Лабораторные задания по курсу «Основы информационных технологий. Программирование с использованием JavaScript»

Теоретические сведения

Для встраивания скрипта в HTML страницу используйте следующую конструкцию:

```
<html>
 <head>
 </head>
    <body>
      <script type='text/javascript'>
       // Здесь находится код JavaScript.
      </script>
  </body>
</html>
```

Для подключения внешнего скрипта:

```
<script type='text/javascript' src='code.js'></script>
```

Ввод и вывод информации в JavaScript

1 Вывод информации:

1.1 Вывод информации при помощи функции ***document.writeln(string)***:

```
<script type='text/javascript'>
    document.writeln('Hello World!');
</script>
```

1.2 Вывод информации при помощи сообщения ***alert***:

```
<script type='text/javascript'>
    alert('Hello World!');
</script>
```

1.3 Вывод информации при помощи функции getElementById(tagId):

```
…
<div id='placeForText'></div>
  <script type='text/javascript'>
      document.getElementById('placeForText').innerHTML='Hello     World!';
  </script>
…
```

Все в HTML коде может иметь параметр id. Любой тэг имеет вложенную в него строку (строка может быть и пустой). Через id тэга можно обратиться к его содержимому и, соответственно, изменить. При этом другие тэги затронуты не будут.

Возможен вариант, при выполнении которого внутрь элемента будет вставлен отформатированный текст с использованием HTML тэгов.

```
…
<div id='feedback'></div>
 <script type='text/javascript'>
    document.getElementById('feedback').innerHTML='<P><font  color=red>Hello </font>';
 </script>
…
```

2 Ввод данных в скрипт:

2.1 При помощи функции ***prompt(stringTitle, stringText):***

```
<script type='text/javascript'>
   answer = prompt('Enter your greating', 'Hello World!');
</script>
```

2.2 Пользовательский ввод:

Для создания поля ввода и кнопки подтверждения ввода используйте следующий код HTML:

```
<input id='userInput' size=60>
<button onClick='userSubmit()'>Submit</button><br>
<P><div id='result'></div>
```

Теперь припишем код для взаимодействия пользователя с программой:

```
<script type='text/javascript'>
    function userSubmit() {
        var UI=document.getElementById('userInput').value;
        document.getElementById('result').innerHTML='Вы ввели:'+UI;
    }
</script>
```

При нажатии на кнопку, пустой абзац будет отображать только что введенный текст. Здесь происходить следующая последовательность действий:

- В переменную UI записывается значение поля ввода “userInput”;
- В абзац “result” записывается текст “Вы ввели: ” плюс содержимое переменной UI (то есть то, что находится сейчас в поле ввода “userInput”).

Можно пойти дальше и отказаться от кнопки “Submit”, для этого замените первый введенный блок кода HTML на следующий:

```
<input id='userInput' onKeyUp="userSubmit()" size=60><BR>
<P><div id='result'></div>
```

Теперь текст будет появляться на экране непосредственно после ввода новой буквы в поле ввода “userInput”.

Переменные в JavaScript

JavaScript не является строго типизированным языком программирования, а потому, переменные могут хранить в себе абсолютно все, что угодно, даже указатели на функции, без указания типа данных при их объявлении. Кроме того, переменные можно вовсе не объявлять (но не желательно). При первом ее упоминании она будет создана автоматически.

Переменная в языке JavaScript может начинаться с цифры или любого символа, кроме '$' и '_'. Вторую, третью и последующие позиции в названии могут занимать любые символы. Также рекомендуется избегать совпадения имен переменных с именами тэгов HTML, например (в Internet Explorer произойдет ошибка, так как он не различает два пространства имен).

