Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования «Томский государственный университет систем управления и радиоэлектроники» (ТУСУР)

Программирование и программное обеспечение проектной деятельности

Методические указания по выполнению практических работ и заданий самостоятельной подготовки для студентов ВУЗа

Томск
2018

РАССМОТРЕНО И УТВЕРЖДЕНО на заседании кафедры экономической математики, информатики и статистики факультета вычислительных систем ТУСУР.
Протокол № 10 от «23» апреля 2018 г.

Методические указания направлены на изучения вопросов программирования на языке Javascript.

Составитель:
старший преподаватель кафедры ЭМИС Матолыгин А.А.

СОДЕРЖАНИЕ

1. Цели и задачи дисциплины

1.1. Цели дисциплины

Формирование знаний, умений и навыков решения стандартных задач профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.

Формирование навыков документального оформления решений в управлении операционной (производственной) деятельности организаций при внедрении технологических, продуктовых инноваций или организационных изменений.

Формирование навыков количественного и качественного анализа информации при принятии управленческих решений, построения экономических, финансовых и организационно-управленческих моделей путем их адаптации к конкретным задачам управления.

1.2. Задачи дисциплины

Научить студентов применять имеющиеся на рынке программных продуктов элементы информационных систем и информационные технологии в своей профессиональной деятельности. Сформировать знания по построению управленческих информационных систем, по технологиям автоматизации решения профессиональных задач и получить навыки и умения программирования при решении профессиональных задач

2. Требования к результатам освоения дисциплины

Процесс изучения дисциплины направлен на формирование следующих компетенций:

– ОПК-7 способностью решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности;

– ПК-8 владением навыками документального оформления решений в управлении операционной (производственной) деятельности организаций при внедрении технологических, продуктовых инноваций или организационных изменений.

В результате изучения дисциплины обучающийся должен:

– **знать** - структуру управленческой информационной системы;
- информационные технологии автоматизации решения управленческих задач;
- программное обеспечение профессиональной деятельности имеющееся на современном рынке;

– **уметь** - использовать элементы управленческих информационных систем для решения профессиональных задач; - составлять вычислительные программы для решения профессиональных задач;

– **владеть** - навыками применения алгоритмических языков программирования при решении практических задач.

3. Практическая работа №1 «Введение в JavaScript»

Цель работы: научиться формировать html-документ и изучить основы Javascript.

Web-документ, отображаемый браузером, – это результат исполнения программ, созданных на разных языках. Для описания структуры используется язык разметки (xhtml), для описания внешнего вида – язык стилей (css). Для описания поведения документа, его реакции на действия пользователя используется язык сценариев (javascript).

Подключение к странице

Исполняет JavaScript-код браузер. В него встроен интерпретатор JavaScript. Следовательно, выполнение программы зависит от того, когда этот интерпретатор получает управление. Опишем несколько способов размещения кода JavaScript на странице:
1. В теговом контейнере <BODY>...</BODY>.

```
<body>
…
<script > команды скрипта</script>
…
</body>
```

2. В теговом контейнере <HEAD>...</HEAD> - если код скрипта представляет собой функцию, которая вызывается в ответ на какое-либо событие.

```
<head>
…
<script type=”text/javascript”> Здесь находятся команды сценария </script>
…
</head>
```

3. Во внешнем файле. По аналогии с тем, как стили подключаются к станице с помощью элемента link, сценарии подключаются с помощью элемента script, только файл имеет расширение не .css, а .js.

```
<head>
…
<script type=”text/javascript” src=”my.js” > </script>
…
</head>
```

4. Обработчик события указывается прямо в теге, без заключения в теги <script > </script>

```
<input type="button" value="Нажать" onClick="window.alert('Нажмите еще раз')">
```

Выполнение операторов сценария

Существует несколько способов определения момента запуска сценария. Вот некоторые из них:
1. При загрузке документа;
2. Сразу после загрузки документа;
3. В ответ на действия пользователя.

Синтаксис языка

JavaScript –**зависит от регистра**. Имена JavaScript и Javascript - разные имена!! Все ключевые слова используют только нижний регистр.

Требования к именам переменных такие же, как в Паскале.

Операторы разделяются точкой с запятой, которую можно опустить, если оператор заканчивается символом новой строки (Enter).

Комментарии:

```
// однострочный комментарий,
/*
..многострочный комментарий
*/
```

Типы данных

Переменные не имеют строгой типизации. Объявляются с помощью оператора var, который можно опускать, за исключением объявления локальных переменных в теле функции. Возможно объявление с одновременной инициализацией, например:

var s = 123 //объявляется целочисленная переменная x, имеющая десятичное значение 123
var d=3.14 //объявляется переменная с плавающей точкой имеющая десятичное значение
var str1=’Строковая переменная’
var p=true //объявляется логическая переменная

Тип переменной может изменяться в процессе выполнения программы. Если в выражении содержатся и числовые и строковые переменные, то числовые переменные автоматически приводятся к строковому виду (таблица 3.1).

Математические операции

Математические операции представлены в таблице 3.1.

Таблица 3.1 – Математические операции

Оператор или операция	Действие	Пример	Значение, которое примет X
+	Сложение	x=100+5 str2=’Начало’ str1=str2+’ конец’	105 Начало Начало конец
-	Вычитание	X=100-5	95
*	Умножение	X=2*3	6
/	Деление	X=12/2	6
%	Остаток от деления (аналогично mod)	X=16%3	1
++	Значение увеличивается на 1	X=2; X++;	3
--	Значение уменьшается на 1	X=2; X– –;	1

Операции сравнения

Операции сравнения представлены в таблице 3.2.

Таблица 3.2 – Операции сравнения

Операция	Действие	Пример	Аналогично, в Паскале
= =	Равно	X==10	X=10
!=	Не равно	X!=5	X<>5
>	Больше	X>0	X>0
<	Меньше	X<4	X<4
>=	Больше либо равно	X>=Y	X>=Y
<=	Меньше либо равно	X<=5	X<=5

Логические операции

Логические операции представлены в таблице 3.3.

Таблица 3.3 – Логические операции

Операция	Действие	Пример	Аналогично, в Паскале
&&	Аналогично логической операции and	X>=2 && y>=2	(X>=2)and(Y>=2)
||	Аналогично логической операции or	x>0 || y>0	(x>0)or(y>0)
!	Аналогично логической операции not	!(1 < x && x < 10)	Not((1 < x) and (x < 10))

Операторы присваивания

Операторы присваивания представлены в таблице 3.4.

Таблица 3.4 – Операторы присваивания

Оператор	Действие	Пример	Значение, которое	Аналогично, в Паскале

			примет X	
=	Присваивает значение переменной	X=1000;	1000	X:=1000;
+=	Увеличивает значение переменной на указанную величину	X=1000; X+=100;	1100	X:=1000; X:=X+100;
-=	Уменьшает значение переменной на указанную величину	X=1000; X-=12;	988	X:=1000; X:=X-12;
*=	Умножает значение переменной на указанную величину	X=1000; X*=2;	2000	X:=1000; X:=X*2;
/=	Делит значение переменной на указанную величину	X=1000; X/=2;	500	X:=1000; X:=X/2;
%=	Делит значение переменной на указанную величину и возвращает остаток	X=1000; X%=5;	0	X:=1000; X:=X mod 5;

Условный оператор

Синтаксис условного оператора для Javascript в сравнении со синтаксисом Паскаля

JavaScript	Pascal
if (a==2) z=2; else z=3	if a=2 then z:=2 else z:=3;

if (x>=2 && x<=6)
{y=0; z=1}
else
{y=1; z=0}
	if (x>=2)and(x<=6) then
	begin y:=0; z:=1; end
	else begin y:=1; z:=0; end;

Отличия от Паскаля.

1. Then отсутствует.
2. Точка с запятой перед else ставится, если else находится на той же строке.
3. Все условие берется в скобки. Простые условия в скобки можно не брать.
4. Вместо операторных скобок (begin end) с той же целью используются фигурные скобки.

Ввод/вывод данных. Диалоговые окна

В JavaScript существует 3 функции (метода), позволяющие пользователю выводить диалоговые окна:

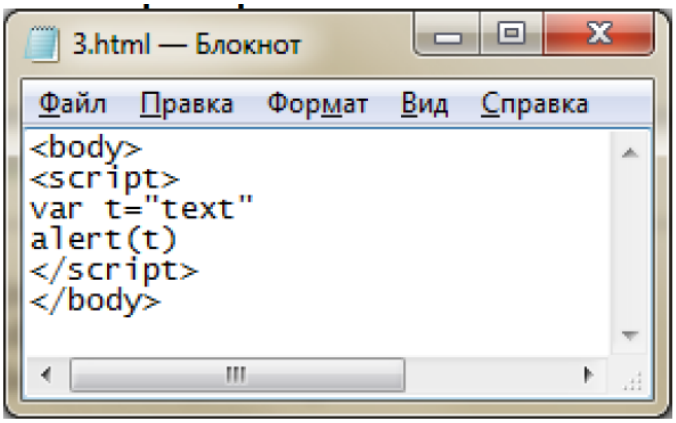
- alert
- confirm
- prompt

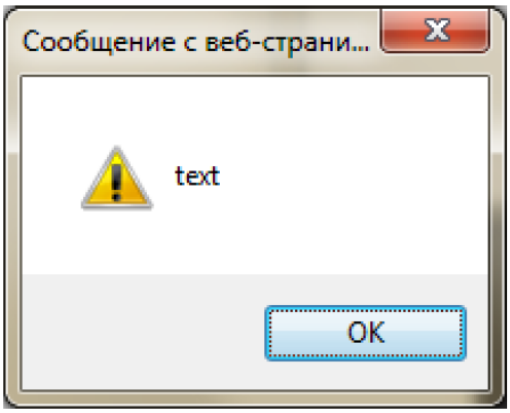
alert (“строка”)

Метод alert используется для вывода простейшего диалогового окна, содержащего текст сообщения и единственную кнопку "Ok". Программа выводит сообщение и ожидает нажатия кнопки. После нажатия на кнопку, программа начинает выполняться дальше.

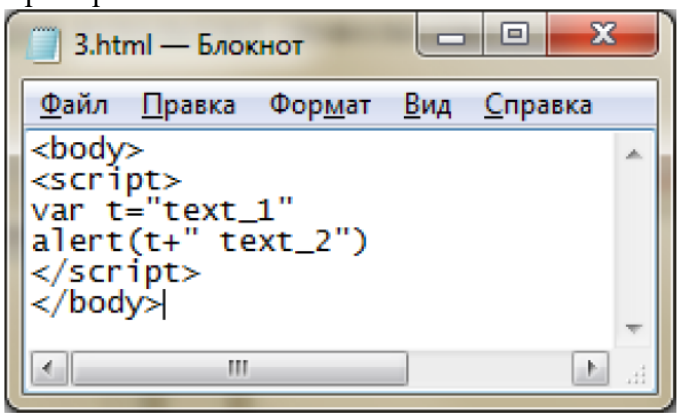
Текст сообщения может сцепляться с любой текстовой переменной с помощью знака «+». Чтобы текст выводился в несколько строк используют символы «\n».

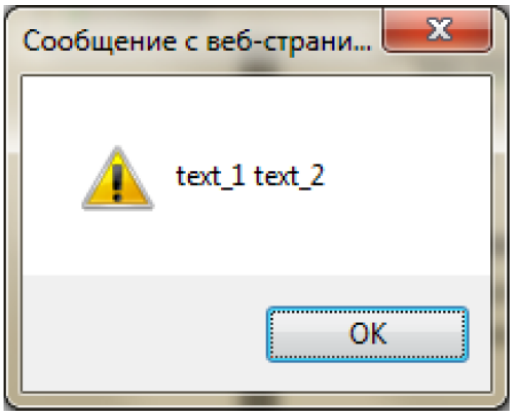
Пример 1.



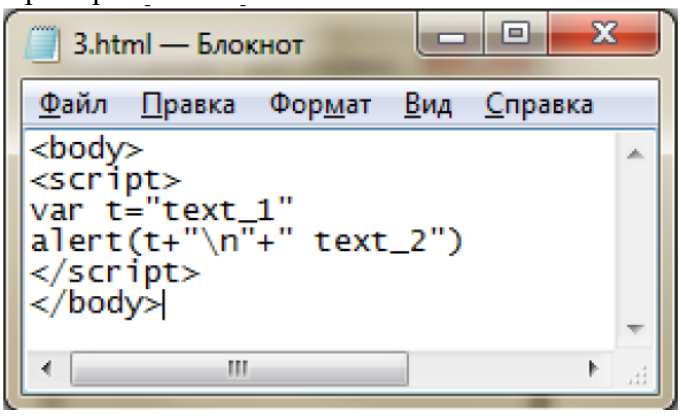


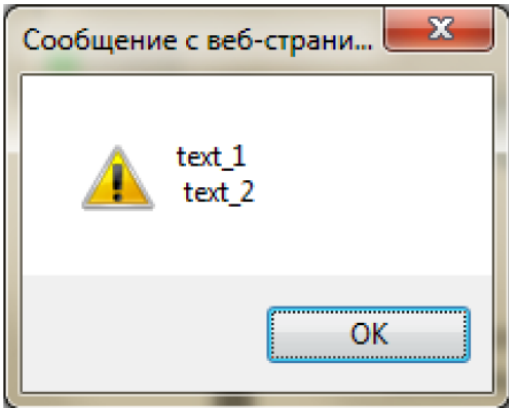
Пример 2.





Пример 3.





Confirm (“строка”)

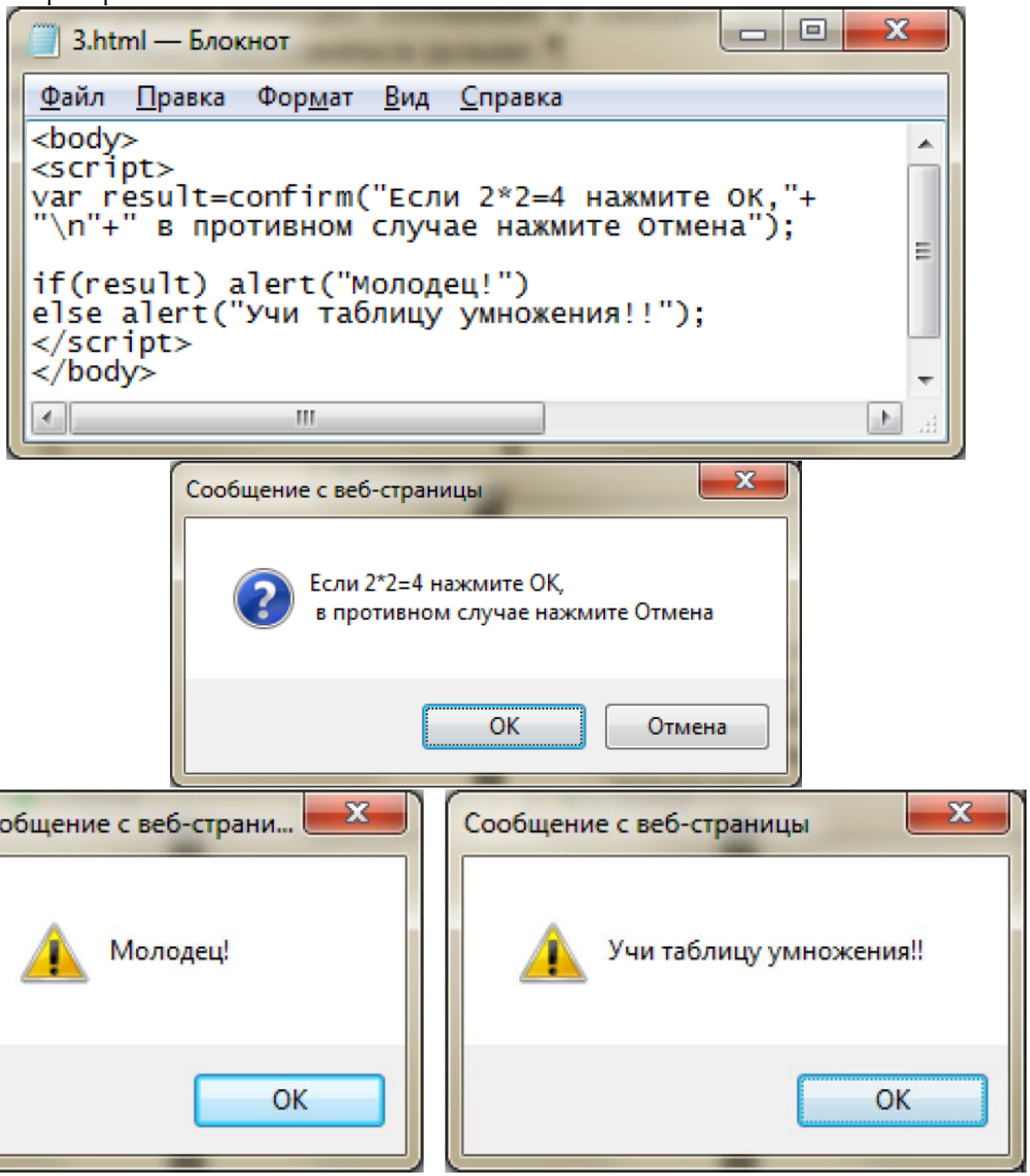
Метод confirm используется в тех случаях, когда пользователь должен сделать выбор. Метод confirm позволяет пользователю вывести диалоговое окно, содержащее текст вопроса и кнопки "ОК" и "Отмена". Функция confirm возвращает логическое значение в зависимости от нажатой пользователем кнопки:

• "ОК" соответствует значению true,

• "Отмена" - значению false.

