CloseWnd`. В ней вызывается метод `close` для объекта `GlobalWindow`. В форме же указывается, что при нажатии на кнопку «Открыть» следует выполнить функцию `OpenWnd()`, а при нажатии на кнопку «Закрыть» – `CloseWnd()`.

[1] Формы – диалоговые элементы HTML-страницы. Подробнее будут рассмотрены чуть позже.
[2] Дескриптор – это описатель объекта, можно считать, что это указатель на объект или даже сам объект.

Упражнение №14

Метод объекта window **confirm**("*Текст*") возвращает значения «истина» или «ложь», т.е. может использоваться в операторе if:

```
if ( confirm('Закрыть окно') ) {  … }   else { … }
```

Напишите страницу (модифицировав пример №14) , в которой с помощью этой функции выдавался бы запрос на разрешение открытия (или закрытия) дополнительного окна.

Упражнение №15

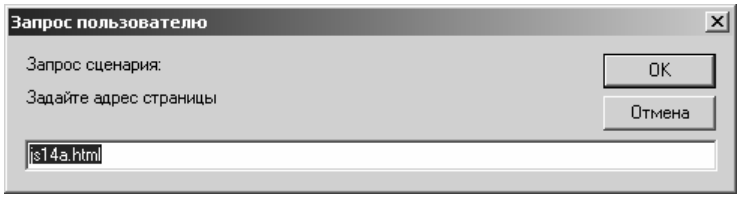
Практически любой тэг (например <P>, <TD>, <H1> и пр.) поддерживают события onMouseover, onClick, onMouseOut. Задействуйте эти события для открытия дополнительных окон с помощью методов `open` или `alert`.

Упражнение №16

Метод объекта window **prompt** возвращает введенную пользователем текстовую строку, т.е. возможен вариант такой вариант использования этой функции

```
s = prompt('Задайте адрес страницы','js14a.html');
```

Текст, заданный первым параметром функции отображается в открывшемся окне, второй параметр задает значение функции по умолчанию, которое также отображается в поле ввода. Этот параметр можно задать пустым.



Используйте этот метод, для предоставления пользователю возможности выбора файла для открытия дополнительного окна.

Упражнение №17*

Присвоение значения полю window.status меняет текст в строке статуса браузера. Используйте эту возможность в сочетании с параметрами onMouseover и onMouseout. При грамотном употреблении строки статуса создается хороший визуальный эффект.

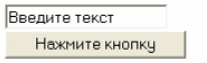
Занятие №6

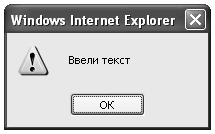
JavaScript и формы в HTML

Формы – способы организации диалога с пользователем через web-страницы. С использованием форм вы все хорошо знакомы: любая поисковая система – пример простейшей формы. Данные, набранные пользователем в форме, могут быть отосланы броузером серверу на обработку (так и делает поисковая система), но данные форм можно обрабатывать и непосредственно в броузере с помощью JavaScript. Это мы и освоим.

Форма состоит из элементов управления: текстовых полей, переключателей, кнопок, заключенных в парный тег <FORM>.

Пример №16






```
<FORM NAME="FM1">
 <INPUT TYPE="text"   VALUE="Введите текст" NAME="text1"><BR>
 <INPUT TYPE="button" VALUE="Нажмите кнопку" onClick="alert(FM1.text1.value)">
</FORM>
```

Давайте подробно разберем этот пример. Тег `<FORM NAME="FM1">` указывает начало формы, задавая ее имя, которое используется в обработчике события onClick. Следующий тег `<INPUT TYPE="text">` определяет поля для ввода текста. Параметр Value задает текст, появляющейся в поле по умолчанию (см. рисунок). Параметр NAME опять задает имя объекта в JavaScript. Таким образом, если вы планируете обращаться к тегам HTML из программ на JavaScript, то всегда необходимо задавать параметр NAME. Учтите, что имя, указанное в этом параметре чувствительно к регистру букв (так как это будет объект JavaScript). Последний тег `<INPUT TYPE="button">` строит кнопку, надпись на которой задается параметром VALUE. Параметра NAME у кнопки в нашем случае нет, так как мы не собираемся управлять поведением кнопки из JavaScript. Однако у этого тэга есть новый интересный параметр onClick, который задает обработку события «щелчок мышью». Значением этого параметра является строка на JavaScript, которую следует выполнить при наступлении события. В данном случае – это вызов метода alert, параметром которого является значение поля text1 формы FM1. Еще раз *обратите внимание* на эту запись: `FM1.text1.value`.

Рассмотрим другой пример с использованием текстовых полей.