Ниже приведен код с комментариями, который показывает каким образом можно использовать переменные:

```
var thisIsAString = 'This is a string'; // Объявление переменной, содержащей строку
var alsoAString = '25'; // Аналогично
var isANumber = 25; // Объявление переменной, содержащей число
var isEqual = (alsoAString == isANumber); // Истина, так как обе переменные равны 25
var isEqual = (alsoAString === isANumber); //Ложь, так как переменные имеют разный тип
var concat = alsoAString + isANumber; // Переменная concat равно 2525
var addition = isANumber + isANumber; // Переменная addition равно 50
var alsoANumber=3.05; // Переменная содержит число с плавающей точкой
var floatError=0.06+0.01; // Переменная содержит число 0.06999999999999999
var anExponent=1.23e+3; // Результат вычисления равен 1230
var hexadecimal = 0xff; // Результат равен 255 (шестнадцатеричная система)
var octal = 0377; // Результат равен 255 (восьмеричная система)
var isTrue = true; // Булева переменная
var isFalse= false; // Аналогично
var isArray = [0, 'one', 2, 3, '4', 5]; // Объявление массива
var four = isArray[4]; // Присвоение пятого (нумерация с нуля) элемента массива
переменной
var isObject = { 'color': 'blue', // Объявление объекта
                 'dog': 'bark',
                 'array': [0,1,2,3,4,5],
                 'myfunc': function () { alert('do something!'); }
               }
var dog = isObject.dog; // Переменная dog теперь хранит значение 'bark'
isObject.myfunc(); // Появится окно с надписью "do something!"
```

```
var someFunction = function() { // Переменная указывает на функцию
    return "I am a function!";
}
var alsoAFunction = someFunction; // И эта переменная теперь указывает на ту же функцию
var result = alsoAFunction(); // А эта переменная хранит строку "I am a function!"
```

События в JavaScript

JavaScript – язык программирования, основанный на событиях. Написанный код не будет постоянно выполняться в браузере, а, вместо этого, будет ожидать возникновения различных событий в системе. Допустим, произошло какое-либо событие. Если код умеет его обрабатывать, то ему будет выделено и время, и ресурсы компьютера для выполнения этой задачи.

Все в HTML имеет id, и кроме того, имеет события, которые срабатывают при различном взаимодействии с объектом (клик мыши, нажатие кнопки на клавиатуре, двойной клик и т.п.)

```
<div id='feedback' onClick='goodbyeWorld()'>Включите JavaScript!</div>
   <script type='text/javascript'>
      document.getElementById('feedback').innerHTML='Привет!';
      function goodbyeWorld() {
            document.getElementById('feedback').innerHTML='Давай,до свидания!';
      }
   </script>
```

Ниже представлена таблица основных событий, которые наиболее часто используются:

Событие	***Описание***
onAbort	*Невозможность загрузки изображения.*
onBeforeUnload	*Пользователь переключился с просматриваемой страницы на другую.*
onBlur	*Потеря фокуса с элемента.*
onChange	*Изменение содержимого элемента.*
onClick	*Клик мышью по элементу.*
onDblClick	*Двойной клик мышью по элементу.*
onError	*Возникновение ошибки при загрузке изображения.*
onFocus	*Установка фокуса на элемент.*
onKeyDown	*Нажатие клавиши клавиатуры.*
onKeyPress	*Нажатие или отпускание клавиши клавиатуры.*
onKeyUp	*Отпускание клавиши клавиатуры.*
onLoad	*Окончание загрузки элемента (iframe, image, script).*
onMouseDown	*Нажатие клавиши мыши на элементе.*

Событие	***Описание***
onMouseMove	*Перемещение мыши по элементу.*
onMouseOut	*Перемещение мыши за границу элемента.*
onMouseOver	*Появление мыши внутри границы элемента.*
onMouseUp	*Отпускание клавиши мыши на элементе.*
onReset	*Нажатие кнопки “Reset” формы.*
qonResize	*Изменение размеров окна или фрейма.*
onSelect	*Выделение текста.*
onSubmit	*Нажатие кнопки “Submit” формы.*
onUnload	*Уход со страницы.*

Арифметические и логические операции

В JavaScript существует две операции сравнения: одна из них сравнивает два объекта на равенство по существу (“==”), а другая сравнивает равенство по сущности (“===”). Пусть у нас есть число 25 и строка “25”. По существу это одно и тоже, и операция “==” вернет результат true. По сущности же это разные объекты, ведь один из них число, а другой – строка. Поэтому операция “===” вернет значение false.