Как правило, результат работы функции присваивают логической переменной, для дальнейшего анализа, как это показано в примере.

Пример.



prompt (“строка1”, “строка2”)

Метод prompt используется в тех случаях, когда пользователю нужно ввести значение в переменную.

В окно выводится сообщение «строка1», в поле ввода помещается умалчиваемое значение «строка2».

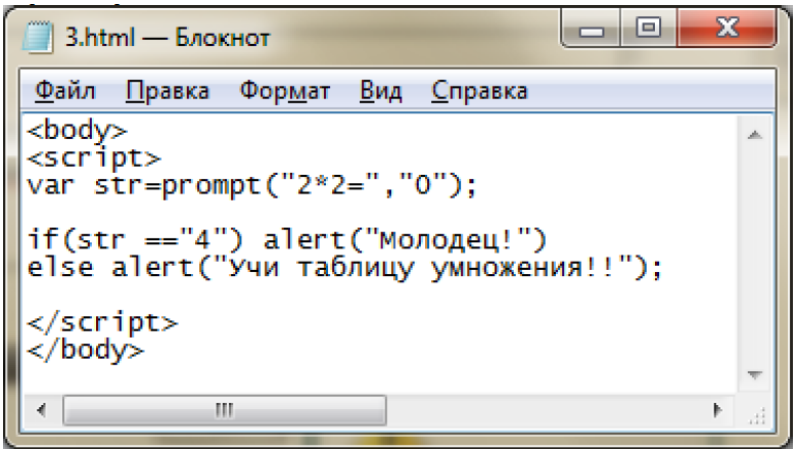
Этот метод позволяет вывести диалоговое окно запроса на ввод данных. Результат работы функции присваивают переменной строкового типа.

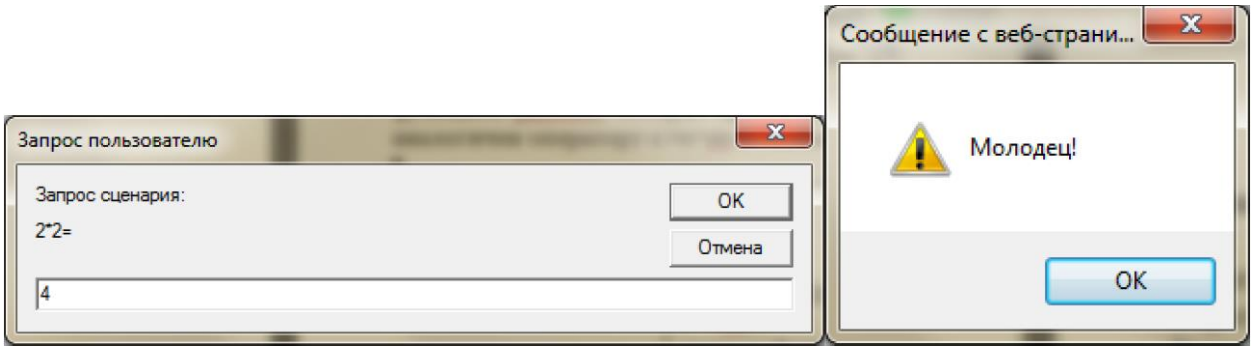
Если введенные данные нужно использовать в арифметических выражениях, необходимо выполнить преобразование введенной строки к числовому типу. Это можно сделать при помощи следующих функций:

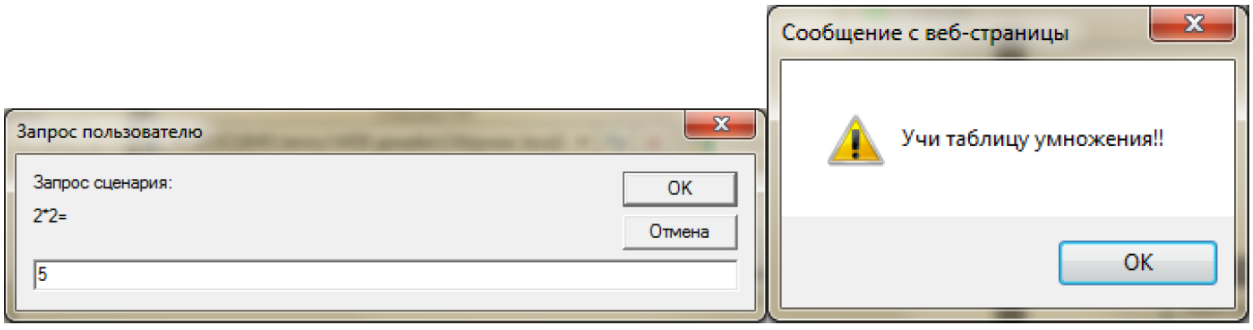
parseInt("строка") - преобразует строку в целое число;

parseFloat("строка") - преобразует строку в число с плавающей точкой.

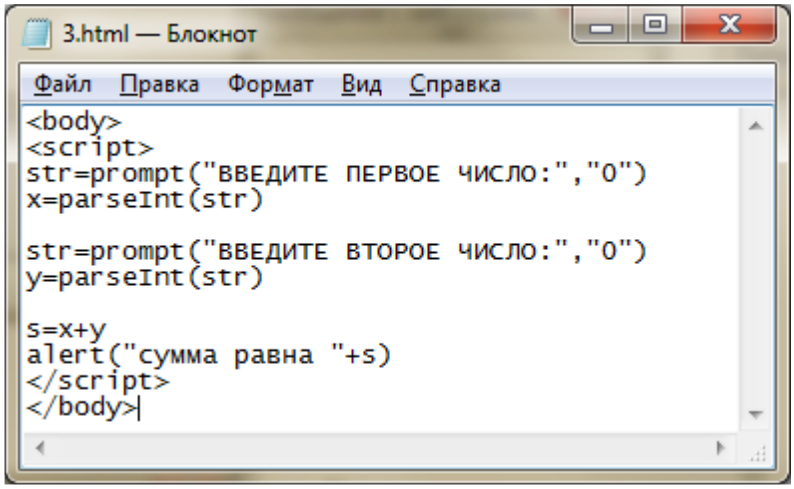
Пример 1. Задача на умножение 2*2.





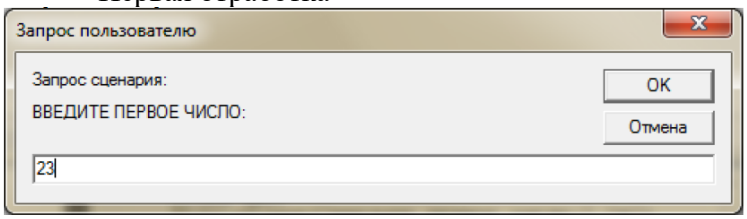


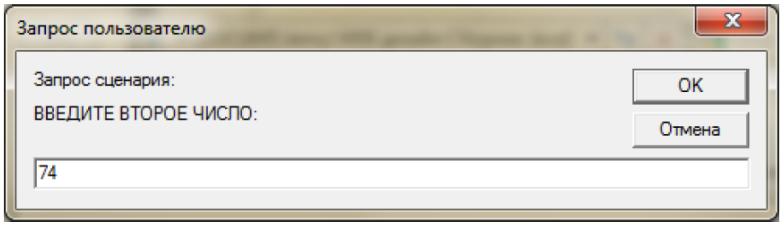
Пример 2.
Введем два числа, найдем их сумму и выведем ее на экран.

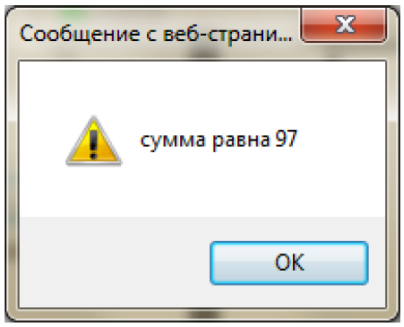


В данном скрипте переменные не описываются, что допускается. Переменная str будет иметь строковый тип, так как результат функции prompt должен быть строкового типа. Функция parseInt преобразовывает переменную str строкового типа в переменную x числового типа. Переменная s в операторе присваивания имеет числовой тип, так как переменные x и y имеют числовой тип. Переменная s в функции alert будет преобразована в строковый тип, так как параметр этой функции должен быть строкой.

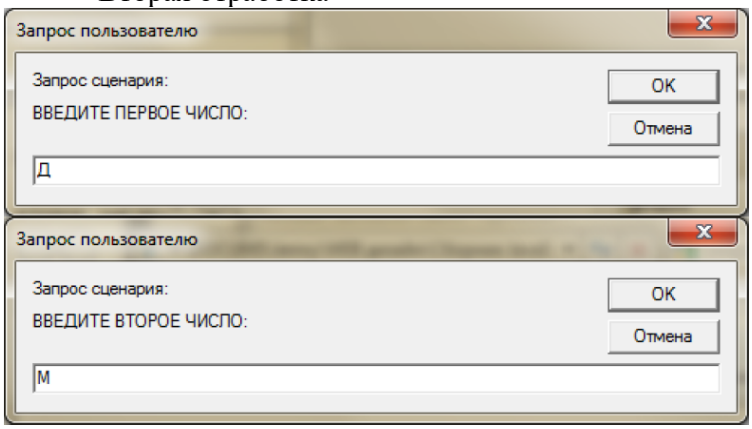
Первая отработка

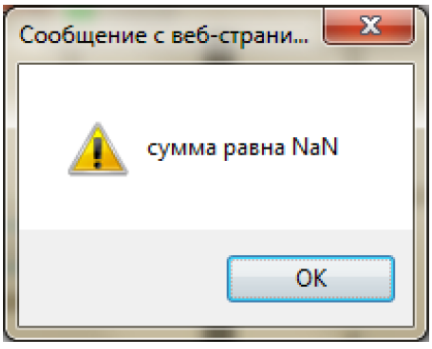






Вторая отработка





Буквы не могут быть преобразованы в числа. Поэтому переменные x, y и s не будут иметь значений.

Когда переменная не имеет значения, то выводится NaN.

Метод document.write()

JavaScript это объектно-ориентированный язык. Основной единицей в объектно-ориентированном языке является объект, который объединяет в себе данные (свойства) и средства обработки этих данных (методы). Если говорить образно, то объекты – это

«существительные», свойства объекта – это «прилагательные», а методы объекта – это «глаголы». Значения свойств объектов можно изменять.

Про JavaScript говорят, что в нем все объект. А именно: объектами являются окно, в котором открывается документ, сам документ, все элементы документа и даже свойства этих элементов. Есть также специальные встроенные объекты. Для упорядочивания огромного количества объектов создатели браузеров придумали объектную модель документа. Эта модель является структурой организации объектов на странице.

Объект document соответствует всему HTML-документу.

Изучим один метод этого объекта, позволяющий динамически формировать документ.

Метод document.write(“строка html-кода”) - выводит строку в окно документа.

Метод document.writeln (“строка html-кода”) - выводит строку в окно документа, в конце выводится символ "пробел".

Метод, применяемый к объекту, пишется после имени объекта через точку.

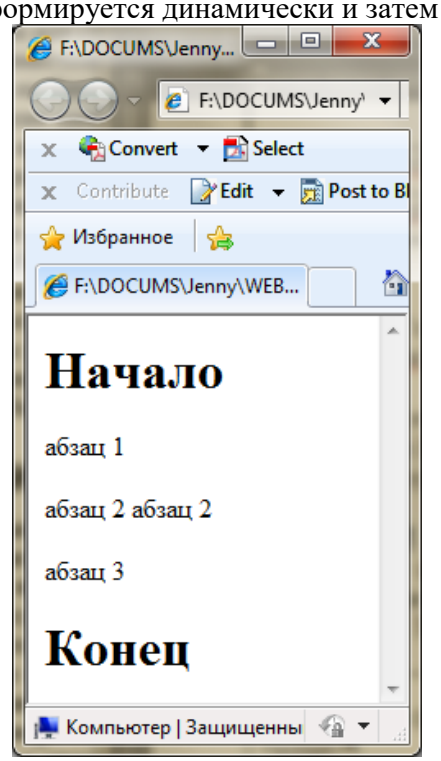
Содержимое строки должно быть в кавычках или это может быть объединение (сумма) нескольких строк или строковых переменных.

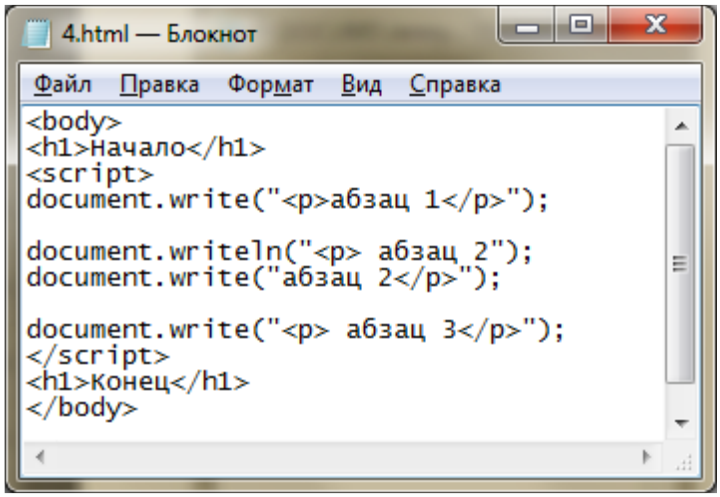
Строка должна содержать элементы разметки страницы (теги и их содержимое).

Метод исполняется в процессе загрузки документа.

Пример 1.

В данном примере заголовки (Начало и Конец) находятся в документе, а текст абзацев формируется динамически и затем выводится в документ.





Синтаксис языка

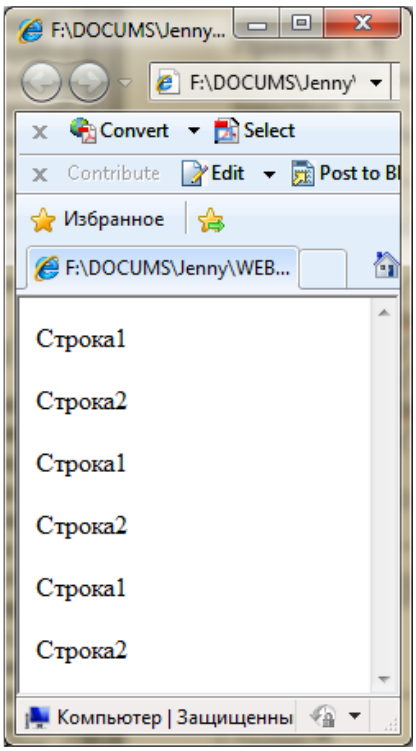
Оператор цикла for

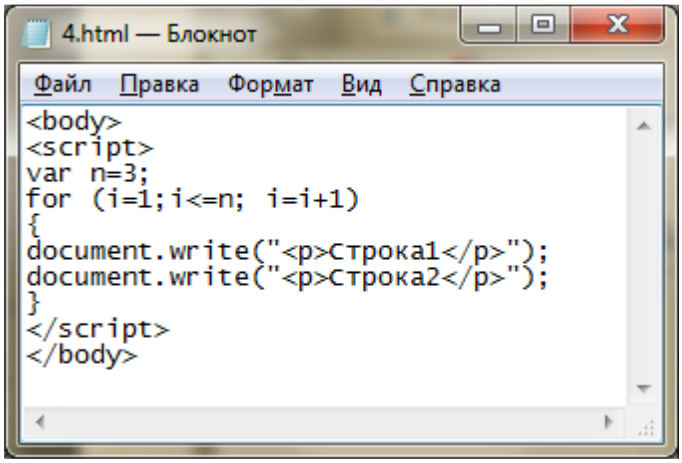
Отличия от Паскаля.

1. Возможен цикл с любым шагом.
2. Отсутствует do.
3. Начальное и конечное значение параметра, а также шаг указываются в скобках в заголовке цикла.
4. Вместо операторных скобок (begin end) с той же целью используются фигурные скобки.

Пример 2.

В данном примере весь документ формируется динамически.

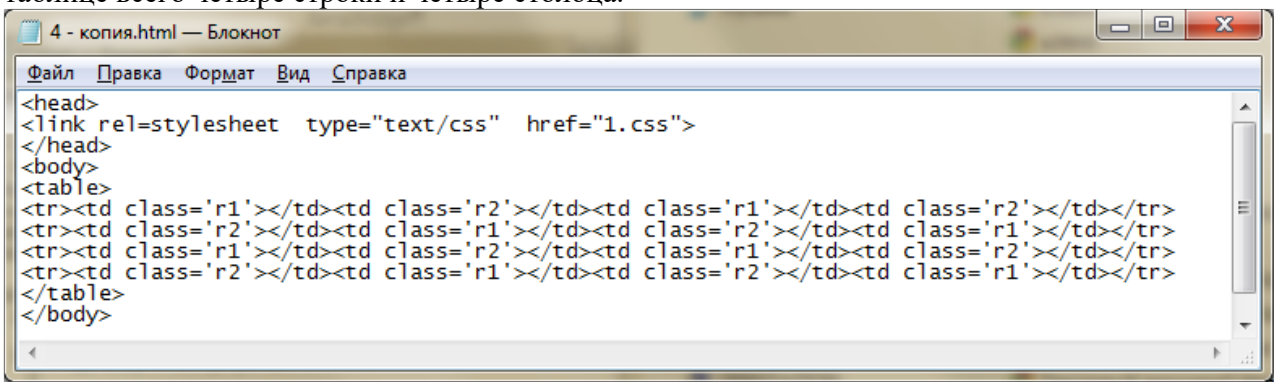




Пример 3.