Пример №17

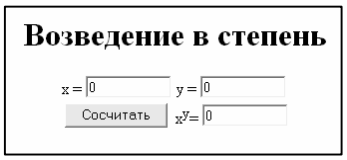

```
<HTML>
<HEAD>  <SCRIPT>
          function sqr(t)  { return t*t }
        </SCRIPT>
</HEAD>

<BODY>
<H1 ALIGN=CENTER>Возведение в квадрат</H1>
<P ALIGN=CENTER>
<FORM NAME="FM2">
<TABLE>
<TR ALIGN=CENTER>
  <TD><INPUT TYPE="text" VALUE="0" NAME="x" SIZE=10></TD>
  <TD><INPUT TYPE="text" VALUE="0" NAME="y" SIZE=10></TD>
</TR>
<TR ALIGN=CENTER><TD COLSPAN=2>
 <INPUT TYPE="button" VALUE="Сосчитать" onClick="FM2.y.value=sqr(FM2.x.value)">
</TD></TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

В данном примере определена функция sqr, возвращающая квадрат своего аргумента. В форме используется два текстовых поля с именами **x** и **y**. В тегах указана с помощью параметра SIZE длина поля (в ширинах символа m), однако текст можно вводить и из большего числа символов (будет осуществляться автоматический скроллинг). Кнопка <**INPUT TYPE="button"**> содержит обработчик события. Обратите внимание, что для доступа к значению мы всегда после имени поля пишем слово value, так как сами по себе объекты x и y содержат много других свойств.

Заметьте, что в примере №17 для улучшения внешнего вида формы мы используем таблицу. Это очень распространенный приём!

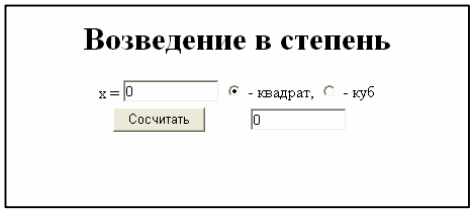
Упражнение №18

Модифицируйте пример 17, добавив еще одно текстовое поле, так чтобы он производил возведение в степень. Подсказка: функцию «возведение в степень» не обязательно программировать самому. Можно использовать встроенную функцию pow объекта Math: Math.pow(x,y) (см. стр. 11).



Кроме кнопок и полей для ввода текста в формах можно использовать такие элементы управления как «радиокнопки» и «флажки». Следующий html-текст показывает использование «радио-кнопок» (взаимоисключающих переключателей).

Пример №18




```
<HTML>
<HEAD>
 <SCRIPT>
  var kind=2; // Глобальная переменная
  function sqr(t)  { return t*t }
  function cube(t) { return t*t*t }
  function todo(t)
    { if (kind==2) return sqr(t); else return cube(t); }
 </SCRIPT>
</HEAD>

<BODY>
<H1 ALIGN=CENTER>Возведение в степень</H1>
<P ALIGN=CENTER>
<FORM NAME="FM3">
<TABLE>
<TR ALIGN=CENTER>
  <TD>x = <INPUT TYPE="text" VALUE="0" NAME="x" SIZE=10></TD>
  <TD><INPUT TYPE="radio" NAME="p" onClick="if (this.checked) kind=2" CHECKED> - квадрат,
      <INPUT TYPE="radio" NAME="p" onClick="if (this.checked) kind=3"> - куб
  </TD>
</TR>
<TR ALIGN=CENTER>
  <TD><INPUT TYPE="button" VALUE="Сосчитать" onClick="FM3.z.value=todo(FM3.x.value)"></TD>
  <TD><INPUT TYPE="text" VALUE="0" NAME="z" SIZE=10></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

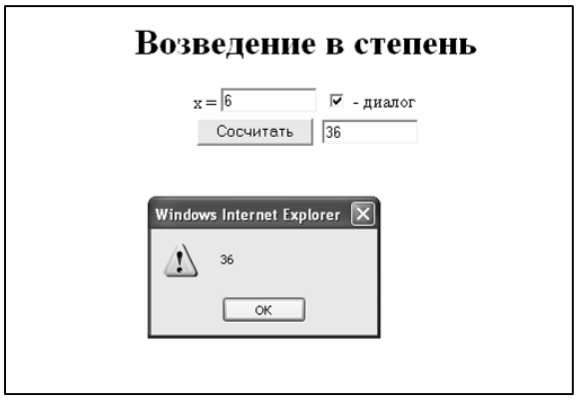
Рассмотрим подробнее этот пример. В секции HEAD определены три функции и одна глобальная переменная kind. Эта переменная описана вне и перед всеми функциями, следовательно она «видна» во всех частях нашего html-текста.

Переключающиеся кнопки заданы тегом <**INPUT TYPE="radio"**> с одинаковым именем **p**, что обеспечивает автоматическое переключение кнопок. У одной из радио-кнопок указан параметр CHECKED, в силу этого кнопка появляется сразу отмеченной. На событие «щелчок мышью» происходит обработка: в операторе if происходит проверка на «отмеченность» кнопки. Ключевое слово

this заменяет имя объекта (можно было бы написать FM3.p.checked, но согласитесь, this.checked – понятнее). Если после щелчка мыши кнопка перешла в отмеченное состояние, то значение глобальной переменной устанавливается в 2 или в 3 соответственно.

