**Тема:** Решение комбинированных задач посредством JavaScript

**Цель:** изучить способы управления окнами и таблицами.

**Вид занятия:** Лабораторная работа.

**Тип занятия:** Формирование навыков.

**Время выполнения:** 2 часа

**Оборудование и методическое обеспечение:** IBM PC, приложение Блокнот, MS Internet Explorer карточки с заданием.

**Литература:**

1. .

**План работы:**

1. Организационный этап занятия
2. Проверка домашнего задания
3. Информационно-подготовительный этап занятий
4. Исполнительский этап занятий
5. Постановка домашнего задания
6. Оценочно-рефлексивный этап занятия

**Ход занятия**

Устный опрос учащихся по следующим вопросам:

1. Для чего предназначен объект Image?
2. Перечислите известные вам методы объекта Image
3. Перечислите управляющие элементы формы

***Информационно-подготовительный этап занятий***

Язык JavaScript отличается от других языков высокого уровня, прежде всего тем, что программы, написанные на нём, не могут выполняться самостоятельно. Они либо встроены в HTML-документ, либо расположены в отдельных файлах и присоединяются к HTML-документу при его загрузке в браузер. Исходные данные берутся из HTML-документа, а результаты выполнения программы возвращаются в него. Программы на JavaScript называют сценариями или скриптами.

Так как сценарии передаются на компьютер пользователя по сети, то от их размера зависит время загрузки сайта. Часто сценарий служит для внесения очень простых изменений в HTML-документ, например, требуется изменить цвет или размер какого-нибудь объекта на экране. Поэтому сценарии, как правило, небольшие, часто состоят из нескольких строк. Формально на их размер никаких ограничений не накладывается.

**Создание нового окна**

Объект window находится в вершине дерева объектной модели W3C DOM и имеет много свойств и методов. В сценариях, особенно при отладке, часто используется метод alert. Свойства setInterval, clearInterval и setTimeout необходимы для оживления страницы, организации движения объектов. Чтобы не загромождать основное окно, часто для просмотра дополнительной информации (графиков, таблиц и т.д.) удобно открыть новое дополнительное окно. Для создания нового окна используется метод open, имеющий следующий синтаксис: var windowObject=window.open([параметры) Рассмотрим сценарий (пример 1), по которому окно с описываемым на странице графиком появляется при наведении курсора мышки на слова *на графике*.

<HTML>   
<HEAD><TITLE></TITLE></HEAD>   
<BODY>   
<h1 align=center>График открывается в новом окне</h1>

Зависимость y=f(x) <B onmouseover="gr('График 1','prim1.gif')" style="color:blue;"> на графике 1</b> имеет линейный характер.

<p> Изменение цены и объёмов продаж показаны

<B onmouseover="gr('График 2','prim2.gif')" style="color:green">

на графике 2</b>. Объёмы продаж серого цвета.

<P><font size=-1>

Примечание. Для просмотра графиков подведите курсор мышки к словам "на

графике"<BR> </p>

<script>

function gr(txt,fil)

{ grWin=window.open("","","width=650,height=400")

  grWin.document.write("<H1 align=center>"+ txt+

  "</h1><CENTER><IMG SRC='"+fil+"'></center>")

}

</script>

</BODY>

</HTML>

Первые два параметра, URL и название окна в примере оставлены пустыми. В третьем параметре заданы ширина и высота окна в пикселях.

|  |
| --- |
| http://i.voenmeh.ru/kafi5/Kam.loc/LabJS/Ris7_1.jpg |
| Рис. 5 |

**Задание.**

**Задача 1**. Создайте сайт, состоящий из двух страниц. Сайт служит для вывода таблицы значений тригонометрической функции (sin, cos или tg) в заданном диапазоне и с заданным шагом. На первой странице (рис. 5) пользователь задаёт исходные данные, а на второй получает соответствующую таблицу. Окно с новой страницей должно открываться методом open(). Исходные данные должны проверяться сразу после ввода и после нажатия кнопки Вычислить. Таблица должна иметь следующий вид:

|  |  |  |
| --- | --- | --- |
| Угол | | sin |
| в градусах | в радианах |
| 30 | 0.5236 | 0.5 |
| 32 | 0.5585 | 0.5299 |
| 34 | 0.5934 | 0.5592 |
| 36 | 0.6283 | 0.5878 |

*Указания*. Данная задача является расширением задачи 3.2. Для проверки цифровых символов служит следующая функция:

function isNumb(fld)

{ t=fld.value

 for (i=0;i<t.length;i++)

 { var ch=t.charAt(i)

if((ch<"0" ||ch>"9") & !(ch=="," ||ch=="."))

