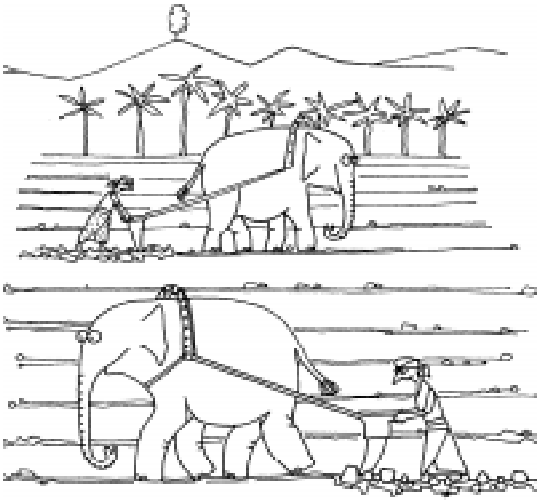
Дмитриева Марина Валерьевна

JAVASCRIPT.
ЗАНЯТИЕ 3. ОБЪЕКТЫ ЯЗЫКА JAVASCRIPT

ОБРАБОТКА СТРОКОВЫХ ДАННЫХ



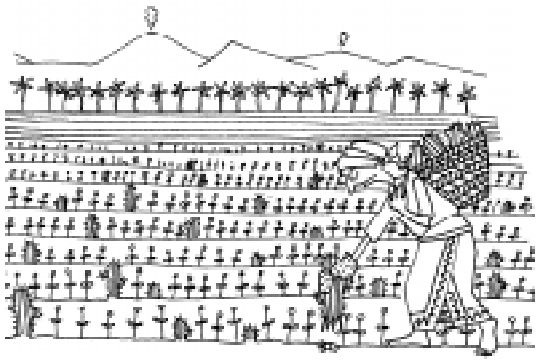
Строка является универсальным способом представления данных. Многие языки программирования позволяют обрабатывать строковые данные. Строка – конечная последовательность символов некоторого алфавита. Строковый литерал представляет собой последовательность символов, заключенную в одинарные или двойные кавычки. В строке важен порядок символов, так, строки 01 и 10 различны. Длина строки равна числу символов в строке. Пустая строка – это строка, не содержащая ни одного символа. Длина пустой строки равна нулю. В предыдущих примерах использовались строковые литералы, например, при выводе некоторой информации в документ **Document.write("Сумма равна", s)**.

Строковые литералы или строковые переменные являются в языке JavaScript объектом типа **string**, к которому могут быть применены методы, определенные в языке. Создание нового объекта требует вызова функции-конструктора объекта. Для того чтобы создать строковый объект надо применить конструктор **new String**, например, **s= new String ("Произведение=")**

Объект **string** имеет единственное свойство **length** (длина строки). Выражение **s.length** выдает значение 13, равное длине строки, содержащейся в строковом объекте **s**. Объект **string** имеет два типа методов. Мы познакомимся с одим из типов, который включает методы, непосредственно влияющие на саму строку.

Одним из часто используемых методов является метод выделения из строки отдельного символа. Метод **charAt(n1)** возвращает символ, позицию которого определяет параметр **n1**. Символы в строке перенумерованы, начиная с 0.

ПОИСК СЛОВА В ТЕКСТЕ



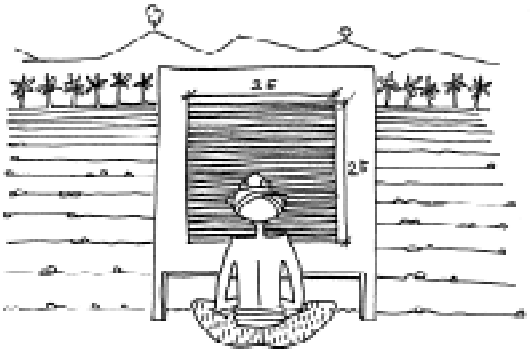
Напишем сценарий, который определяет, сколько раз заданное слово встречается в определенном тексте. В тексте слова разделяются пробелами. После того, как очередное слово найдено, просмотр продолжается с символа, следую-

щего за найденным словом. Метод **substr(i,m)** возвращает подстроку, которая начинается с заданной позиции (i), и содержит заданное количество символов (m). HTML-код документа хранится в листинге 1.