Создадим таблицу 10 на 10. Высота и ширина ячейки 50 пикселей. Закрасим таблицу в шахматном порядке. Таблицу будем формировать динамически, с помощью метода write. Для закраски таблицы используем вложенные операторы цикла.

Фактически, нужно сформировать документ аналогичный данному. Правда, в этой таблице всего четыре строки и четыре столбца.



1.css — Блокнот

Файл Правка Формат Вид Справка

```
table { border-collapse:collapse;}
td    {width: 50px; height: 50px}
.r1   {background-color:red}
.r2   {background-color:blue;}
```

4.html — Блокнот

Файл Правка Формат Вид Справка

```
<head>
<link rel=stylesheet  type="text/css"  href="1.css">
</head>
<body>
<script>
var n=10;var s;
document.write("<table>");

for (i=1;i<=n;i=i+1)
{ document.write("<tr>")

  for (j=1;j<=n; j=j+1)
   {
    st=(i+j)%2
    if (st==0) s="class='r1'";
    else s="class='r2'";
    document.write("<td "+s+"> </td>");
   }

  document.write("</tr>");
}

document.write("</table>");
</script>
</body>
```

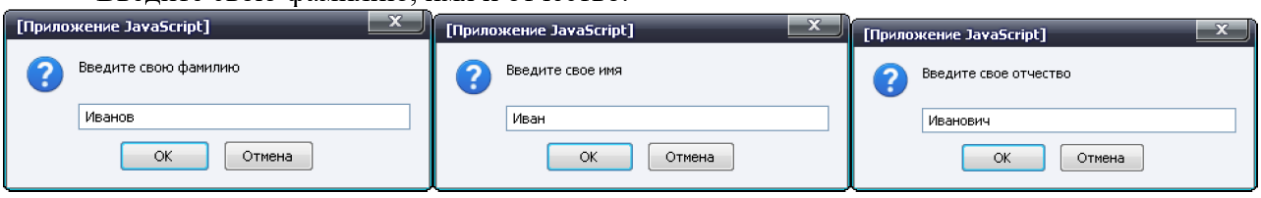
До цикла и после цикла выводятся теги <table> и </table>. Во внешнем цикле в документ выводятся теги <tr> и </tr>, формирующие строку таблицы. Во внутреннем цикле в документ выводятся теги <td> и </td>, формирующие ячейки таблицы. От четности st зависит значение текстовой переменной s, которая определяет класс ячейки. Ячейка класса r1 имеет красную заливку, ячейка класса r2 заливается синим цветом.

Задания на практическую работу

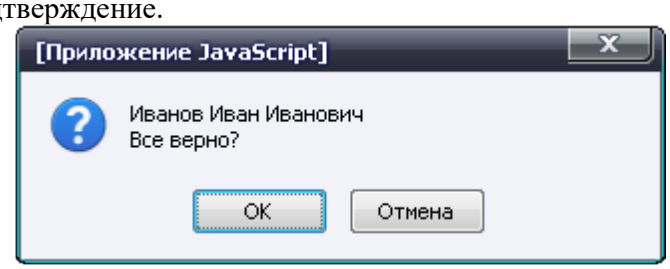
Вариант 1

Задание 1.

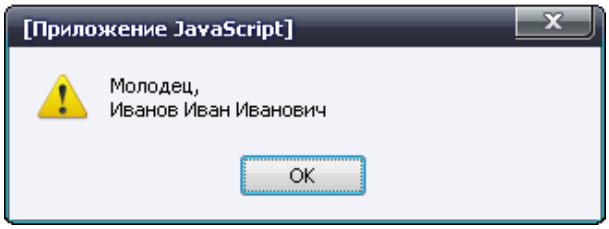
Введите свою фамилию, имя и отчество.

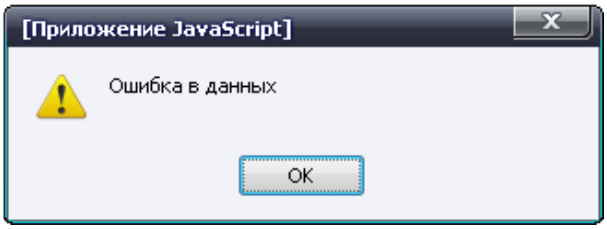


Запросите подтверждение.



Если все верно, то вывести приветствие, если нет, вывести сообщение об ошибке.


[Приложение JavaScript]
Молодец,
Иванов Иван Иванович
OK

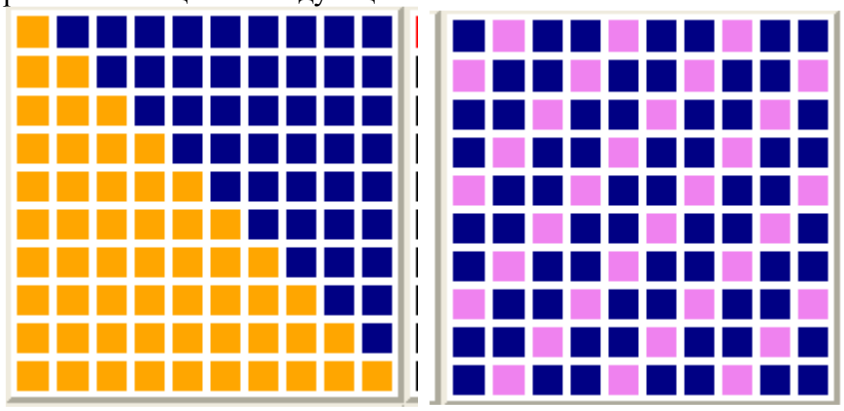

[Приложение JavaScript]
Ошибка в данных
OK

Задание 2

Введите время в часах и минутах. Определите время, которое будет через минуту. Используйте вложенные операторы IF.

Задание 3

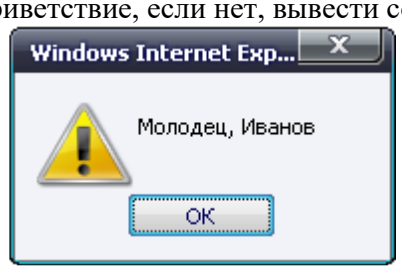
Сформировать таблицы по следующим шаблонам.

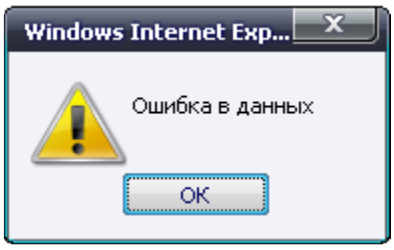


Вариант 2

Задание 1

Введите свою фамилию, пол и возраст. Запросите подтверждение. Если данные верны, то вывести приветствие, если нет, вывести сообщение об ошибке.



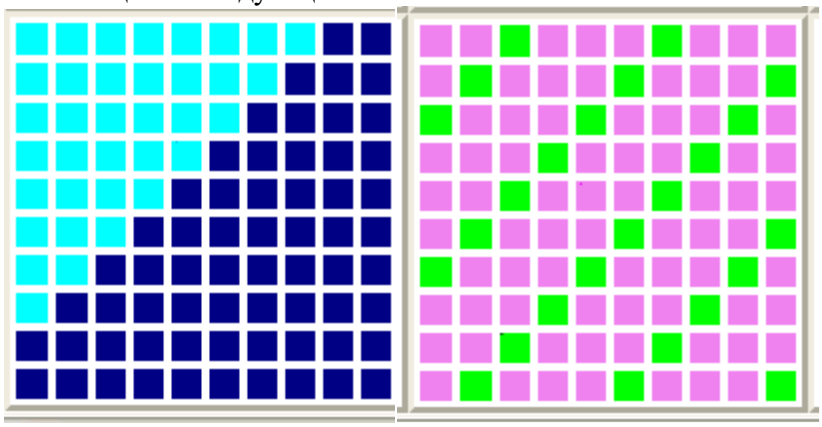


Задание 2

Дано два числа x, y и знак арифметической операции (+, -, *, /). Найти x+y, x-y, x*y, x/y, в зависимости от введенного знака. В случае ошибки в знаке или деления на 0 вывести сообщение об ошибке.

Задание 3

Сформировать таблицы по следующим шаблонам.



4. Практическая работа №2 «События и функции»

Цель работы: научиться обрабатывать события с помощью Javascript.

События JavaScript

Практически все JavaScript-приложения выполняют те или иные действия, откликаясь на различные события.

Событие - это сигнал от браузера о том, что что-то произошло.

События делятся на несколько категорий:

1. события, связанные с документом;
2. события, связанные с элементами документа;
3. события, связанные с окнами.

Для того чтобы скрипт реагировал на событие - нужно назначить обработчик события.

Обычно обработчики называют "on+имя события", например: onclick.

Назначение обработчиков событий для элементов

Существует несколько способов назначать обработчик на конкретное событие элемента. Один из этих способов – обработчик события записывается прямо в открывающем теге элемента.

Например, для обработки события click на кнопке input, можно назначить обработчик onclick вот так:

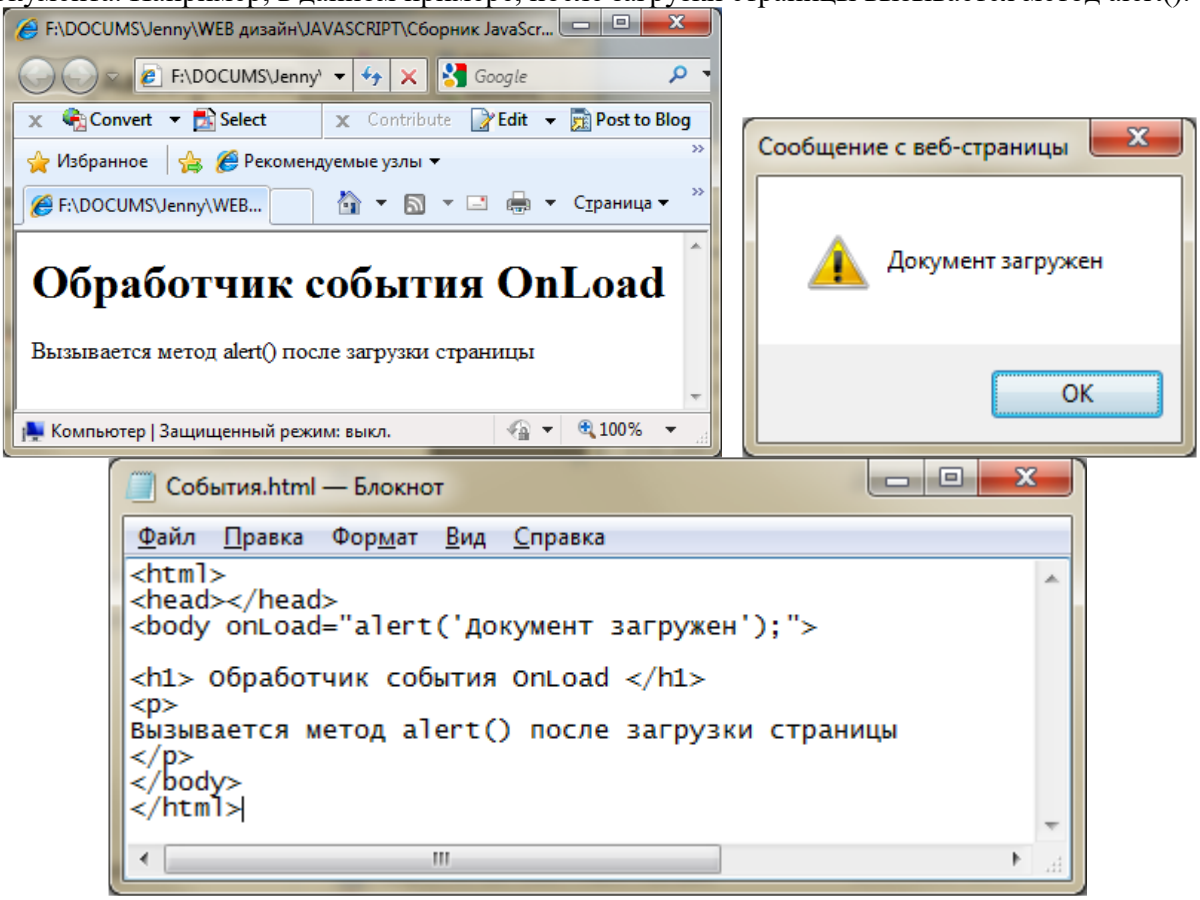
<input type="button" value="Нажми Меня" onclick="alert('Спасибо!');" />

В этом случае JavaScript код пишется в кавычках в одну строку.

Такой способ установки обработчиков очень удобен - он нагляден и прост, поэтому часто используется в решении простых задач.

Событие Load и его обработчик onLoad

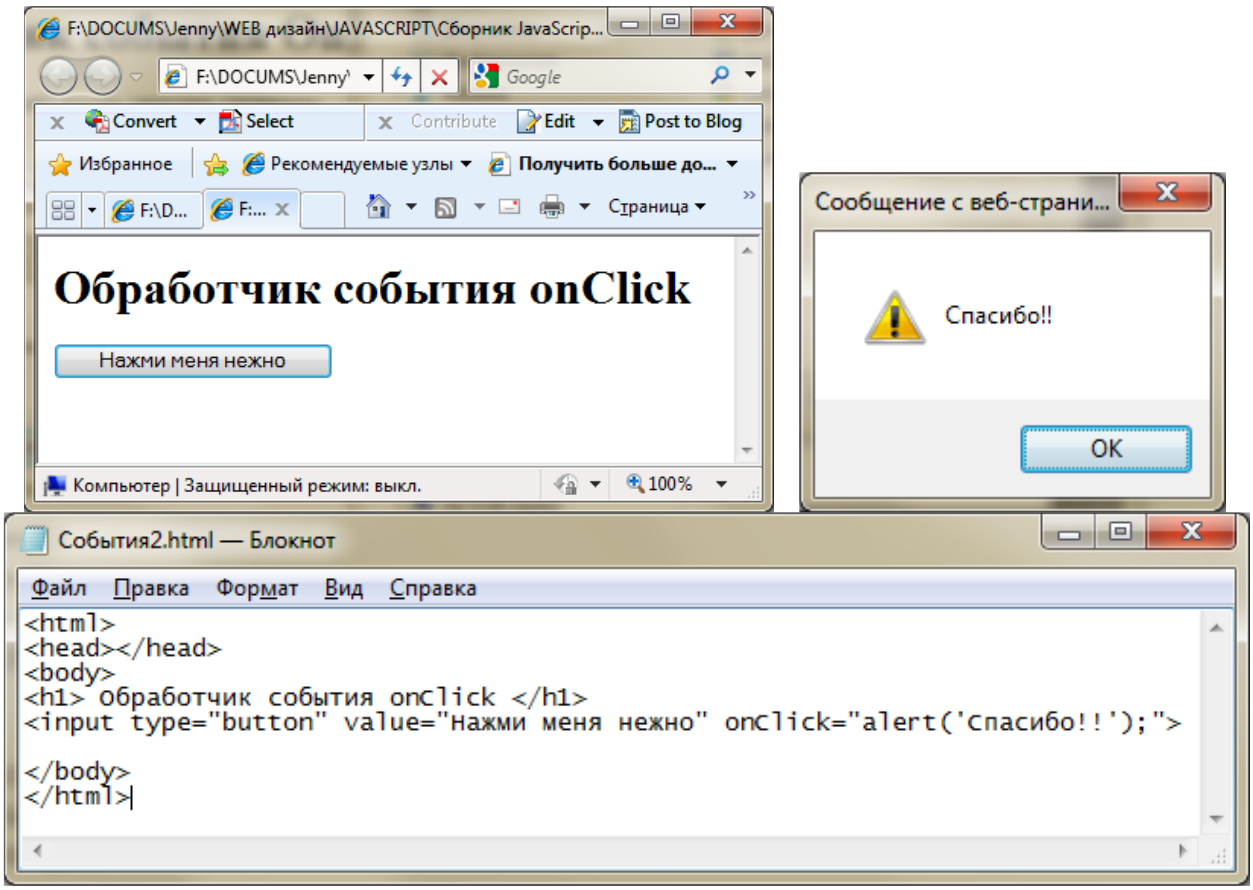
Событие Load возникает для элементов body и frameset когда закончена загрузка документа. Например, в данном примере, после загрузки страницы вызывается метод alert().



Событие Click и его обработчик onClick

Событие Click – одинарный щелчок (нажата и отпущена кнопка мыши) возникает фактически для всех элементов страницы.

Например, после нажатия на кнопку вызывается метод alert().



Назначение обработчиков событий

У этого способа установки обработчика событий есть и минусы.

Как только обработчик начинает занимать больше одной строки - читабельность резко падает.

В этом случае для обработки события нужно использовать функцию. При этом в обработчике события указывают только имя функции, а сама функция описывается в разделе <head>. Описание функции

Синтаксис:

function Имя_Функции (необязательный список формальных аргументов через запятые)

{

...