{ alert("Не цифра") fld.focus(); fld.select()

  return false

}

 }

 return true

}

Параметр fld – объект, значение которого проверяется. Пример обращения к функции isNumb(fld):

<INPUT TYPE="text" NAME="h" onchange=isNumb(this)>

Результат выполнения оператора ch=t.charAt(i) – i-й символ строковой переменной (объекта) t. Функция сообщает об ошибке, если введён любой символ кроме десятичных цифр, точки или запятой.

**Управление таблицей**

Для занесения данных в ячейку таблицы служит выражение:

document.all.tab1.rows[i].cells[j].innerHTML =x

//где tabl – id таблицы

Чтение данных из ячейки в браузере Internet Explorer симметрично занесению:

x = document.all.tab1.rows[i].cells[j].innerHTML

Браузер Mozilla не воспринимает этот оператор. Оба браузера правильно выполняют следующий оператор:

x = document.all.tab1.rows[i].cells[3].firstChild.nodeValue

Содержимое ячейки таблицы рассматривается как дочерний узел по отношению к узлу, соответствующему тегу <TD>. В примере извлекаются данные из i-й ячейки третьей строки таблицы, имеющей ID=tab1. Нумерация строк и ячеек начинается с нуля. С помощью сценария можно менять структуру таблицы, добавляя или удаляя строки и ячейки. Операция вставки строки в таблицу состоит из следующих этапов.

* Вызов метода insertRow() и сохранение возвращенной ссылки на новую незаполненную строку.
* Использование ссылки на строку при вызове метода insertCell() для каждой ячейки в строке (при этом сохраняется возвращенная ссылка на каждую новую незаполненную ячейку).
* Присвоение значений свойствам ячейки, включая ее содержимое.

Пример добавления строки и ячейки:

NewRow= document.all.tab1.insertRow() //вставка строки

NewCell=NewRow.insertCell(0) //вставка ячейки

Для удаления строк и столбцов служат методы deleteRow(индекс строки) и deleteCell(индекс ячейки). Пример удаления всех строк таблицы:

while (tableRef.rows.length > 0)

{tableRef.deleteRow(0);}

**Задача 2**. Создайте страницу (рис. 6) для учёта поступления товаров. Пользователь может менять в таблице количество и цену выбранного товара, вводя новые значения в поля, расположенные под таблицей. Введёнными значениями заменяются соответствующие данные в таблице и автоматически подсчитывается суммарная стоимость всех товаров.

|  |
| --- |
| http://i.voenmeh.ru/kafi5/Kam.loc/LabJS/Ris7_2.jpg |
| Рис. 6 |

**Задания.**

1. Треугольник. Напишите цикл, выводит такой треугольник:

#

##

###

####

#####

######

#######

1. FizzBuzz. Напишите программу, которая выводит через console.log все числа от 1 до 100, с двумя исключениями. Для чисел, нацело делящихся на 3, она должна выводить ‘Fizz’, а для чисел, делящихся на 5 (но не на 3) – ‘Buzz’.Когда сумеете – исправьте её так, чтобы она выводила «FizzBuzz» для всех чисел, которые делятся и на 3 и на 5.
2. Шахматная доска. Напишите программу, создающую строку, содержащую решётку 8х8, в которой линии разделяются символами новой строки. На каждой позиции либо пробел, либо #. В результате должна получиться шахматная доска.