Для создание операции “не равно”, необходимо первый символ “=” заменить на “!”. То есть теперь получаться операции “!=” и “!==” соответственно.

Операция	**Описание**
=	Присвоение значения: x=5
==	Проверка равенства по существу: x==5
===	Проверка равенства по сущности: x===5
!=	Проверка не равенства по существу: x!=5
!==	Проверка не равенства по сущности: x!==5
!	Логическое отрицание: !true===false
||	Логическое ИЛИ: (x==5) || (y==7)
&&	Логическое И: (x==5) && (y==7)
<, >, <=, >=	Сравнения на меньше, больше, меньше или равно, больше или равно

Условные операторы

1 Условный оператор IF

Этот оператор позволит выполнить некоторый код только в случае успешного завершения некоторого логического теста. Пример:

```
var x=5;
if (x==5) {
    alert('x равно 5!');
}
```

Также можно указать действия, которые необходимо выполнить при ложности входного логического выражения. Пример:

```
var x=5;
if (x==5) {
    alert('x равно 5!');
} else {
    alert('x не равно 5!');
}
```

Для этого используется ключевое слово “***else***”.

Существует сокращенная запись нескольких вложенных условных операторов. Пример:

```
var x=5;
if (x==1) {
    alert('x равно 1');
} else if (x==2) {
    alert('x равно 2');
} else if (x==5) {
    alert('x равно 5');
} else {
    alert("x не 1, 2 или 5");
}
```

2 Оператор выбора SWITCH

Очень полезный оператор, который позволяет избавиться от большого количества “else if”. Пример:

```
var x=5;
switch (x) {
    case 1: alert('x равно 1!'); break;
    case 2: alert('x равно 2!'); break;
    case 5: alert('x равно 5!'); break;
    default: alert("x не равно 1, 2 или 5!");
}
```

Первым пишется слово “***switch***”, затем имя переменной в скобках, значение которой мы будем сравнивать со значениями, указанными после ключевого слова “***case***”. Если значения совпадут, то выполнится соответствующий код, причем эта точка будет началом выполнения блока кода, начинающегося от нее и до самого конца блока, то есть до закрывающей фигурной скобки. Чтобы этого не произошло, пишется ключевое слово “***break***”, которое приказывает интерпретатору закончить выполнение оператора “***switch***”. Ключевое слово “***default***” обозначает “любое значение”, то есть код, начинающийся справа от этого места будет выполнен в случае, если иные значения не подойдут.

Абсолютно верен следующий блок кода. В нем используется тот факт, что оператор выбора сравнивает значение после “***case***” со значением, указанным в круглых скобках с помощью оператора “==”:

```
var x=5;
switch (true) {
    case (x==1): alert('x равно 1!'; break;
    case (x==2): alert('x равно 2!'; break;
    case (x==5): alert('x равно 5!'; break;
    default: alert("x не равно 1, 2 или 5!");
}
```

3 Фактические параметры функции

В JavaScript нет возможности задания значения параметров функции по-умолчанию (в случае, если пользователь их не передал вовсе или передал неинициализированную переменную). Однако его можно легко обойти, используя следующий трюк:

```
var someVariable = (присвоить это значение, если оно не ноль или null) || (иначе
присвоить это значение)
```

Задание значений параметров функции по умолчанию:

```
function doAddition(firstVar, secondVar) {
    var first = firstVar || 5;
    var second= secondVar || 10;
    return first+second;
}
doAddition(12);
```

Переменной “***first***” будет присвоено значение переменной “***firstVar***”, если оно не равно нулю или ***null***, иначе будет присвоено значение ***5***. То же самое происходит и с переменной “***second***”. ***Внимание***: при использовании этого метода в функцию не получится передать ноль, так как он будет заменен на другое значение.