операторы

...

return значение;

}

Команда return, возвращающая значение функции, может быть не одна, может и вовсе отсутствовать. В последнем случае функция не возвращает никакого значения и ее вызов нельзя использовать в выражениях. Если в функцию или из нее не передаются параметры - то после имени функции ставятся круглые скобки без параметров. Вызов функции

Синтаксис:

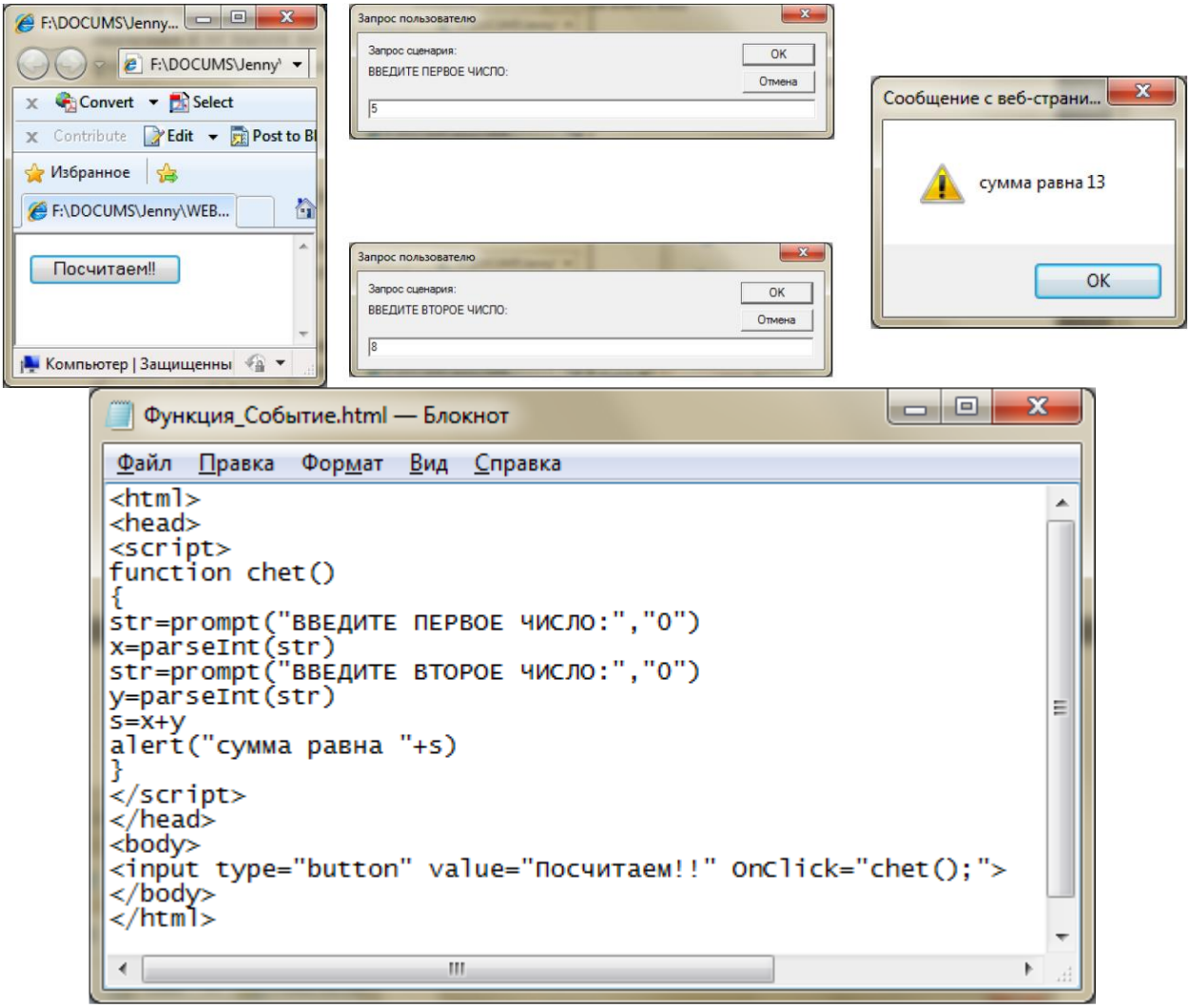
Имя_Функции(список фактических аргументов через запятые)

Фактическим аргументом функции может быть константа, переменная, выражение и, в частности, вызов другой функции.

Функция не может быть выполнена до тех пор, пока не будет явного обращения к ней.

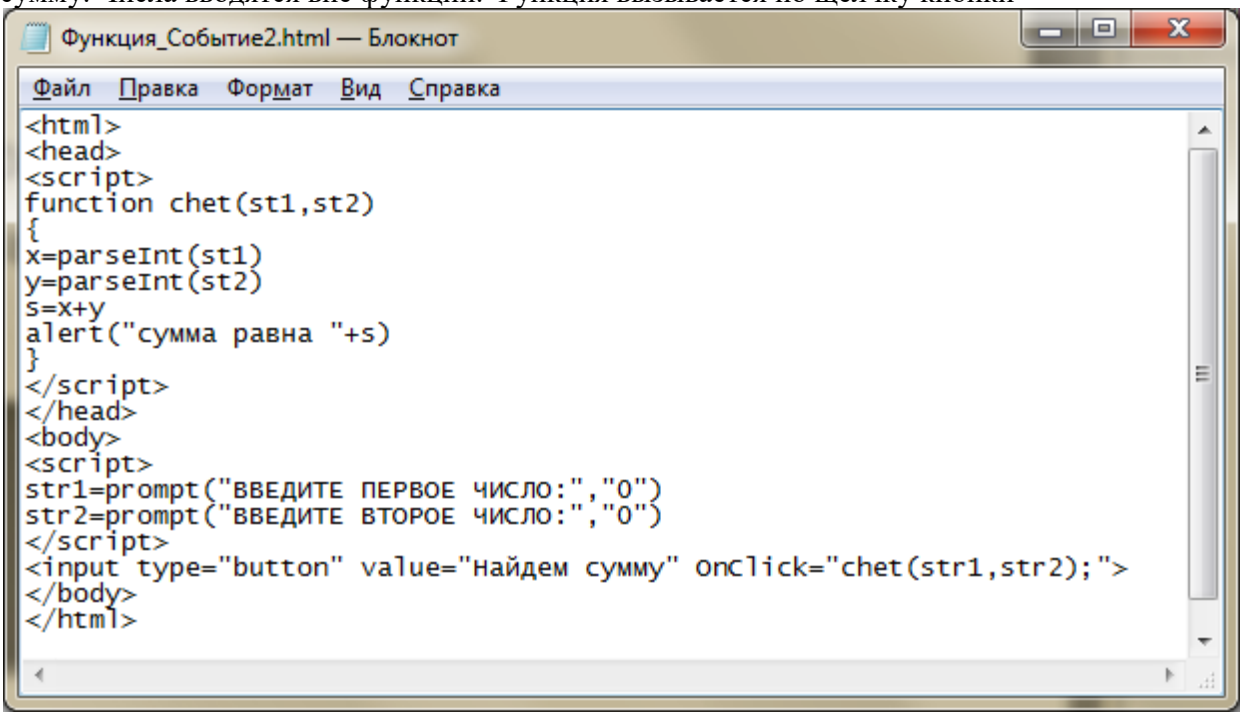
Пример 1

Оформим ввод и вычисление суммы двух чисел как функцию, которая вызывается по щелчку кнопки.



Пример 2

Оформим функцию, которая в качестве параметра получает два числа и находит их сумму. Числа вводятся вне функции. Функция вызывается по щелчку кнопки



Данная функция получает два параметра в строковом виде, преобразует их в числовой формат, складывает и выводит результат.

Задания на практическую работу

Задание

Оформить задание 2 из Практической работы 1 в виде функции и вызвать эту функцию по нажатию кнопки.

5. Практическая работа №3 «Встроенные объекты»

Цель работы: изучить встроенные объекты Javascript.

Основной единицей в языке JavaScript является объект, который объединяет в себе данные (свойства) и средства обработки этих данных (методы).

Все объекты, которые используются в языке JavaScript, делятся на такие большие группы:

- Встроенные объекты базового языка;
- Объекты документа. Объект Math.

В языке JavaScript существует специальный объект «Math», в котором собраны основные математические функции и константы. Все свойства и методы объекта являются статическими, поэтому сам объект создавать не нужно. В таблицах ниже приведён список некоторых из его методов (таблица 5.1).

Таблица 5.1 – Методы класса Math

Метод	**Описание**
abs(число)	Возвращает модуль (абсолютную величину) числа.
sqrt(число)	Возвращает квадратный корень из числа.
pow(основание, степень)	Возвращает результат возведения основания в указанную степень. Например, Math.pow(5, 3) вернёт 53=125.
random()	Возвращает псевдослучайное число в диапазоне от 0 до 1.
ceil(число)	Производит округление в большую сторону, то есть возвращает наименьшее целое число, большее либо равное аргументу.
floor(число)	Производит округление в меньшую сторону, то есть возвращает наибольшее целое число, меньшее либо равное аргументу.
round(число)	Округляет указанный аргумент до целочисленного значения.
cos(число)	Возвращает косинус числа.
sin(число)	Возвращает синус числа.
exp(число)	Возводит число e (основание натурального логарифма) в указанную степень.
log(число)	Возвращает натуральный логарифм числа.

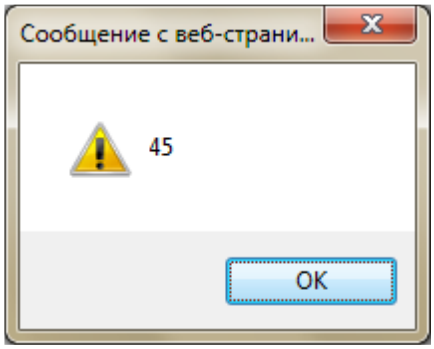
Чтобы задействовать метод (обычно это называют вызовом метода), в операторе JavaScript нужно сделать на него ссылку. Для этого используется синтаксис с использованием точки.

Очень часто используется также свойство этого объекта, которое имеет значение числа пи, Math.PI

Пример

В данном скрипте к объекту Math применили метод floor. В качестве параметра метода использовали значение переменной d1. Для этого использовали конструкцию Math.floor(d1). Этот метод отбрасывает дробную часть у аргумента. Это значение присваивается переменной d2 и выводится в окно с помощью метода alert(d2).

```
<script>
d1=45.6;
d2=Math.floor(d1);
alert(d2);
</script>
```



Объект Date

Объект Date предназначен для манипуляций с датами и временем. Его примитивным значением является число, равное количеству миллисекунд относительно базового времени, равного полуночи 1 января 1970 г.

День состоит из 86400000 миллисекунд.

Объект и экземпляр объекта

Объект -- это шаблон. Экземпляр объекта -- это рабочая копия.

Например, объектом является комплект документации на заводе, по которой изготовляются телевизоры. Сам телевизор является экземпляром этого объекта. Все телевизоры, которые сходят с конвейера, имеют одни и те же свойства изображения и одни и те же методы управления этими свойствами.

Создание экземпляра объекта Date

Для создания экземпляра объекта используется ключевое слово new.

Создание экземпляра объекта с текущей датой и временем: new Date()

var a = new Date();

В переменной «a» текущая дата и время.

Создание экземпляра объекта с датой и временем, заданными аргументами конструктора:

new Date(год,месяц, день [,часы [,минуты [,секунды [,мс]?]?]?]?)

Здесь:

год — числовое выражение, задающее полный номер года (например, 1988, а не 88);

месяц — числовое выражение, задающее номер месяца (0 = январь, 1 = февраль, …, 11 = декабрь);

день — числовое выражение, задающее номер дня в месяце от 1 до 31;

часы — необязательное числовое выражение, задающее номер часа от 0 до 23;

минуты — необязательное числовое выражение, задающее номер минуты от 0 до 59;

секунды — необязательное числовое выражение, задающее номер секунды от 0 до 59;

мс — необязательное числовое выражение, задающее номер миллисекунды от 0 до 999.

Например, var c = new Date(1958, 4, 21, 10, 15);

В переменной «c» дата – 21 мая 1958 года и время 10 часов 15 мин. Нумерация месяцев начинается с 0.

Методы объекта Date

Метод это действие которое выполняется для объекта или с объектом. По своей сути это команда, но ее действия связаны с определенным объектом. У каждого объекта может быть много методов (таблица 5.2).

Таблица 5.2 – Методы класса Date

Метод	**Описание**
getTime()	Возвращает примитивное значение объекта.
getYear()	Возвращает номер года по местному времени
getMonth()	Возвращает месяц по местному времени.
getDate()	Возвращает день месяца по местному времени.
getDay()	Возвращает день недели по местному времени.
getHours()	Возвращает часы по местному времени.
getMinutes()	Возвращает минуты по местному времени.
getSeconds()	Возвращает секунды по местному времени.

Пример 1

Выведем в окно номер текущего года.

```
<body>
<script>
var today= new Date();
var god=today.getYear();
alert(god);
</script>
</body>
```

В данном скрипте создается экземпляр объекта Date. Он имеет имя today и содержит текущую дату. Для применения метода getYear() к объекту today используется конструкция today.getYear(). Она возвращает значение года из этой даты. Это значение присваивается переменной god и выводится в окно с помощью метода alert(god).

Пример 2

Выведем в окно номер дня недели для даты 21 мая 1958 года.

```
<body>
<script>
var dr = new Date(1958, 4, 21);
dn=dr.getDay()
alert(dn)
</script>
</body>
```

В данном скрипте создается экземпляр объекта Date. Он имеет имя dr и содержит дату 21 мая 1958 года. Для применения метода getDay() к объекту dr используется конструкция dr.getDay(). Она возвращает значение дня недели для этой даты. Это значение присваивается переменной dn и выводится в окно с помощью метода alert(dn). Выводится число 3, то есть среда.

Объект Array

Объект Array предназначен для хранения массивов данных. Массив -- это упорядоченный набор элементов. Доступ к отдельному элементу производится по имени и индексу (номеру). Нумерация элементов в JavaScript начинается с нуля.

Пример 3

Выведем в окно названия дня недели для даты 21 мая 1958 года. Для хранения названий дней недели создадим dayn – экземпляр объекта Array. В качестве индекса для выбора из массива названия дня недели используется вычисленное значение переменной dn.

```
<body>
<script>
var dr = new Date(1958, 4, 21);
dn=dr.getDay()
var dayn=new Array('воскресенье',
'понедельник','вторник','среда',
'четерг','пятница','суббота')
```

```
alert(dayn[dn])
</script>
</body>
```

Ссылка на объект документа. Метод getElementById

Для размещения значений, возвращаемых описанными методами на Web–странице, необходимо уметь обратиться к элементам этой страницы (документа).

Наилучший способ заключается в том, чтобы обратится к объекту по имени. Поэтому нужно каждому элементу (объекту) присвоить имя. Для этого используется атрибут id (или name).

Примеры:

```
<p id=”first”>
<img id=’logo1’ name=”logo1” src=’1.gif’>
```

Рекомендуется назначать одно и то же значение атрибутам id и name.

Для обращения к элементу, который имеет имя, в JavaScript используется следующая конструкция:

```
document.getElementById(“Имя_объекта”)
```

Например, если мы хотим обратиться к элементу с именем ”first”, то будем использовать следующую ссылку:

```
document.getElementById(“first”)
```

Обратите внимание, что все ключевые слова чувствительны к регистру. В названии метода используются три буквы в верхнем регистре.

Метод getElementById() относится к объекту document. С его помощью мы можем найти любой поименованный элемент документа. Для разделения элементов иерархической ссылки используется точка.

Обращение к свойствам объекта

Каждый объект обладает некими характеристиками, которые называются свойствами.

Например: пусть объект текстовое поле описывается так:

```
<input type=’text’ id=’entry’ name=’entry’ value=’User Name?’>
```

Для получения доступа к свойству объекта используется тот же тип синтаксиса с точками. Ссылка на данное свойство состоит из ссылки на данный объект плюс еще одно расширение, указывающее на нужное свойство. Например, чтобы обратиться к свойству value элемента текстовое поле, который имеет имя ’entry’ нужно записать выражение вида:

```
document.getElementById(“entry”).value
```

Или можно сопоставить наш объект с переменной ob и присвоить свойству value этого объекта новое значение ’Введите имя ’.

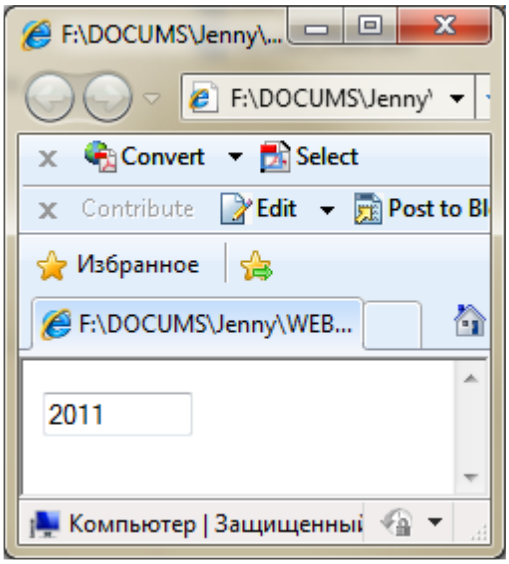
```
var ob=document.getElementById(“entry”)
ob.value=’Введите имя ’;
```

Пример 4

Выведем год извлеченный из текущей даты с помощью метода не в окно, как в примере 1, а в текстовое поле формы. Это текстовой поле имеет имя id='f1'. Наш скрипт оформлен в виде функции gd(), которая вызывается при загрузке документа. В этой функции с помощью метода getElementById определена переменная p1, которая есть наш объект текстовое поле. У этого объекта есть свойство value, которому мы присваиваем значение god.