В языке JavaScript определены стандартные функции, при работе с которыми не требуется создавать никакого объекта. Функции осуществляют анализ своих аргументов.

Функция **Number(s)** преобразует строковый параметр **s** в число. Эту функцию уже неоднократно использовали в предыдущих сценариях. Функция **String(n)** преобразует число **n** в строку.

АВТОМОРФНЫЕ ЧИСЛА В ЗАДАННОМ ИНТЕРВАЛЕ



Напишем сценарий, определяющий все автоморфные числа в заданном интервале. Напомним, что натуральное чис-

Листинг 1. Количество заданных слов в тексте

```
<html>
<head>
<title>Количество заданных слов в тексте</title>
<script language="JavaScript">
<!-- //
function numword(obj)
{ var s=obj.textin.value // заданный текст
  var h=obj.data.value // слово
  s='  '+s+' '
  h=' '+h+' '
  var m=h.length // длина слова
  var res=0 // количество слов в тексте
       var i=0
     while ( i <= s.length-1)
        { ch=s.substr(i,m) // выделение подстроки
            if (ch==h) {res+=1; i = i+m-1}
            else
           i++
       }
obj.result.value=res
}
//->
</script>
</head>
<body   bgcolor="#FFFFCC">
<h4>Количество заданных слов в тексте</h4>
<form name="form1">
Введите текст:<br>
<textarea name="textin" rows=4 cols=20></textarea><hr>
Введите слово: <input type="text" name="data" size="8"><hr>
<input type="button" value="Определить" onClick="numword(form1)"><hr>
Количество слов в тексте: <input type="text" name="result" size=8><hr>
<input type="reset" value="Отменить">
</form></body></html>
```

ло называется автоморфным, если оно содержится в качестве младших цифр в своем квадрате. Например, число 25 является автоморфным, так как 25^2=625.

Опишем функцию **avtomorf**, которая преобразует введенную строку в число, находит квадрат числа. Далее работа осуществляется со строками. Выделяются последние символы строки, представляющей квадрат числа, и сравниваются со строкой, соответствующей числу.

Функция **interav** обеспечивает перебор всех натуральных чисел в заданном диапазоне. Для каждого из чисел диапазона проверяется, является ли оно автоморфным, и если это так, то формируется строка результата. В строку результата заносится само число и его квадрат. Кроме того, информация о каждом числе располагается на отдельной строке. Строка результата помещается в текстовое поле.

Результат работы сценария для интервала [20; 9999] изображен на рисунке 1.

Сценарий, реализующий поиск автоморфных чисел в заданном интервале, и HTML-документ представлен в листинге 2.

Функция **parseInt(s,n)** возвращает целое число по основанию, заданному в качестве второго параметра. Если первый символ в строке не является цифрой в указанной вторым параметром системе счисления, то функция **parseInt(s,n)** возвращает значение **"NaN"**. Функция

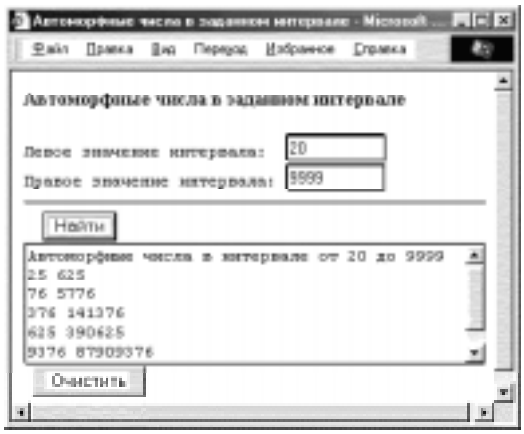


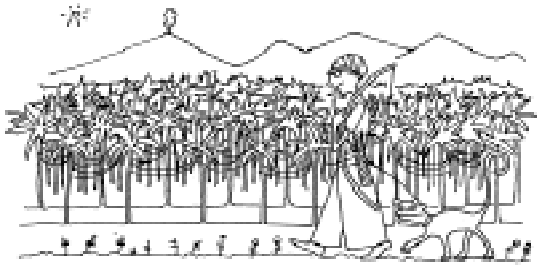
Рисунок 1. Автоморфные числа в заданном диапазоне