# # # #

# # # #

# # # #

# # # #

# # # #

# # # #

# # # #

# # # #

1. Минимум. Напишите функцию min, принимающую два аргумента, и возвращающую минимальный из них.
2. Рекурсия. Ноль чётный. Единица нечётная. У любого числа N чётность такая же, как у N-2.Напишите рекурсивную функцию isEven согласно этим правилам. Она должна принимать число и возвращать булевское значение. Потестируйте её на 50 и 75. Попробуйте задать ей -1. Почему она ведёт себя таким образом? Можно ли её как-то исправить?
3. Считаем бобы. Символ номер N строки можно получить, добавив к ней .charAt(N) ( “строчка”.charAt(5) ) – схожим образом с получением длины строки при помощи .length. Возвращаемое значение будет строковым, состоящим из одного символа (к примеру, “к”). У первого символа строки позиция 0, что означает, что у последнего символа позиция будет string.length – 1. Другими словами, у строки из двух символов длина 2, а позиции её символов будут 0 и 1.Напишите функцию countBs, которая принимает строку в качестве аргумента, и возвращает количество символов “B”, содержащихся в строке.Затем напишите функцию countChar, которая работает примерно как countBs, только принимает второй параметр — символ, который мы будем искать в строке (вместо того, чтобы просто считать количество символов “B”). Для этого переделайте функцию countBs.
4. Сумма диапазона. Напишите функцию range, принимающую два аргумента, начало и конец диапазона, и возвращающую массив, который содержит все числа из него, включая начальное и конечное.Затем напишите функцию sum, принимающую массив чисел и возвращающую их сумму. Запустите указанную выше инструкцию и убедитесь, что она возвращает 55.В качестве бонуса дополните функцию range, чтобы она могла принимать необязательный третий аргумент – шаг для построения массива. Если он не задан, шаг равен единице. Вызов функции range(1, 10, 2) должен будет вернуть [1, 3, 5, 7, 9]. Убедитесь, что она работает с отрицательным шагом так, что вызов range(5, 2, -1) возвращает [5, 4, 3, 2].
5. Обращаем массив вспять. Напишите две функции, reverseArray и reverseArrayInPlace. Первая получает массив как аргумент и выдаёт новый массив, с обратным порядком элементов. Вторая работает как оригинальный метод reverse – она меняет порядок элементов на обратный в том массиве, который был ей передан в качестве аргумента. Не используйте стандартный метод reverse.
6. Список.Объекты могут быть использованы для построения различных структур данных. Часто встречающаяся структура – список (не путайте с массивом). Список – связанный набор объектов, где первый объект содержит ссылку на второй, второй – на третий, и т.п. Списки удобны тем, что они могут делиться частью своей структуры. Например, можно сделать два списка, {value: 0, rest: list} и {value: -1, rest: list}, где list – это ссылка на ранее объявленную переменную. Это два независимых списка, при этом у них есть общая структура list, которая включает три последних элемента каждого из них. Кроме того, оригинальный список также сохраняет свои свойства как отдельный список из трёх элементов.

Напишите функцию arrayToList, которая строит такую структуру, получая в качестве аргумента [1, 2, 3], а также функцию listToArray, которая создаёт массив из списка. Также напишите вспомогательную функцию prepend, которая получает элемент и создаёт новый список, где этот элемент добавлен спереди к первоначальному списку, и функцию nth, которая в качестве аргументов принимает список и число, а возвращает элемент на заданной позиции в списке, или же undefined в случае отсутствия такого элемента.

Если ваша версия nth не рекурсивна, тогда напишите её рекурсивную версию.

1. Глубокое сравнение. Оператор == сравнивает переменные объектов, проверяя, ссылаются ли они на один объект. Но иногда полезно было бы сравнить объекты по содержимому. Напишите функцию deepEqual, которая принимает два значения и возвращает true, только если это два одинаковых значения или это объекты, свойства которых имеют одинаковые значения, если их сравнивать рекурсивным вызовом deepEqual.Чтобы узнать, когда сравнивать величины через ===, а когда – объекты по содержимому, используйте оператор typeof. Если он выдаёт “object” для обеих величин, значит нужно делать глубокое сравнение. Не забудьте об одном дурацком исключении, случившемся из-за исторических причин: “typeof null” тоже возвращает “object”.