Последний элемент управления, который мы рассмотрим в сегодняшнем занятии – это «флажок» или checkbox. Его поведение еще проще, чем радио-кнопки, так как его состояние не влияет на другие кнопки. Он просто может либо отмечен, либо нет.

Пример №19




```
<HTML>
<HEAD>
 <SCRIPT>
  var s=false; // Глобальная переменная
  function sqr(t)  { return t*t }
 </SCRIPT>
</HEAD>

<BODY>
<H1 ALIGN=CENTER>Возведение в степень</H1>
<P ALIGN=CENTER>
<FORM NAME="FM4">
<TABLE>
<TR ALIGN=CENTER>
  <TD>x = <INPUT TYPE="text" VALUE="0" NAME="x" SIZE=10></TD>
  <TD><INPUT TYPE="checkbox" onClick="if (this.checked) s=true; else s=false">
           - диалог</TD>
</TR>
<TR ALIGN=CENTER>
  <TD><INPUT TYPE="button" VALUE="Сосчитать"
         onClick="FM4.z.value=sqr(FM4.x.value); if (s) alert(FM4.z.value)"></TD>
  <TD><INPUT TYPE="text" VALUE="0" NAME="z" SIZE=10></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

В этом примере тег **<INPUT TYPE="checkbox">** создает «флажок» (возможно использовать параметр CHECKED с тем же смыслом, что и у радио-кнопки). Обработчик события onClick знакомым уже вам способом устанавливает значение глобальной переменной s в true или false. Обработчик события кнопки **<INPUT TYPE="button">** проверяет значение переменной s и вызывает метод alert при истинном значении этой переменной.

Упражнение №19

Модифицируйте пример 18, создав калькулятор, выполняющий указанное радио-кнопками арифметическое действие с полями x и y, выводя результат в поле z.

Примечание: вместо вложенных операторов if для выбора отмеченного кнопкой действия можно использовать оператор switch – аналог оператора case в языке Pascal. Например, для вычисления можно было бы использовать следующую функцию:

```
function calculate(x,y)
  { switch(s)
      { case 1: return x+y; break;
        case 2: return x-y; break;
        case 3: return x*y; break;
        case 4: return x/y; break;
      }
  }
```

Занятие №7

JavaScript и формы в HTML, тег Select

Рассмотрим устройство одного блока нашей стартовой страницы:

```
<FORM NAME=FM>
 <SELECT NAME="SEL" onChange="window.status=value">
    <OPTION VALUE="http://www.yandex.ru">Яндекс</OPTION>
    <OPTION VALUE="http://www.rambler.ru/">Rambler</OPTION>
    <OPTION VALUE="http://www.stars.ru">Stars</OPTION>
    <OPTION VALUE="http://www.aport.ru/">Aport</OPTION>
    <OPTION VALUE="http://www.turtle.ru/">Turtle</OPTION>
    <OPTION VALUE="http://www.km.ru/">"Кирилл и Мефодий"</OPTION>
 </SELECT>
 <INPUT TYPE="BUTTON" VALUE="GO"
         onClick="window.location.href=FM.SEL.value"
         onMouseOver="title=FM.SEL.value">
</FORM>
```

1. На странице определена форма тэгом <FORM NAME=FM>
2. В форме находятся два управляющих элемента:
   2.1. Раскрывающийся список <SELECT ...>, в который вложены теги <OPTION>. Они задают значения строк, которые будут видны в раскрывающемся списке. Параметр value определяет значение строки, которое будет использовано в дальнейших операторах Javascript.
   2.2. Кнопка <INPUT TYPE=”BUTTON” ...>. Параметр VALUE в данном случае задает надпись на кнопке.
3. В теге <SELECT> определена обработка события **onChange**, по которому меняется текст в строке статуса броузера (*внизу окна на серой рамочке*).
4. В теге <INPUT> определена обработка двух событий:
   4.1. **onClick**, по которому свойству окна **location.href** (которое отвечает за текущий файл, загруженный в броузер) присваивается значение FM.SEL.value, т.е. имя формы–точка–имя элемента управления–точка–value. Таким образом осуществляется переход к указанному адресу.
   4.2. **OnMouseOver**, по которому вызывается всплывающая подсказка (в желтом прямоугольнике), если мышку некоторое время подержать над кнопкой.

Упражнение № 20.

Составьте свой вариант стартовой страницы с сайтами, которые вы чаще всего посещаете.

Упражнение №21*.

Модифицируйте сделанное вами упражнение №19 (калькулятор), но вместо переключателей радио-кнопок используйте список select (укажите название операций, с которые будут выполнены над числами).