parseFloat(s) анализирует значение строкового параметра **s** и определяет, соответствует ли строка **s** представлению вещественного числа. Если в строке содержится символ, отличный от символов, разрешенных при формировании вещественного литерала, то оставшаяся часть строки игнорируется, а в качестве результата функции возвращается числовое значение, которое обнаружено до неправильного символа. Если первый символ в строке не является цифрой, то функция **parseFloat(s)** возвращает значение **"NaN"**.

Функция **isNaN(s)** проверяет, является ли параметр **s** числом. Если параметр **s** не является числом, то функция возвращает значение **true**, в противном случае - значение **false**.

Методы обработки строк позволяют осуществить детальную проверку вводимых данных в зависимости от формата, определяемого разработчиком формы или принятыми соглашениями.

МАССИВЫ. ОБЪЕКТ ARRAY



Массив представляет собой набор элементов, доступ к которым осуществляется по индексу. Массив создается оператором **new** и конструктором массива – системной функцией **Array**. Создать массив из названий дней недели можно, например, следующим образом:

```
var NDays = new Array ("воскресенье",
"понедельник", "вторник", "среда",
"четверг", "пятница","суббота").
```

В качестве параметров конструктору передаются значения элементов масси-

Листинг 2. Автоморфные числа в заданном интервале

```
<html>
<head>
<title>Автоморфные числа в заданном интервале</title>
<script language="JavaScript">
<!--
// перебор чисел в заданном интервале
function  interav (obj )
{ var s=""
   var l=obj.left.value // нижняя граница
   var r=obj.rig.value // верхняя граница
   var s="Автоморфные числа в интервале от "+l+" до " +r+"\r\n"
   for (var i=l; i<=r; i++)
    if ( avtomorf (i)) // проверка, является ли число автоморфным
     s +=i+" "+i*i+"\r\n"
    obj.result.value=s
}
// функция проверяет, является ли число автоморфным
function  avtomorf (a )
 {  var p=a*a
    var sa= String(a)// строка, соответствующая числу
    var sp= String(p) // строка, соответствующая квадрату числа
    var nsa= sa.length // количество цифр в числе
    var nsp= sp.length // количество цифр в квадрате числа
      endsp=sp.substr(nsp-nsa,nsa) // выделение последних разрядов
квадрата
    return (endsp==sa)
    }
//–>
</script>
</head>
<body>
<h4>Автоморфные числа в заданном интервале</h4>
<form name="form1">
<pre>
Левое значение интервала:  <input type="text" size=10 name="left">
Правое значение интервала: <input type="text" size=10 name="rig"><hr>
  <input type="button" value=Найти  onClick=" interav(form1)">
<textarea cols=45 rows=5 name=result></textarea>
 <input type="reset"  value=Очистить>
</pre>
</form></body></html>
```

ва. Можно создать массив, указав в нем лишь число элементов, например, так:

```
var Ndays1= new Array (7).
```

И, наконец, можно использовать конструктор без параметра

```
var Ndays2= new Array().
```

В этом случае определяется лишь, что переменная **Ndays2** используется в качестве массива. Все элементы массива перенумерованы, начиная с нуля. Для получения значения элемента массива необходимо задать имя массива и в квадратных скобках порядковый номер элемента. Для того чтобы получить доступ к первому элементу массива следует воспользоваться конструкцией **Ndays[0]**. Свойство **length** позволяет определить число элементов в массиве или, как говорят, длину массива. Доступ к последнему элементу массива

можно осуществить, например, следующим образом: **Ndays[Ndays.length-1]**.