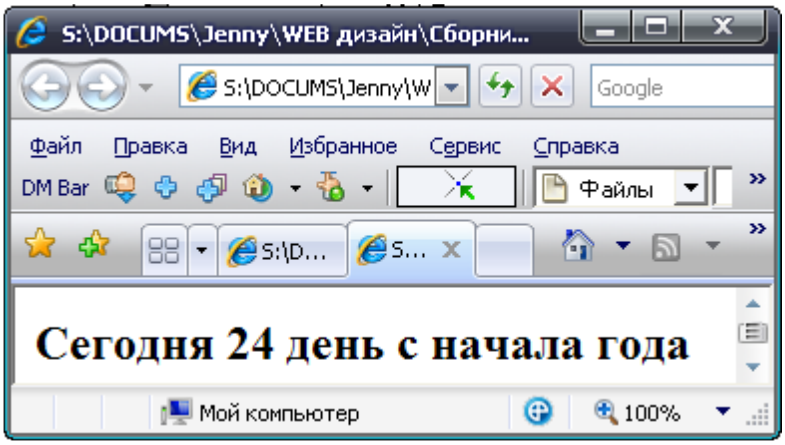
```
<html>
<head>
<script>
function gd()
{
var today= new Date();
var god=today.getYear();
p1=document.getElementById('f1');
p1.value=god;
}
</script>
```

```
</head>
<body onLoad="gd()">
<form>
<input type=’text’ id='f1' size=8>
</form>
</body>
</html>
```



Пример 5

Создать HTML-страницу, которая при загрузке сообщает, сколько дней прошло от начала года.



Для решения задачи нужно определить:

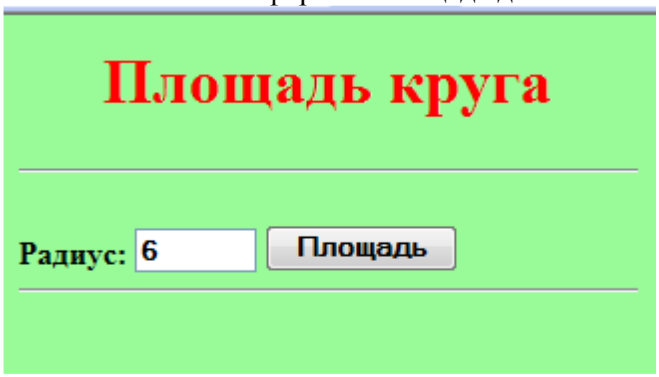
- количество миллисекунд от 1 января 1970 г до текущей даты
- количество миллисекунд от 1 января 1970 г до 1 января текущего года.,
- из первого вычесть второе и разделить на 86400000 (количество миллисекунд в сутках).

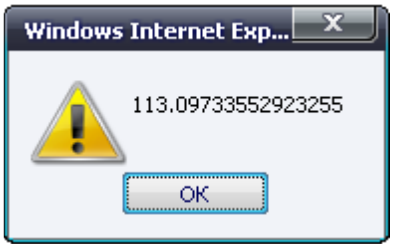
```
<html>
<head>
</head>
<body>
<script>
// переменная today имеет значение текущее время и дата
var today = new Date();
// переменная y- текущий год
var y = today.getYear();
// переменная yearBegin имеет значение начало текущего года
var yearBegin = new Date(y,0,1);
```

```
// переменная d - количество миллисекунд от начала года.
var d = today.getTime() - yearBegin.getTime();
// Переведем d в сутки
d =d/ (86400000);
// прибавим 1, иначе 1 января будет нулевым днем
d=d+1;
// отбросим дробную часть
d=Math.floor(d)
// печать результата
document.write("<h2>Сегодня "+d+" день с начала года</h2>");
</script>
</body>
</html>
```

Пример 6

Напишите приложение для вычисления площади круга. Радиус для вычисления вводить в текстовое поле формы. Площадь должна вычисляться при нажатии кнопки.






```
<html>
<head>
<style type="text/css">
h1 {color:red;text-align:center;}
body {background-color:PaleGreen; font-weight:bold;}
input {font-weight:bold;}
</style>
<script>
function sqr()
{
// переменная p объект с именем t
var p=document.getElementById('t');
// переменная r содержит значение свойства value этого объекта
r=p.value;
var s= Math.PI*r*r;
alert(s)
}
</script>
</head>
<body>
<h1>Площадь круга</h1>
<hr>
<form>
Радиус:
<input type=text value="1" id='t' name="t" size="5">
<input type=button value="Площадь" OnClick="sqr()">
<hr>
</form>
```

</body>
</html>

По нажатию кнопки "Площадь" начинает работать функция sqr(). Она обращается к элементу с именем t и присваивает значение свойства value этого элемента переменной r. Используя это значение в качестве радиуса, и обращаясь к свойству PI объекта Math, вычисляем площадь круга и выводим ее в окно с помощью метода alert.

Задания на практическую работу

Вариант 1

1. Создать HTML-страницу, которая при загрузке выводит на страницу текущий день недели, число, месяц и год. Для месяцев и дней недели организовать массивы.

2. Создать HTML-страницу, которая при загрузке запрашивает дату вашего рождения и выводит количество дней, которые Вы прожили в текстовое поле формы.

3. Ввести координаты точки в поля формы и определить расстояние от этой точки до начала координат. Расстояние должно вычисляться по нажатию кнопки.

Вариант 2

1. Создать HTML-страницу, которая при загрузке запрашивает дату вашего рождения и выводит на страницу день недели, число, месяц и год этой даты. Для месяцев и дней недели организовать массивы.

2. Создать HTML-страницу, которая при загрузке выводит в текстовое поле формы, сколько дней осталось до каникул.

3. Ввести длину и ширину прямоугольника в поля формы и определить площадь этого прямоугольника. Площадь должна вычисляться по нажатию кнопки.

6. Практическая работа №4 «Объект Window»

Цель работы: научиться пользоваться методами Javascript для изменения контента страницы браузера.

Объект window находится в вершине иерархии объектов и содержит в себе все другие объекты браузера. Объект window описывает текущее окно браузера и его содержимое.

Методы объекта Window

Основные методы объекта Window представлены в таблице 6.1.

Таблица 6.1 - Основные методы объекта Window

метод	**описание**
`open`	Открывает новое окно браузера.
`close`	Закрывает окно браузера.
`alert, prompt, confirm`	Стандартные диалоговые панели.
`setInterval`	Указывает функции выполняться периодически через заданное количество миллисекунд.
`setTimeout`	Запускает функцию через заданное количество миллисекунд.
`clearInterval`	Отменяет действие метода setInterval.
`clearTimeout`	Отменяет действие метода setTimeout.

Открыть новое окно и закрыть его

var переменная = open();

var переменная = open(файл);

var переменная = open(файл, имя_окна);

var переменная = open(файл, имя_окна, параметры_окна);

переменная.close();

где

переменная -- экземпляр создаваемого окна.

файл -- строка, имя файла, который нужно отобразить в созданном окне. Если параметр не задан, создается пустое окно. Тогда формирование документа в этом окне

нужно осуществлять динамически, с помощью метода document.write.

имя_окна -- строка, имя окна; используется для обращения к созданному окну (значение свойства "id" объекта window).

параметры_окна -- строка, свойства окна; когда параметр не задан, используются свойства окна по умолчанию. Параметры окна задаются в строке параметры_окна в виде:

"параметр1=значение1,параметр2=значение2,...,параметрN=значениеN"

Замечание. Не рекомендуется оставлять пробел после разделительной запятой.

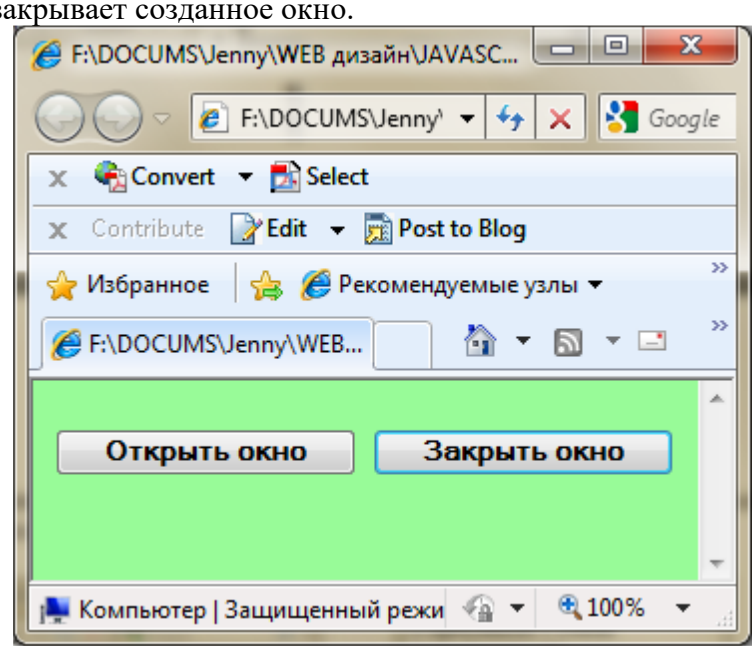
Описание некоторых параметров окна приведено в таблице 6.2.

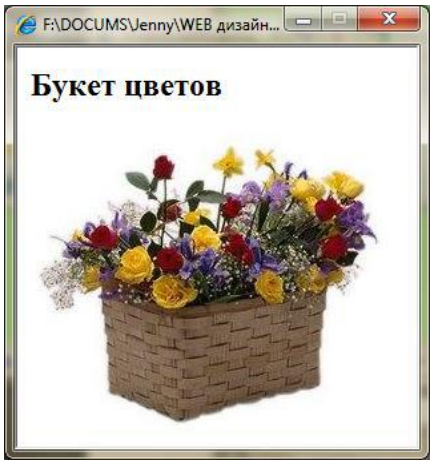
Таблица 6.2 - Описание некоторых параметров окна

параметр	**значение**	**описание**
width	число	Ширина окна в пикселах. Минимальное значение -- 100.
height	число	Высота окна в пикселах. Минимальное значение -- 100.
left	число	Задает горизонтальную координату левого верхнего угла окна в пикселах.
top	число	Задает вертикальную координату левого верхнего угла окна в пикселах.

Пример 1

Напишите приложение, которое по нажатию кнопки создает новое окно, в которое выводится существующий html-документ (11.html) и по нажатию другой кнопки, закрывает созданное окно.






```
<html>
<head>
<style type="text/css">
body {background-color:PaleGreen;}
input {font-weight:bold;}
</style>
<script>
function op()
{
// переменная win экземпляр объекта window
// создается окно размерами 300 на 300,
// в окно открывается файл 11.html
// у нового окна id=”www”
win=window.open("11.html","www","width=300,height=300");
```

```
}
</script>
</head>
<body>
<form>
// по этой кнопке вызывается функция op()
<input type=button value="Открыть окно" OnClick="op()">
// по этой кнопке метод close() применяется к объекту win
<input type=button value="Закрыть окно" OnClick="win.close()">
</form>
</body>
</html>
```

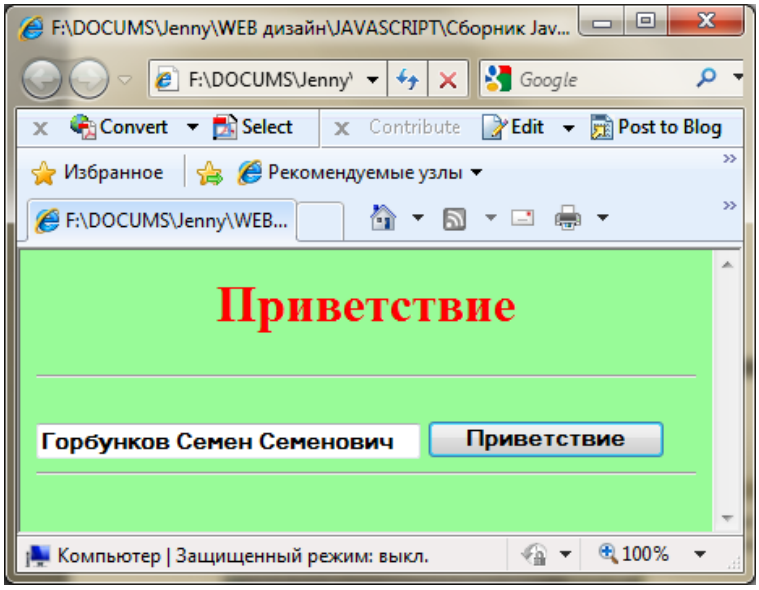
Методы open( ) и close() объекта document

Если документ в дочернем окне формируется динамически, с помощью метода document.write(), нужно обязательно перед использованием этого метода открыть запись в новое окно методом document.open(), а после завершения вывода в новое окно закрыть запись методом document.close(). При выводе в новое окно обязательно нужно указывать имя объекта. Например,

```
win.document.write('<h2>Примите в подарок этот букет!</h2>');
```

Пример 2

Напишите приложение, которое вводит имя человека в поле формы и по нажатию кнопки создает новое окно с приветствием этому человеку. Документ во втором окне формировать динамически, с помощью метода document.write(). Кнопка для закрытия нового документа должна быть в нем самом.



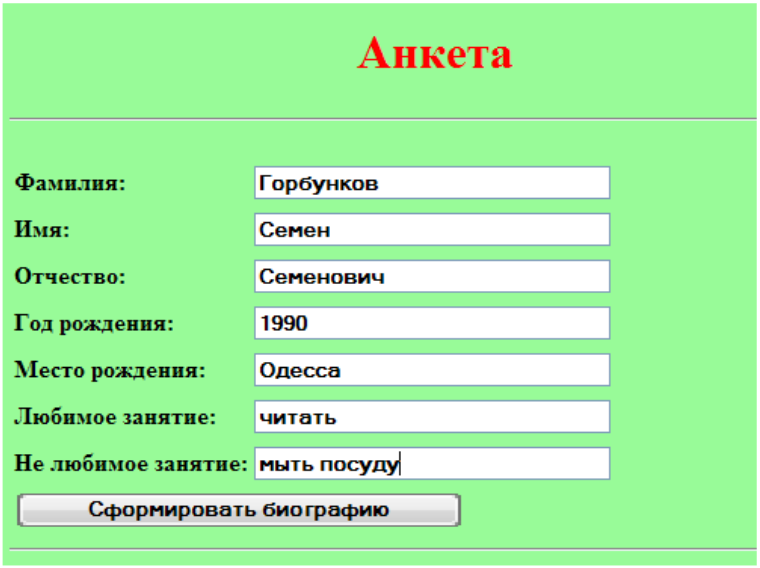

```
<html>
<head>
<style>
h1 {color:red;text-align:center;}
body {background-color:PaleGreen; font-weight:bold;}
input {font-weight:bold;}
</style>
<script>
function hello()
{
// переменная p объект текстовое поле с именем t
var p=document.getElementById('t');
```

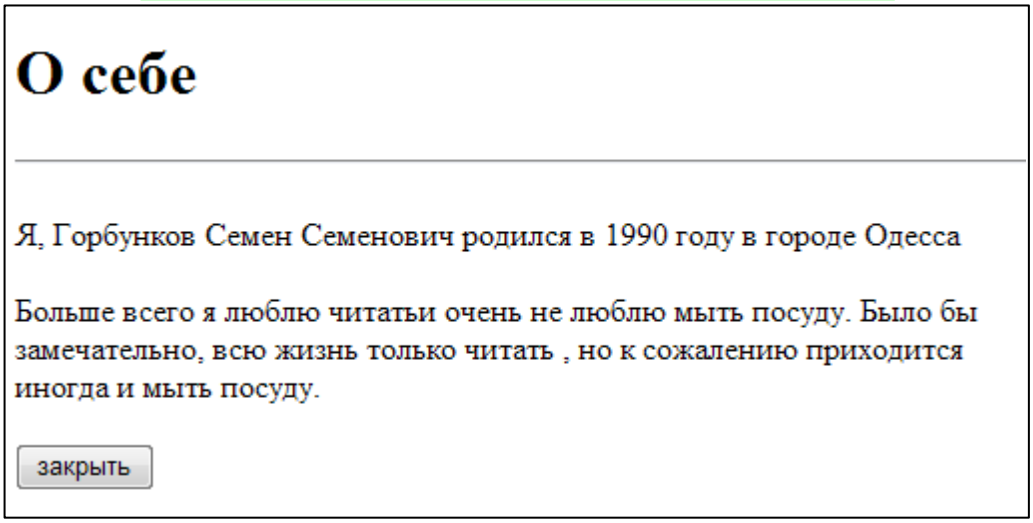
```
// переменная r содержит значение свойства value этого объекта
r=p.value;
// переменная win экземпляр объекта window
// создается пустое окно размерами 400 на 500,
var win=window.open("","","width=400,height=500");
// открываем запись в это окно - объект win
win.document.open();
// формирование строки str
// содержимое в заголовок передается из поля формы, переменная r
var str = "<h1>"+r+"!!!</h1><hr><P>";
// вывод строки в документ окна win
win.document.write(str);
// формирование и вывод рисунка
str = '<IMG src=”flower.jpg”>';
win.document.write(str);
str ='<h2>Примите в подарок этот букет!</h2>';
win.document.write(str);
// формирование и вывод кнопки закрытия
str ='<input type="button" value="закрыть"' +'onClick="window.close();">';
win.document.write(str);
// закрываем вывод в документ объекта win
win.document.close();
}
</script>
</head>
<body>
<h1>Приветствие</h1>
<hr>
<form>
<input type=text value="Горбунков Семен Семенович" name="t" id='t' size="30">
<input type=button value="Приветствие" OnClick="hello()">
<hr>
</form>
</body>
</html>
```

Задания на практическую работу

Вариант 1

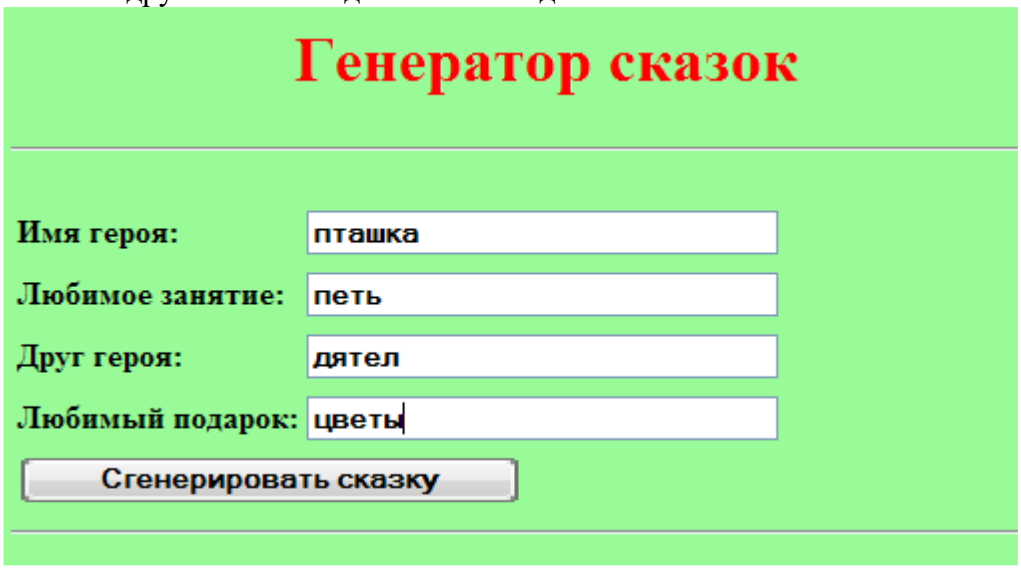
Ввести данные в анкету, состоящую из текстовых полей формы и сформировать биографию по этим данным на отдельной странице.