4 Сокращенный условный оператор

Использование этого варианта условного оператора не особо желательно, но в некоторых случаях он позволяет значительно сократить объем кода. Пример:

```
var userName = 'Bob';
var hello = (userName=='Bob') ? 'Hello Bob!' : 'Hello Not Bob!';
```

Переменная будет наделена либо значением, которое стоит после знака вопроса, либо значением, которое стоит после двоеточия. До знака вопроса происходит логическая проверка. Схема такова:

```
var someVariable = (логическое выражение) ? (если выражение истинно) : (если выражение ложно);
```

Циклы в JavaScript

Циклы – очень важная составляющая любого языка программирования. Благодаря им возможно создавать очень короткий код, который будет работать бесконечно долго.

1 Цикл FOR

Данный вид цикла использует некоторую переменную, значение которой изменяется по определенному закону при каждой итерации (прохождения тела цикла целиком). Пример:

```
for (var i=0; (i<5); i++) {
    document.writeln('I равно '+i+'<br>');
}
// Будет напечатано: I равно 0
// Будет напечатано: I равно 1
// Будет напечатано: I рвано 2
// Будет напечатано: I рожно 3
// Будет напечатано: I равно 4
```

2 Цикл FOR/IN

Очень удобный вариант предыдущего цикла с небольшим изменением. Он пригоден для использования с массивами и прочими объектами, содержащими в себе наборы перечислимых данных. Пусть у нас имеется следующий объект:

```
var myObject = { 'animal' : 'dog',
                 'growls' : true,
                 'hasFleas': true,
                 'loyal' : true }
```

Теперь мы можем работать с ним следующим образом, абсолютно не беспокоясь о количестве элементов внутри объекта. Словестно такой вид цикла можно описать как “выполнять для каждого элемента из ...”:

```
for (var property in myObject) {
    document.writeln(property + ' содержит ' + myObject[property]+'<br>');
}
// Будет напечатано: animal содержит dog
// Будет напечатано: growls содержит true
// Будет напечатано: hasFleas содержит true
// Будет напечатано: loyal содержит true
```

3 Цикл WHILE

Цикл, который будет выполняться неопределенное заранее число раз. Его название так и переводится с английского, как “пока ...”. Пример:

```
var x = 1;
while (x<5) {
    x = x +1;
}
var x = 1;
while (true) {
    x = x + 1;
    if (x>=5) {
        break;
    }
}
```

Существует модификация цикла ***WHILE***, когда условие выполнения цикла определено в конце. Таким образом интерпретатор обязательно выполнит тело цикла хотя бы один раз:

```
var x=1;
do {
    x = x + 1;
} while (x < 5);
```

4 Управление циклом

Внутри циклов и только внутри них могут быть использованы два ключевых слова, которые позволяют управлять ходом выполнения программы внутри циклов.

Первое слово: ***CONTINUE***. Когда интерпретатор дойдет до этого слова, то немедленно перейдет к началу цикла и начнет выполнять его заново, но уже для следующей итерации. Это ключевое слово используют, когда нет необходимости выполнять код для данных, которые используются в цикле на данной итерации. Например, программа, которая считает квадрат каждого значения массива. Это фантастика, вы дочитали до сих пор. Значения массива определены случайным образом. Пусть они будут всегда неотрицательными. При такой постановке задачи у нас нет необходимости возводить в квадрат единицу и ноль, поэтому в самом начале цикла имеет смысл поставить оператор

IF, который будет проверять, является ли очередное значение массива 1 или 0 и, если да, то переходить к следующему значению и не выполнять ненужные вычисления.