Объект **Array** обладает методами, которые позволяют обрабатывать как элементы одного массива, так и двух. Можно отсортировать элементы массива, переставить элементы массива в обратном порядке, т.е. «перевернуть» массив, добавить или удалить элемент из массива, соединить элементы массива в одну строку, объединить два массива в один.

ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ДАТ. ОБЪЕКТ DATE

При решении многих задач важно уметь хранить и обрабатывать данные, связанные с датой и временем. Например, в базах данных, хранящих информацию о сотрудниках, как правило, содержатся сведения о дате рождения, дате заключения контракта, датах поощрений и т. д. В базах данных гостиниц и пансионатов хранится информация о датах проживания клиентов.

JavaScript располагает средствами, позволяющими работать с датой и временем. Встроенный объект **Date** применяется для представления и обработки даты и времени. Он не имеет свойств, но имеет несколько методов, позволяющих устанавливать и изменять дату и время. В языке JavaScript дата определяется числом миллисекунд, прошедших с 1 января 1970 года.

Объект **Date** создается оператором **new** с помощью конструктора **Date**. Если в конструкторе отсутствуют параметры, то значением **new Date()** будет текущая дата и время. Значением переменной **my_date1**, определенной следующим образом: **var my_Date1 = new Date()** будет объект, соответствующий текущей дате и времени.

Параметром конструктора **new Date** может быть строка следующего формата **"месяц, день, год часы:минуты:секунды"**. Опишем переменную **my_Date2** и присвоим ей начальное значение:

```
var my_Date2 =
new Date ("Fv, 15,  1995 16:45:10").
```

Переменная **my_Date2** определяет дату 15 февраля 1995 года и время 16 часов 45 минут и 10 секунд. Значения часов, минут, секунд можно опустить, в этом случае они будут иметь значение ноль:

```
var my_Date3 = new Date ("Fv, 15, 1995").
```

Параметры конструктора **new Date** могут определять год, месяц, число, время, минуты, секунды с помощью чисел. Дату 15 февраля 1995 года и время 16 часов 45 минут и 10 секунд можно задать следующим образом:

```
var my_Date4 = new Date (95,1,15,16,45,10).
```

Если время опустить, то описание будет следующим:

```
var my_Date5 = new Date (95,1,15).
```

Все числовые представления даты нумеруются с нуля, кроме номера дня в месяце. Месяцы представляются числами от 0 (январь) до 11 (декабрь), поэтому второй параметр при задании переменных **my_Date4** и **my_Date5** равен 1.

Методами объекта **Date** можно получать и устанавливать отдельно значения месяца, дня недели, часов, минут и секунд. Метод **getMonth** возвращает номер месяца в году как целое число в интервале между 0 (январь) и 11 (декабрь). Метод **getDate** возвращает число в диапазоне от 1 до 31, представляющее число месяца. Метод **getHours** возвращает час суток. Значение возвращается в 24-часовом формате от 0 (полночь) до 23. Метод **getMinutes** возвращает минуты как целое от 0 до 59.

Метод **getSeconds** возвращает число секунд как целое от 0 до 59.

В следующем примере эти методы используются для формирования текущего времени.

ОПРЕДЕЛЕНИЕ ТЕКУЩЕГО ВРЕМЕНИ

Напишем сценарий, который определяет текущее время и выводит его в текстовое поле в следующем формате **чч:мм:сс**.

В переменной **res** формируется строка, которая затем будет отображена в поле **rest** формы с именем **form1**. Для того чтобы уточнить время следует еще раз нажать на кнопку время и т. д. HTML-код представлен в листинге 3.