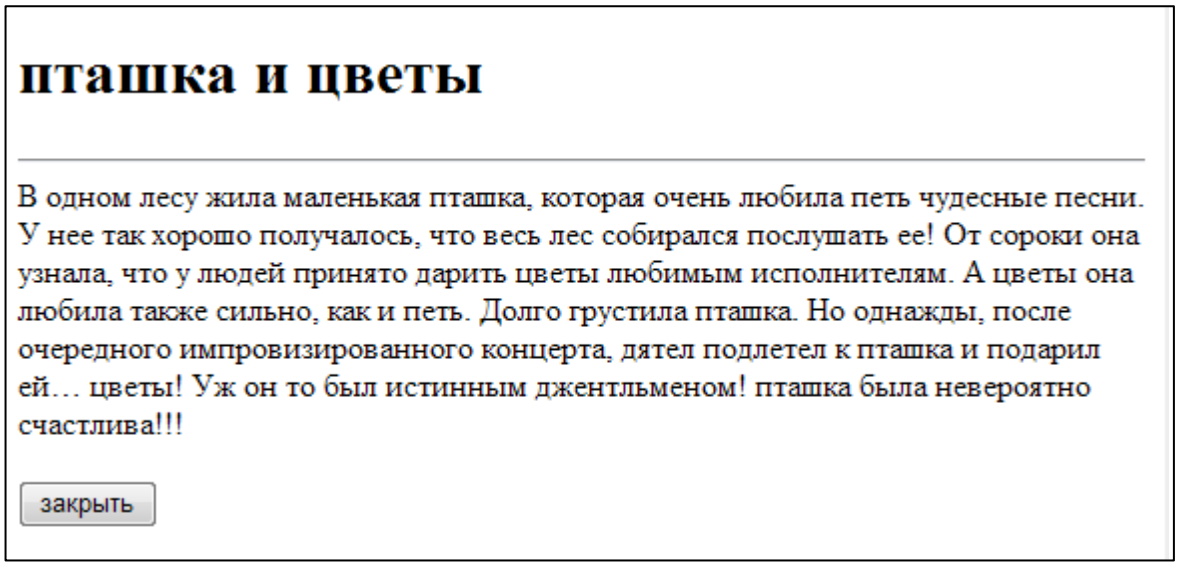




Вариант 2

Создать генератор сказок. Ввод данных в текстовые поля формы. Сгенерированную сказку вывести в другое окно. Падежи в сказке должны соотвествовать.





7. Практическая работа №5 «Обращение к элементам формы – флажки, радиокнопки, списки»

Цель работы: научиться пользоваться элементами формы для изменения контента страницы браузера.

В предыдущих работах рассматривались объекты формы – текстовые поля и кнопки, определяли значение свойства value текстового поля. В этой работе мы должны научиться проверять значения свойств других объектов формы. А именно: флажков, радиокнопок, выпадающих списков. Свойство checked объекта input

Флажки и радиокнопки определяются в документе тегом <input>. С атрибутами:

- type="checkbox" – флажок;
- type="radio" – радиокнопка.

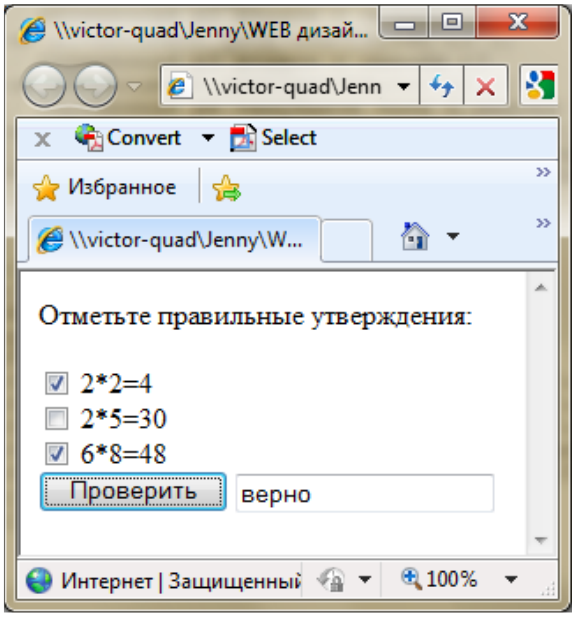
Флажки не зависят один от другого. Их можно устанавливать и сбрасывать в любой комбинации. Радиокнопки предназначены для выбора одного варианта из нескольких альтернативных.

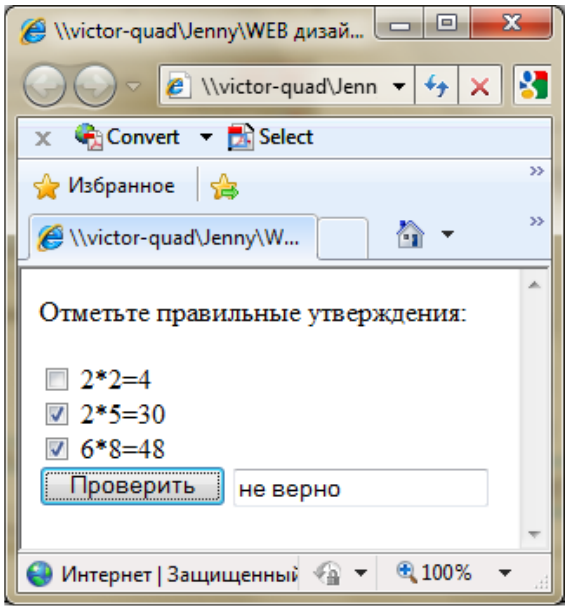
Флажки и радиокнопки разновидности объекта input. У объекта input есть свойство checked. Это свойство имеет значение true, когда флажок установлен (радиокнопка выбрана), и false --в противном случае. Мы должны научиться проверять значение этого свойства.

Обратите внимание, что свойство checked объекта input имеет совсем другой смысл, чем одноименный атрибут тега input. Атрибут указывает, что флажок установлен по умолчанию, а свойство хранит текущее положение флажка, которое может изменяться пользователем или программой.

Пример 1

Напишите приложение для проверки таблицы умножения. Правильные примеры необходимо отметить флажками. Если ошибок нет, в поле формы вывести слово «верно», в противном случае «неверно».

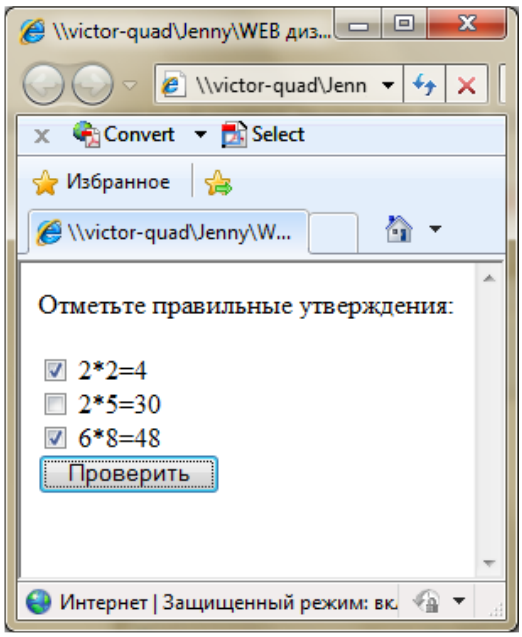


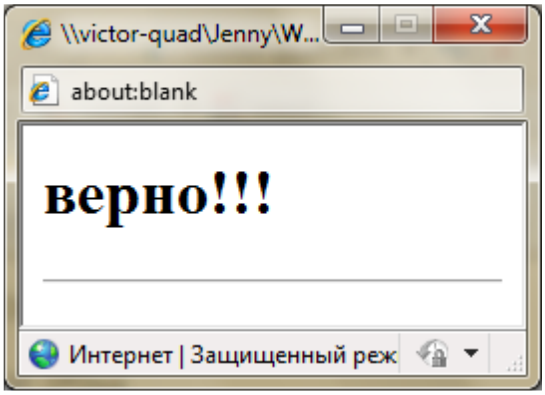



```
<head>
<script>
function checkAll()
{
var ans='не верно';
// переменная p1 флажок с именем c1
var p1=document.getElementById('c1');
// переменная p2 флажок с именем c2
var p2=document.getElementById('c2');
// переменная p3 флажок с именем c3
var p3=document.getElementById('c3');
// переменная p4 текстовое поле с именем result
var p4=document.getElementById('result');
// если флажок c1 поднят, флажок c2 не поднят
// и флажок c3 поднят
if (p1.checked && !p2.checked && p3.checked) ans='верно';
// вывод результата в текстовое поле
p4.value=ans;
}
</script>
</head>
<body>
<p> Отметьте правильные утверждения:</p>
<form>
<input type="checkbox" id="c1" name="c1"> 2*2=4 <br/>
<input type="checkbox" id="c2" name="c2"> 2*5=30<br/>
<input type="checkbox" id="c3" name="c3"> 6*8=48<br/>
<input type="button" value="Проверить" onClick="checkAll();">
<input type="text"id="result" name="result" value="">
</form>
</body>
```

Пример 2

То же самое приложение переделать так, чтобы результат выводился в новое окно.



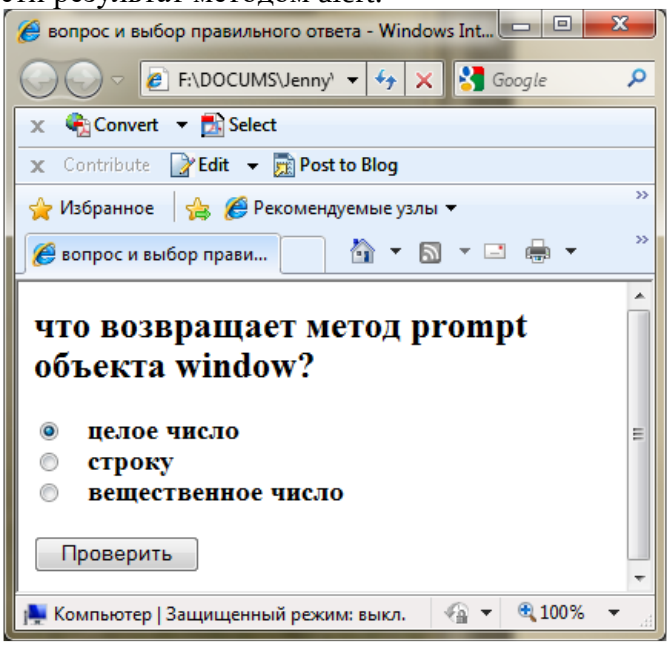


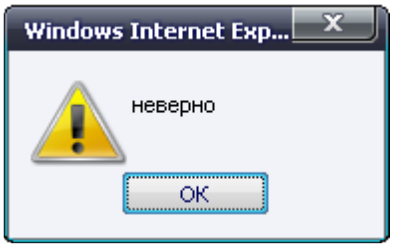

```
<head>
<script>
function checkAll()
{
var ans='не верно';
// переменная p1 флажок с именем c1
var p1=document.getElementById('c1');
// переменная p2 флажок с именем c2
var p2=document.getElementById('c2');
// переменная p3 флажок с именем c3
var p3=document.getElementById('c3');
// если флажок c1 поднят, флажок c2 не поднят и флажлк c3 поднят
if (p1.checked && !p2.checked && p3.checked) ans='верно';
// вывод результата в новое окно
// переменная win экземпляр объекта window
// создается пустое окно размерами 100 на 100,
var win=window.open("","","width=100,height=100");
// открываем запись в документ окна win
win.document.open();
// формирование строки str для вывода
// содержимое в заголовок передается из переменной ans
var str = "<h1>"+ans+"!!!</h1><hr>";
// вывод строки в документ окна win
win.document.write(str);
// закрываем вывод в документ окна win
win.document.close();
}
</script>
</head>
<body>
<p> Отметьте правильные утверждения:</p>
<form>
<input type="checkbox" id="c1" name="c1"> 2*2=4 <br/>
<input type="checkbox" id="c2" name="c2"> 2*5=30<br/>
```

<input type="checkbox" id="c3" name="c3"> 6*8=48
<input type="button" value="Проверить" onClick="checkAll();">
</form>
</body>

Пример 3

Напишите приложение, которое проверяет, правильно ли выбрана радиокнопка. Вывести результат методом alert.



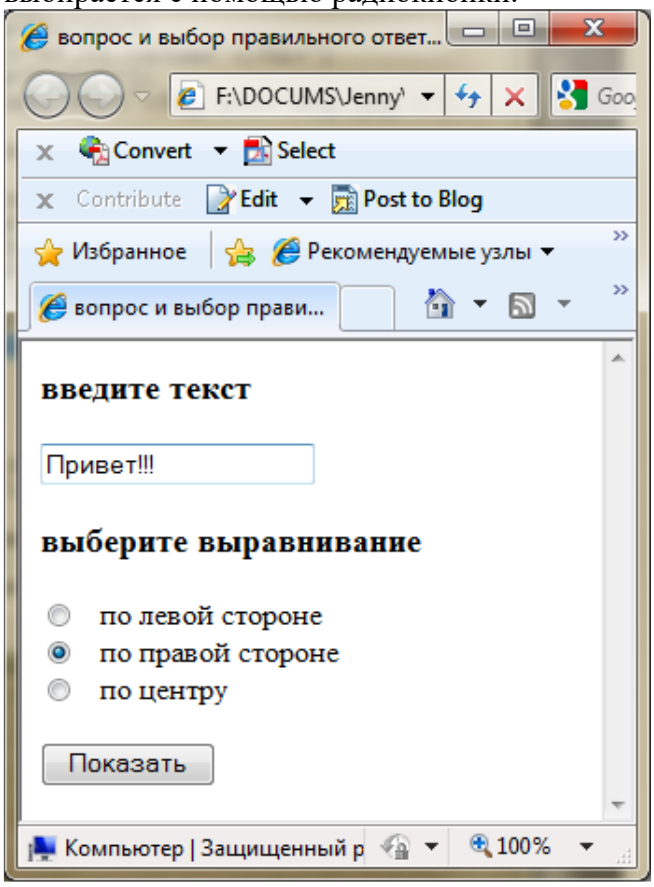


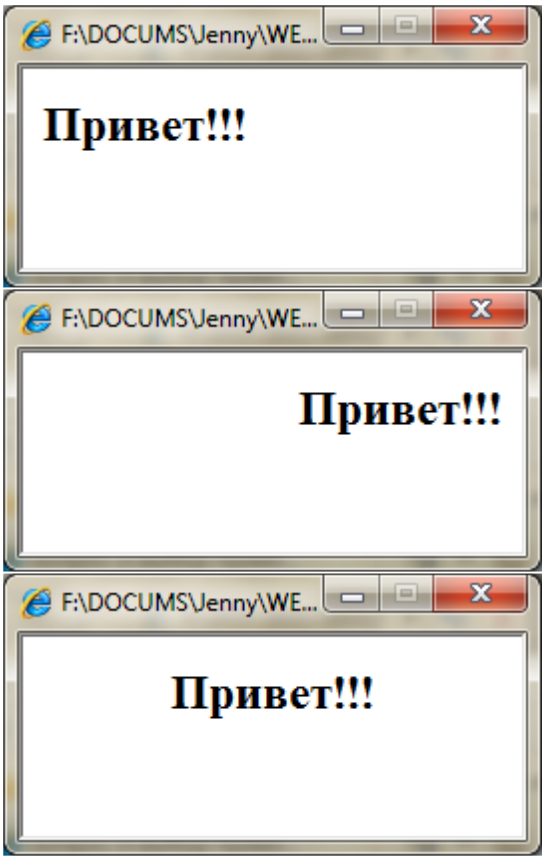

```
<html>
<head>
<script type="text/JavaScript">
// получить значение из радиокнопок
function validate()
{
// переменная p1 радиокнопка с именем k2
var p1=document.getElementById('k2');
// если выбрана вторая радиокнопка
if (p1.checked) alert("верно");
else alert("неверно");
}
</script>
</head>
<body>
<h2>
что возвращает метод prompt объекта window?
</h2>
<form>
<h3>
<input type='radio' id='k1' name='v' checked />   целое число <br />
<input type='radio' id='k2' name='v' />   строку <br />
<input type='radio' id='k3' name='v' />   вещественное число <br />
</h3>
<input type="button" value="Проверить" onClick="validate()" />
</form>
</body>
```