Второе слово: ***BREAK***. Оно прекращает выполнение цикла и переходит к обработке следующей за циклом кодовой строчке.

Практическая часть

1. Вычислить значение выражения по формуле (все переменные принимают вещественные значения): $\frac{x^2 - 7x + 10}{x^2 - 8x + 12}$

2. Известна масса трех гантелей. Найти сумму масс гантелей, выделить самую тяжелую и самую легкую.

3. Посчитать количество согласных и гласных в Вашей фамилии.

4. Определить, сколько раз в строке длиной 20 символов повторяется символ под номером 5.

5. Даны два вещественных положительных числа x и y. арифметические действия над ними пронумерованы (сложение – 1, вычитание – 2, умножение – 3, деление - 4). Составить программу, которая по введённому номеру выполняет то или иное действие над числами.

6. Вычислить значение сопротивления участка цепи по введенным напряжению и току на данном участке.

7. Определить эквивалентное сопротивление двух последовательно и параллельно соединенных проводников с сопротивлениями R1 и R2.

8. Дана последовательность слов. Напечатать все слова, предварительно выполнив преобразования по правилу: заменить в каждом слове первую встречную букву а буквой о, удалив все остальные (если в слове нет такой буквы, то ничего не делать).

9. Написать скрипт, определяющий текущее время и вычисляющий время завтрака (7^{00}-8^{00}), обеда (13^{00}-14^{00}), ужина (19^{00}-20^{00}), работы (9^{00}-18^{00}), отдыха (18^{00}-23^{00}), сна (23^{00}-6^{30}).

10. Создайте код, который выводит все простые числа из интервала от **2** до **10**.(Натуральное число, большее 1, называется *простым*, если оно ни на что не делится, кроме себя и 1**)**

11. Подсчитать количество слов во введенном строке, найти длину максимального слова, минимального слова и среднюю длину слов. Предусмотреть возможность присутствия в тексте нескольких пробелов и знаков препинания.

12. Определить, является ли введенная строка палиндромом (справа налево читается так же, как и слева направо).

13. Написать программу определения корректности ввода пользователем адреса электронной почты. Программа должна проверять следующие условия:

    - наличие символа «@»;
    - минимальный размер адреса – 6 символов (например, a@b.by);
    - слева от символа «@» должен быть, как минимум, один символ;
    - справа от символа «@» должна быть, как минимум, одна точка;
    - справа от последней точки должно быть, как минимум, 2 символа;
    - между символом «@» и следующей за ним точкой должен быть, как минимум, один символ.

    Программа должна выдавать сообщение с указанием соответствующей ошибки.

14. Задана точка с координатами X,Y. Определить, относится ли она кругу с центром C (Xc, Yc) и радиусом R.

15. Вывести значения функции Y = 4 * SIN (T) - 0.5 * SIN (T) на промежутке [0.1, 0.8], с шагом 0.05.

16. Напишите программу, в которой по известной начальной скорости $\boldsymbol{V_0}$ и времени полета тела $\boldsymbol{t}$ определяется угол, под которым тело брошено по отношению к горизонту.

17. Посчитайте площадь треугольника по:

    - длинам трех сторон;
    - трем углам и стороне;
    - двум сторонам и углу между ними;
    - двум сторонам и необразованному ими углу;
    - высоте треугольника и стороне к которой опущена эта высота;
    - трем углам и стороне.

    В задаче предусмотреть проверку на существование треугольника

18. Вывести ряд чисел Фибоначчи, состоящий из n элементов. (Числа Фибоначчи – это элементы числовой последовательности 0, 1, 1, 2, 3, 5, 8, 13, 21, 34…, в которой каждое последующее число равно сумме двух предыдущих)

19. Написать скрипт, который оставляет в строке только один экземпляр каждого встречающегося символа.

20. Создать массив из n (входное значение) элементов, автоматически заполнить его случайными числами от 0 до 10 и вывести массив и сумму его элементов.