ЭЛЕКТРОННЫЕ ЧАСЫ

Листинг 3. Определение времени

```
<html>
<head>
<title>Определение времени</title>
<script language= "JavaScript">
<!--
// Вывод времени в заданном формате
function cl ()
   { var d=document
     var t = new Date () // текущая дата
     var h = t.getHours () // часы
     var m = t.getMinutes () // минуты
     var s = t.getSeconds ()  // секунды
     var res = ""
    if (h < 10)
      res += "0" + h
    else  res += h
    if (m < 10)   res += ":0"+m
    else res += ":"+m
    if (s  < 10)   res += ":0"+s
    else res += ":"+s
     d.form1.rest.value = res
}
//–>
</script>
</head>
<body  bgcolor="#FFFFCC">
<center>
<img src=alarmWHT.gif >
<h4>При нажатии на кнопку время, Вы узнаете, который час</h4>
<form name="form1">
  <input type="button" value=время   onClick="cl()">
 <input type="text" size=10 name="rest" ><br>
</form></body></html>
```

Напишем сценарий, который позволяет разместить на странице электронные часы. Вид часов изображен на рисунке 2.

Для изображения времени требуется подготовить рисунки для цифр. Предположим, что рисунки хранятся в текущей папке в файлах с именами g0.bmp – g9.bmp. Сначала в сценарии формируются три строки так, чтобы для каждого зна-

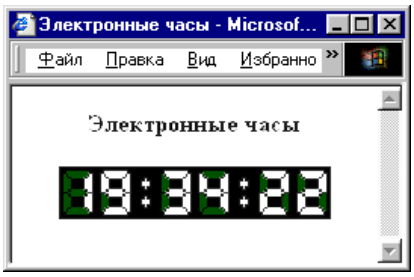


Рисунок 2. Электронные часы

Листинг 4. Электронные часы

```
<HTML>
<HEAD>
<TITLE>Электронные часы</TITLE>
<script>
<!--//
  function eclock()
   {   var d=document
       var t=new Date() // текущая дата
       var h=t.getHours() // часы
       var m=t.getMinutes()  // минуты
       var s=t.getSeconds() // секунды
      if (h<10)  h="0"+h
      else h= String(h)
      if (m<10)  m="0"+m
      else m=String(m)
      if (s<10)  s="0"+s
      else s = String(s)
      d.h1.src="g"+h.charAt(0)+".gif"
      d.h2.src="g"+h.charAt(1)+".gif"
      d.m1.src="g"+m.charAt(0)+".gif"
      d.m2.src="g"+m.charAt(1)+".gif"
      d.s1.src="g"+s.charAt(0)+".gif"
      d.s2.src="g"+s.charAt(1)+".gif"
      setTimeout("eclock()",1000)
    }
//-->
</script>
</HEAD>
<BODY onLoad="eclock()">
<h4 align=center> Электронные часы</h4>
<table  width="50"  border="0"  cellspacing="0"  cellpadding="0"
align="center">
  <tr>
    <td ><img src="g0.gif" height=35 width=25 name=h1></td>
    <td ><img src="g0.gif" height=35 width=25 name=h2></td>
    <td ><img src="gc.gif" height=35 width=15 name=c></td>
    <td ><img src="g0.gif" height=35 width=25 name=m1></td>
    <td><img src="g0.gif" height=35 width=25 name=m2></td>
    <td ><img src="gc.gif" height=35 width=15 name=c></td>
    <td ><img src="g0.gif" height=35 width=25 name=s1></td>
    <td><img src="g0.gifм height=35 width=25 name=s2></td>
  </tr>
</table></BODY></HTML>
```

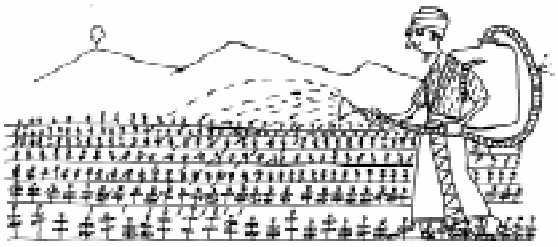
чения отводилось две позиции. Затем по строке строится изображение. HTML-код документа приведен в листинге 4.