</html>

Пример 4

Напишите приложение, которое создает новое окно и выводит в него введённый в текстовое поле формы текст в виде заголовка. Вид выравнивания для заголовка выбирается с помощью радиокнопки.






```
<html>
<head>
<script type="text/JavaScript">
// получить значение из радиокнопок
function validate()
{
// переменная p текстовое поле с именем tx
var p=document.getElementById('tx');
// переменная textt содержит текст из текстового поля
var textt=p.value
// переменная p1 радиокнопка с именем k1
var p1=document.getElementById('k1');
// переменная p2 радиокнопка с именем k2
var p2=document.getElementById('k2');
// переменная p3 радиокнопка с именем k3
var p3=document.getElementById('k3');
// значение переменной vt – вид выравнивания
// если выбрана 1-я радиокнопка
if (p1.checked) vt='left'
// если выбрана 2-я радиокнопка
if (p2.checked) vt='right'
// если выбрана 3-я радиокнопка
if (p3.checked) vt='center'
```

```
var win=window.open("","","width=100,height=50");
win.document.open();
// переменная st содержит
// сформированную таблицу стилей нового документа
var st="<head><style> h2 {text-align:"+vt+";}</style></head>"
win.document.write(st);
// переменная st содержит
// сформированное тело нового документа
var st="<body><h2>"+textt+"</h2></body>"
win.document.write(st);
win.document.close();
}
</script>
</head>
<body>
<form>
<h3>введите текст</h3>
<input type="text"id="tx" name="tx" value="">
<h3>выберите выравнивание</h3>
<input type='radio' id='k1' name='v' checked />
   по левой стороне <br />
<input type='radio' id='k2' name='v' />
   по правой стороне <br />
<input type='radio' id='k3' name='v' />
   по центру <br />
<br />
<input type="button" value="Показать" onClick="validate()" />
</form>
</body>
</html>
```

Свойства объекта option

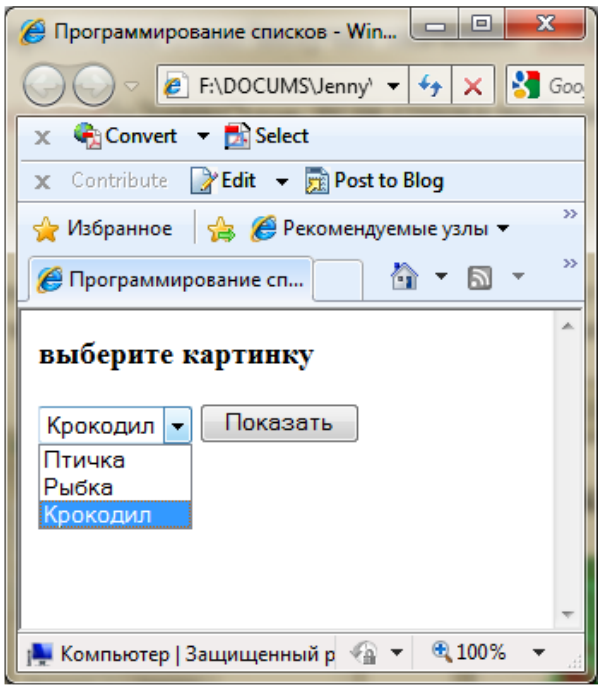
Для создания раскрывающегося списка используется элемент <select>, который включает в себя несколько элементов <option>, по одному элементу на каждую строку списка.

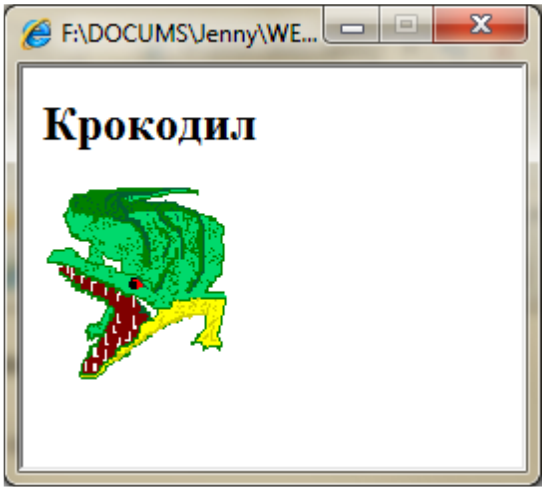
Каждый из этих объектов option имеет следующие свойства:

- text – содержит текст соответствующей строки списка.
- selected – равно true, если строка выбрана, и false – в противном случае.

Пример 5

Напишите приложение, которое выбирает картинку из списка и при нажатии на кнопку выводит в новое окно выбранное название в виде заголовка и саму картинку.





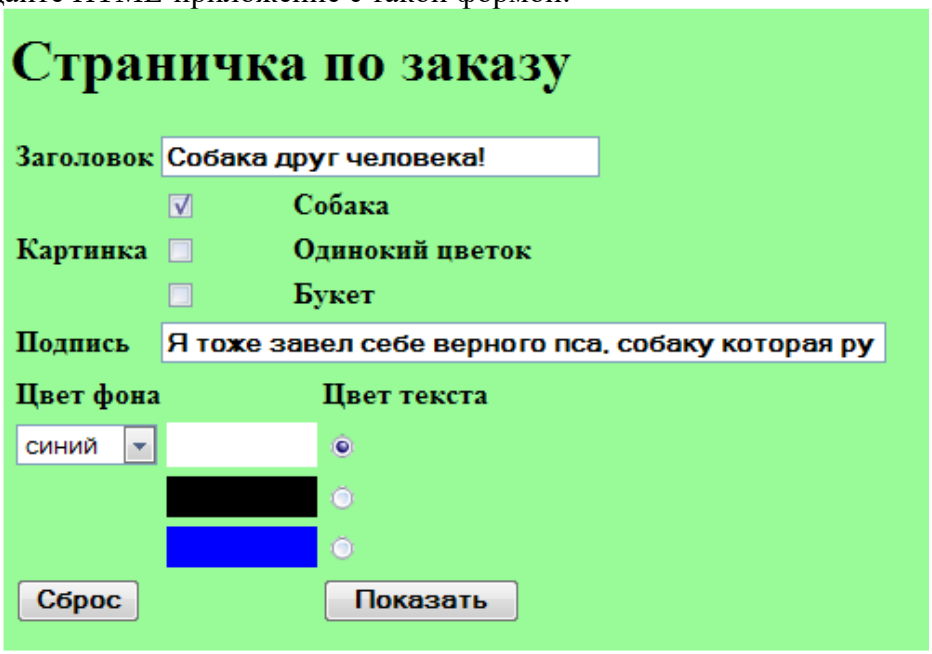

```
<html>
<head>
<title>Программирование списков</title>
<script type="text/JavaScript">
function GO()
{
// переменная p1 элемент списка с именем k1
var p1=document.getElementById('k1');
// переменная p2 элемент списка с именем k2
var p2=document.getElementById('k2');
// переменная p3 элемент списка с именем k3
var p3=document.getElementById('k3');
// формирование постоянной части нового документа
st1="<body><h2>"
st3="</h2><img src="
st5=" /></body>"
// формирование переменной части нового документа
if (p1.selected)
{ st2=p1.text ; st4="'2.GIF"' }
if (p2.selected)
{ st2=p2.text; st4="'200.png"'}
if (p3.selected)
{ st2=p3.text; st4="'ALLIGATO.gif"'}
// открытие нового окна
var win=window.open("","","width=100,height=200");
win.document.open();
// сформированный документ
str=st1+st2+st3+st4+st5
// вывод в окно
win.document.write(str);
win.document.close();
}
</script>
</head>
```

```
<body>
<form>
<h3>выберите картинку</h3>
<select>
<option id='k1'>Птичка</option>
<option id='k2'>Рыбка</option>
<option id='k3'>Крокодил</option>
</select>
<input type="button" value="Показать" onclick="GO()">
</form>
</body>
</html>
```

Задания на практическую работу

Вариант 1

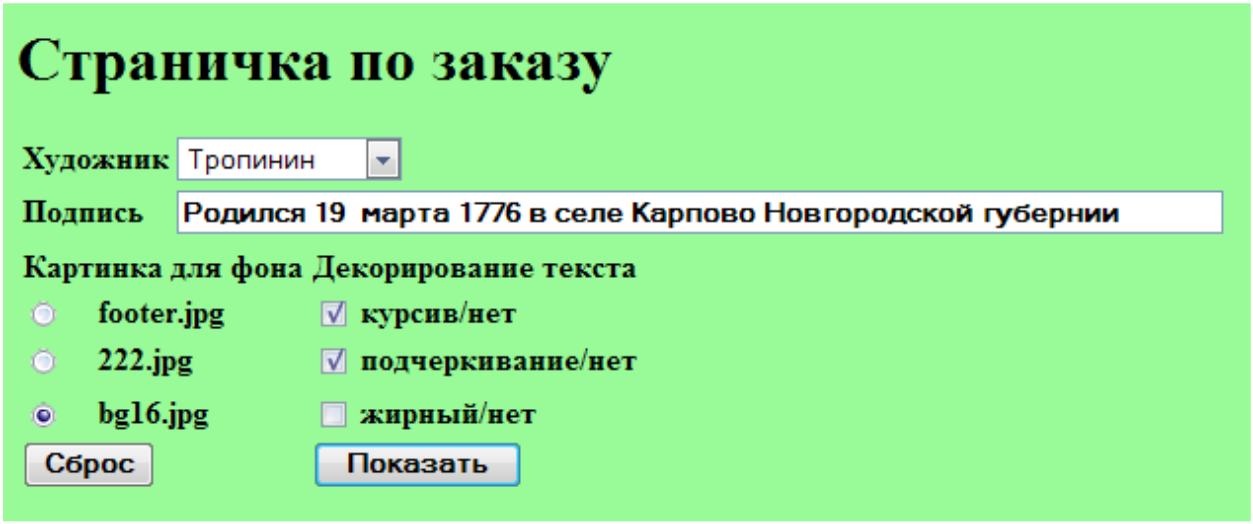
Создайте HTML-приложение с такой формой:



По нажатию на кнопку Показать приложение должно открыть новое окно и показать в нем заказанные картинки с короткими подписями. Новое окно должно создаваться “на лету” с использованием информации, которую ввел в форму пользователь. Картинки, любые, скопировать в свою папку.

Вариант 2

Создайте HTML-приложение с такой формой:



По нажатию на кнопку Показать приложение должно открыть новое окно и показать в нем портреты выбранного художника с короткими подписями. Новое окно должно создаваться “на лету” с использованием информации, которую ввел в форму пользователь. Картинки, любые, скопировать в свою папку.

8. Практическая работа №6 «Объект Image»

Цель работы: научиться пользоваться методами класса Image.

Все картинки (элементы img) в документе являются экземплярами объекта Image. У каждого объекта Image существует свойство src, которое можно менять. Используя это можно вносить изменения в графические образы, присутствующие на web-странице.

События мыши MouseOver и MouseOut

События mouseover и mouseout происходят, когда мышиный курсор перемещается на элемент или соответственно уходит за его пределы. Обработчики событий имеют имена onmouseover и onmouseout соответственно. В примере 1 будем изменять это свойство в зависимости от того, какое наступает событие mouseover или mouseout.

Пример 1

В данном примере при наведении курсора мыши на рисунок, на его месте появляется другой рисунок.

Для обработки событий создана функция doEvent с логическим параметром. При наведении курсора на картинку (возникает событие MouseOver) вызывается функция doEvent с параметром true. В этом случае у картинки меняется значение свойства src на flower.jpg. При уведении курсора за картинку (возникает событие onMouseOut) вызывается функция doEvent с параметром false и значение этого свойства восстанавливается на bigdaisy.jpg.

```
<head>
<style>
body {background-color:#dfd8c5;}
img {width:274; height:233} </style>
<script>
function doEvent(type)
{
// переменная cv объект img с именем pic
var cv=document.getElementById('pic');
// если параметр функции равен true, то
// меняем свойство src объекта cv на flower.jpg
if(type) cv.src="flower.jpg";
// в противном случае показываем
// картинку с именем bigdaisy.jpg
else cv.src="bigdaisy.jpg";
}
</script>
</head>
<body>
<h1>События MouseOver и MouseOut</H1>
<p>Переведи мышиный курсор на цветок!</p>
<img name ="pic" id='pic' src="bigdaisy.jpg"
onMouseOver="doEvent(true);" onMouseOut="doEvent(false);">
</body>
```

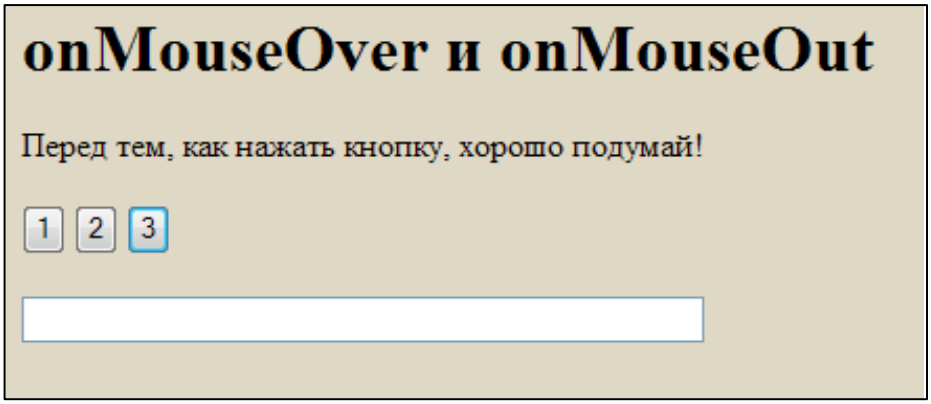
Пример 2

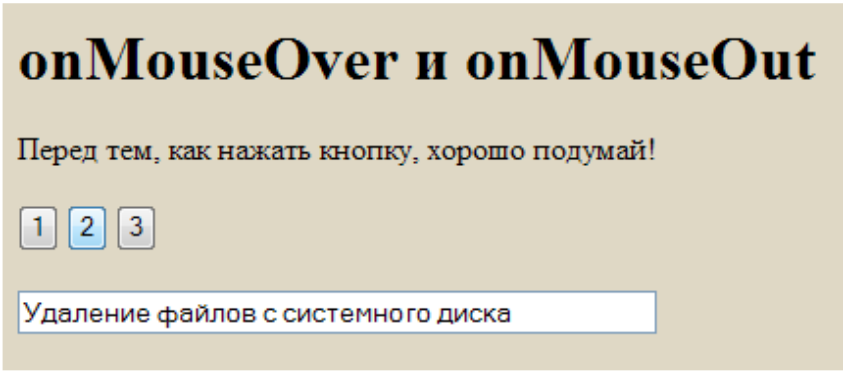
В данном примере при наведении курсора мыши на кнопку, в текстовом поле формы появляется подсказка о назначении этой кнопки. При уведении курсора в сторону подсказка исчезает.

В скипте создан массив mess, содержащий все варианты подсказок.

Для обработки событий создана функция doEvent с параметром, который совпадает с одним из индексов массива. Функция выводит значение элемента массива с этим индексом в поле info формы.

При наведении курсора на любую из кнопок (возникает событие onMouseOver) параметру присваивается значение 1, 2 или 3. По этому событию вызывается функция, которой передаются эти параметры и выводятся в поле соответствующий элемент массива mess. При уведении курсора с кнопки (возникает событие onMouseOut) функции передается значение 0 и в поле выводится элемент массива с индексом 0, а именно пустая строка.