Мы рассмотрели лишь простые примеры, в которых информация о дате и времени помещалась на Web-страницу. Для этого использовали методы, позволяющие получить значения объекта **Date**.

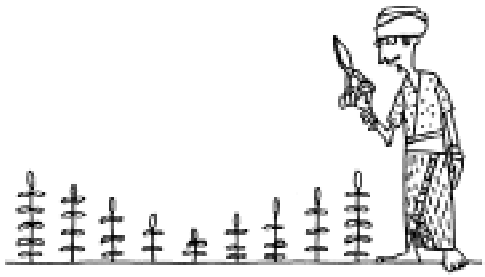
Следующие методы позволяют устанавливать различные значения для объекта **Date**.

- Метод **setYear()** устанавливает значение года для объекта **Date**.
- Метод **setDate()** устанавливает день месяца. Параметр должен быть числом в диапазоне от 1 до 31.
- Метод **setMonth()** устанавливает значение месяца. Параметр должен быть числом в диапазоне от 0 (январь) до 11 (декабрь).
- Метод **setHours()** устанавливает час для текущего времени, использует целое число от 0 (полночь) до 23 для установки даты по 24 часовой шкале.
- Метод **setMinuts()** устанавливает минуты для текущего времени, использует целое число от 0 до 59.
- Метод **setSeconds()** устанавливает секунды для текущего времени, использует целое число от 0 до 59.
- Метод **setTime()** устанавливает значение объекта **Date**. Метод возвращает количество миллисекунд, прошедших с 1 января 1970 года.

Задания



1. Слова в заданном тексте разделяются пробелами. Напишите сценарий, который определяет количество слов в тексте.



2. Напишите функцию, определяющую, является ли заданное предложение палиндромом. Палиндромом считается строка, которая читается одинаково как слева на право, так и справа налево. Палиндромами являются слова: казак, кок, шалаш и др. Палиндромами могут быть и фразы: обычно считается, что пробелы между словами, знаки препинания и различия между маленькими и большими буквами игнорируются. Например, палиндромами являются фразы:

«а роза упала на лапу Азора»,

«кит на море не романтик».

3. Напишите сценарий, определяющий все числа Армстронга, расположенные в заданном интервале. Натуральное число называется числом Армстронга, если оно совпадает с суммой k-ых степеней составляющих его цифр, где k – количество цифр в числе. Например, таким числом является число 153, так как $153 = 1^3 + 5^3 + 3^3$.

4. Напишите сценарий, который определяет, сколько рабочих и выходных дней между двумя заданными датами.

5. Напишите сценарий, который по дате рождения человека определяет, к какому знаку зодиака относится человек.

6. В анкете приводятся данные о семи сотрудниках: фамилия и дата приема на работу. Напишите сценарий вычисления стажа работы (в годах).

7. В анкете приводятся данные о десяти сотрудниках: фамилия, дата рождения и оклад. Все сотрудники в зависимости от возраста разделяются на категории: менее 30 лет – «молодой», от 30 до 55 – «средний возраст», более 55 – «старший возраст». Напишите сценарий определения возраста каждого сотрудни-

ка и размер средней зарплаты в каждой категории. Постройте диаграмму, отражающую среднюю зарплату в каждой из трех категорий.

8. Напишите сценарий, с помощью которого определяются все даты в заданном году, приходящиеся на пятницу 13 числа.

9. Напишите сценарий, который по введенной дате формирует листок календаря с рисунком, соответствующим времени года.

10. В старояпонском календаре был принят 60-летний цикл, состоящий из пяти 12-летних подциклов. Подциклы обозначались названиями цвета: зеленый, красный, желтый, белый, черный. Внутри каждого подцикла годы носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезъяны, курицы, собаки и свиньи. Например, 1984 год (год зеленой крысы) был началом очередного цикла. Напишите сценарий, который по заданной дате определяет название года по старояпонскому календарю.

Дмитриева Марина Валерьевна, доцент кафедры информатики математико-механического факультета Санкт-Петербургского государственного университета.