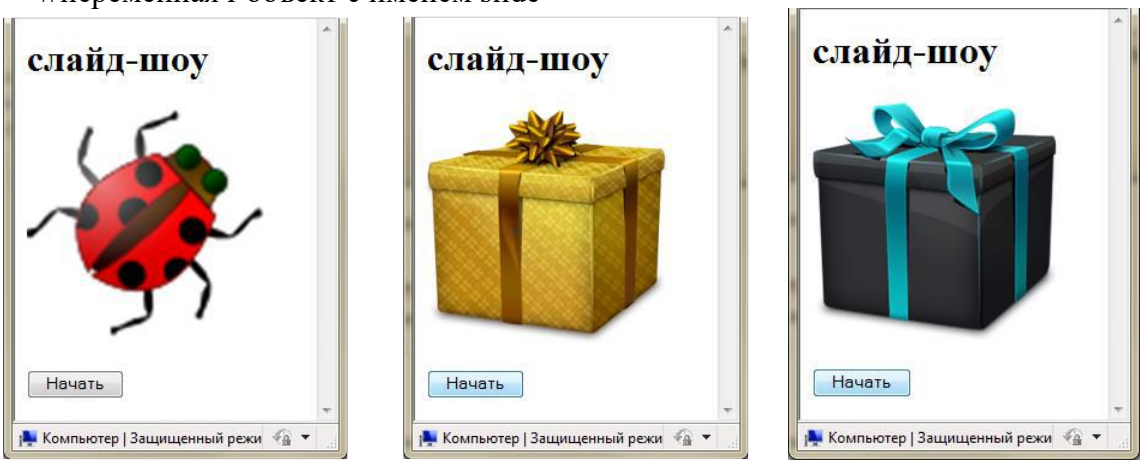

```
<head>
<style>body{background-color:#dfd8c5;}</style>
<script>
var mess = new Array("","Выключение компьютера",
"Удаление файлов с системного диска",
"Форматирование винчестера");
function doEvent(num)
{
// переменная cv объект с именем info
var cv=document.getElementById('info');
// меняем свойство value объекта cv на соответствующий элемент массива
cv.value= mess[num];
}
</script>
</head>
<body>
<h1>onMouseOver и onMouseOut</h1>
<P>Перед тем, как нажать кнопку, хорошо подумай!
<form>
<input type="button" value=" 1 "
onMouseOver="doEvent(1);" onMouseOut="doEvent(0);">
<input type="button" value=" 2 "
onMouseOver="doEvent(2);" onMouseOut="doEvent(0);">
<input type="button" value=" 3 "
onMouseOver="doEvent(3);" onMouseOut="doEvent(0);">
<br/><br/>
<input name="info" id='info' type="text" value="" size="50">
</form>
</body>
```

Пример 3

В данном примере после нажатия на кнопку, картинка в окне меняется автоматически, каждые несколько секунд.

```
<head>
<style> img{width:200;height:200}</style>
<script>
i=0;
function slideShow()
{
// массив картинок
ris = new Array('7.png', '8.png', '9.png', '6.png');
// Проверяем, Не вышел ли счётчик за пределы массива
```

```
if (i >= 4) i = 0; // если вышел, обнуляем счётчик.
//переменная r объект с именем slide
```




```
r=document.getElementById('slide')
// Меняем свойство src этого объекта на элемент из массива
r.src=ris[i];
i++; // Увеличиваем счётчик
// вызываем эту же функцию через каждые 1500 мс
setTimeout("slideShow()",1500);
}
</script>
</head>
<body>
<h1> слайд-шоу </h1>
<img id='slide' src="3.png" />
<form>
<input type="button" value=" Начать " onClick="slideShow() ;">
</form></body>
```

Задания на практическую работу

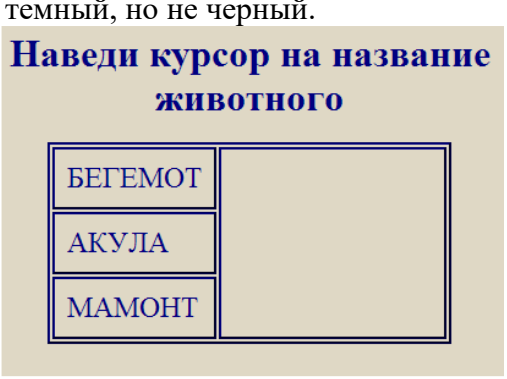
Вариант 1

1. В задании при наведении курсора мыши на фото автомобиля, в текстовом поле формы появляется сообщение о его марке и стоимости. При уведении курсора в сторону сообщение исчезает. Фон страницы любой светлый. Цвет текста темный, но не черный.

2. Организовать показ слайд-шоу из этих же картинок, чтобы каждая следующая картинка показывалась не автоматически, а при нажатии кнопки.

Вариант 2

1. В задании при наведении курсора мыши на названии животного, в ячейке таблицы появляется рисунок этого животного. При уведении курсора в сторону рисунок исчезает (то есть подставляется пустой рисунок). Фон страницы любой светлый. Цвет текста темный, но не черный.



2. Организовать показ слайд-шоу из этих же картинок, чтобы каждая следующая картинка показывалась не автоматически, а при нажатии кнопки.

9. Практическая работа №7 «Свойство style. Объект style и его свойства»

Цель работы: научиться пользоваться методами класса style.

У всех HTML-элементов есть свойство style. Значением этого свойства является объект style, чьи собственные свойства позволяют изменять установки стилей.

Свойства объекта style

Свойства объекта style позволяет изменить стиль любого элемента Web-страницы, просто присвоив нужному свойству необходимое значение.

Существует четкое соответствие между свойствами стиля CSS и свойствами объекта style в JavaScript .

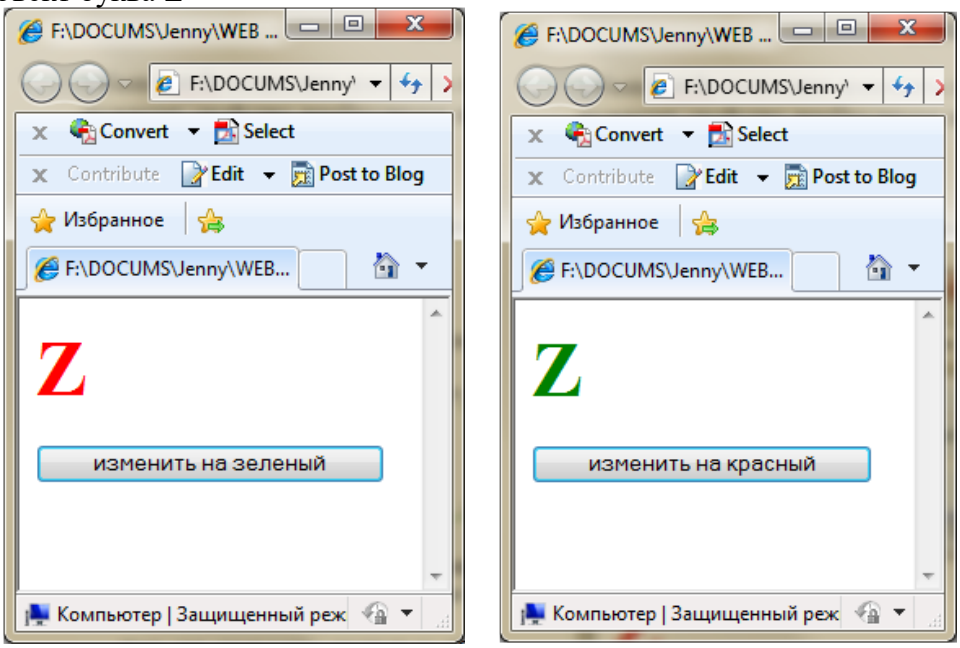
Свойства объекта style в JavaScript имеют почти такие же имена что и в CSS за тем исключением, что символы "-" убираются, а первые буквы всех слов, образующих имя атрибута, кроме первого, делаются прописными (таблица 9.1). В следующей таблице показаны примеры преобразования имен атрибутов стиля в имена свойств объекта style, устанавливающих стиль элемента.

Таблица 9.1 – Свойства объекта style в JavaScript и CSS

CSS	JavaScript
background-attachment	backgroundAttachment
font-family	fontFamily
z-index	zIndex

Пример 1
По нажатию кнопки будем изменять цвет буквы и надпись на кнопке.

```
<head>
<style>
h1
{color:red; font-size:36pt;}
</style>
<script>
i=0;
function f()
{
// объект буква Z
```




```
r1=document.getElementById('e1')
// объект кнопка
r2=document.getElementById('e2')
if (i==0)
{
// у объекта r1 есть свойство style. Это тоже объект
// меняем у объекта style свойство color
r1.style.color='green';
// меняем у объекта r2 свойство value
r2.value='изменить на красный';
i=1;}
else
{
r1.style.color='red'; r2.value='изменить на зеленый';
i=0
```

```
}
}
</script>
</head>
<body>
<h1 id='e1'> Z </h1>
<form>
<input type="button" id='e2' value="изменить на зеленый"
onClick="f();">
</form>
</body>
```

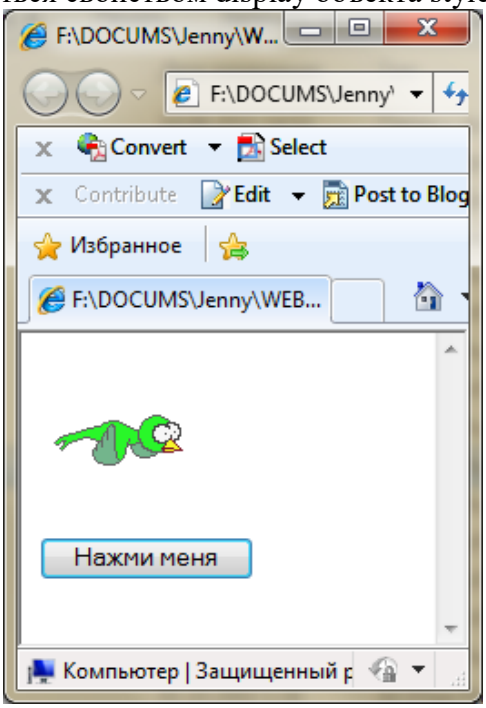
Свойство display

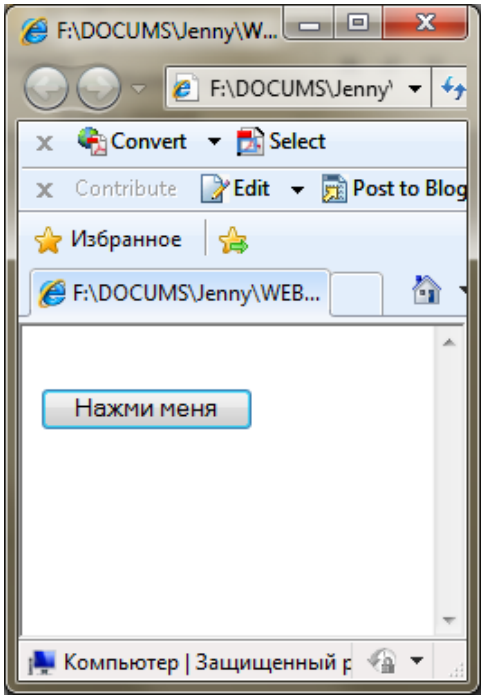
Свойство display задает способ отображения элемента на странице. Некоторые значения:

- inline — элемент ведет себя как линейный;
- block — элемент ведет себя как блочный;
- none — "удаляет" элемент со страницы вместе с содержимым. Элемент не видим, на странице блоки располагаются так, словно элемента нет.
-

Пример 2

По нажатию кнопки картинка будет появляться и исчезать. Для этого будем пользоваться свойством display объекта style.






```
<head>
<script>
function displ()
{
p=document.getElementById('st')
// у объекта p есть свойство style. Это тоже объект
// меняем у объекта style свойство display
if (p.style.display == 'none')
{p.style.display = 'block'}
else {p.style.display = 'none'}
}
```

```
</script>
</head>
<body>
<img src="bird2.gif" id='st' />
<br />
<input type="button" value="Нажми меня" onclick="displ();" >
</body>
```

Основы перемещения элементов

Движение объекта на Web-странице осуществляется путем изменения свойств, задающих его координаты. В CSS координаты объекта задаются с помощью свойств left и top. Свойство left задает горизонтальную координату объекта, которая в случае позиционирования в абсолютных координатах (position:absolute) задает расстояние в пикселях от левой границы экрана. Свойство top задает вертикальную координату объекта, которая в случае позиционирования в абсолютных координатах задает расстояние в пикселях от верхней границы экрана. Если задать элементу абсолютное позиционирование и изменять его координаты top и left, то элемент будет двигаться.

Метод setTimeout (expression, msec)

Метод setTimeout выполняет выражение или функцию по истечении установленного количества миллисекунд.

expression строковое выражение, содержащее имя вызываемой функции, msec числовое значение в миллисекундах. Выражение выполняется однократно и для его повторного выполнения требуется очередной вызов метода.

Пример 3

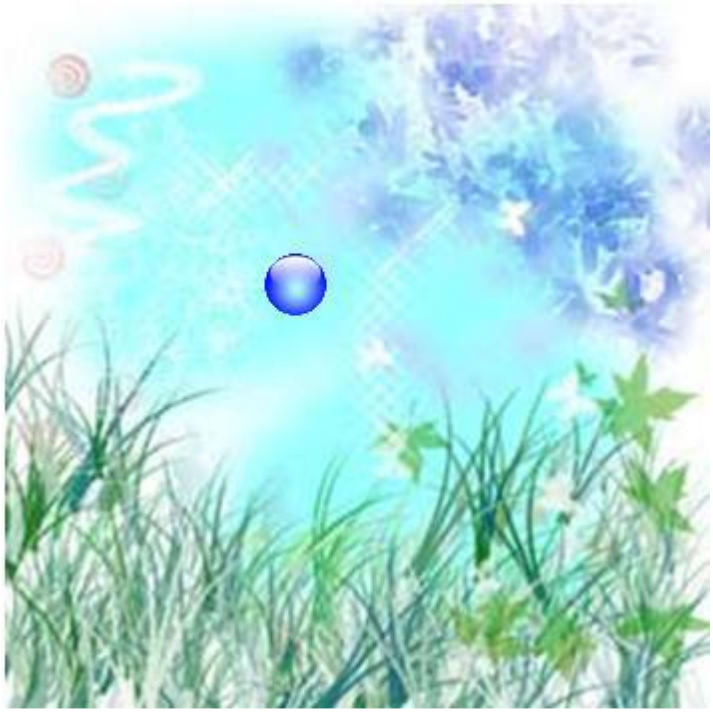
Создадим документ, в котором рисунок p5_0.jpg занимает 50% окна и является нижним слоем. На верхнем слое второй рисунок bird2.gif, который двигается вправо. Когда второй рисунок достигает края первого рисунка, он останавливается.

```
<head>
<style>
#im2 {position:absolute;left:0;top:100}
</style>
</head>
<body>
```

```
<img src="p5_0.jpg" id="im1" width="50%" />
<img src="bird2.gif" id="im2" />
<script type="text/JavaScript">
function f()
{
// сдвиг рисунка на 10
x=x+10;
// изменяем свойство left 2-го рисунка
r2.style.left=x
// если он не дошел до края 1-го рисунка
// то вызываем каждые 1000 мили сек эту же функцию
if (x<x1-x2) setTimeout("f()",700);
}
// начало скрипта
x=0
r1=document.getElementById('im1')
r2=document.getElementById('im2')
x1=r1.width //ширина фона
x2=r2.width //ширина птички
f();
</script>
</body>
```

Пример 4

Изменим предыдущий документ так, чтобы рисунок 4.gif двигался по главной диагонали рисунка p5_0.jpg вниз. При достижении правого нижнего угла рисунок движется вверх по этой же диагонали. При достижении левого верхнего угла этот рисунок снова движется вниз. И так далее.



Шаг – расстояние слева и сверху от левого верхнего угла неподвижного рисунка до движущегося рисунка.

Если рисунок движется вправо и вниз, то шаг положительный. Если рисунок движется влево и вверх, то шаг отрицательный.

```
<html>
<head>
```

```
<style>
#im2 {position:absolute;left:0;top:0}
</style>
</head>
<body>
<img src="p5_0.jpg" id="im1" width="50%" />
<img src="4.gif" id="im2" />
<script type="text/JavaScript">
function f()
{
// сдвиг рисунка на шаг s
x=x+s; y=y+s;
r2.style.left=x
r2.style.top=y
// если долетел до правого края меняем шаг на отрицательный
if (x>=x1) s=-10
// если долетел до левого края меняем шаг на положительный
if (x<=0) s=10
setTimeout("f()",50);
}
x=0 ; y=0; s=10;
r1=document.getElementById('im1')
r2=document.getElementById('im2')
x1=r1.width //ширина фона
f();
</script>
</body>
</html>
```

Задания на практическую работу

Вариант 1

1. Создайте документ, в котором рисунок заключен в рамку серого цвета типа inset и толщиной 50. По нажатию кнопки изменять цвет рамки на синий и надпись на кнопке. При следующем нажатии изменять цвет рамки на серый и надпись на кнопке.

2. Сначала на странице только текст стихотворения и кнопка с надписью «показать свечу». При нажатии на кнопку текст стихотворения прячется, на его месте появляется рисунок, и надпись на кнопке меняется на «показать текст». Если еще раз нажать на кнопку, рисунок исчезает и появляется текст. Параметры форматирования страницы:

Цвет фона темно-синий, размер шрифта 22, курсив, цвет шрифта белый.

Рамка вокруг текста: цвет #ffA089, тип outset, толщина 5.

Картинка размером 200 на 200.

Рамка вокруг картинки: цвет #ffA089, тип outset, толщина 50, отступы по 5.

3. Создайте документ, в котором рисунок занимает 50% окна и является нижним слоем. На верхнем слое второй рисунок, который двигается по горизонтали к правой границе первого рисунка, затем на строку (высоту рисунка) вниз и назад, к левой границе, на строку вниз и к правой границе, и т.д. до нижней границы первого рисунка.

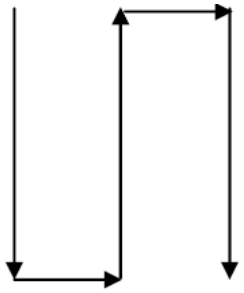
Вариант 2

1. Создайте документ, в котором блоковый элемент размерами 250 на 250 и заливкой красного цвета заключен в рамку темно-синего цвета типа inset и толщиной 10. По нажатию кнопки изменять цвет заливки на голубой и надпись на кнопке. При следующем нажатии изменять цвет заливки на красный и надпись на кнопке.

2. Сначала на странице две картинки в рамках и две кнопки с надписью «спрячь меня». При нажатии на кнопку картинка прячется, и надпись на кнопке меняется на «покажи меня». Если еще раз нажать на кнопку, картинка появляется и меняется надпись на кнопке. Параметры форматирования страницы:

- Цвет фона темно-синий.
- Картинки размером 200 на 200.
- Рамка вокруг первой картинки: цвет #ffA089, тип outset, толщина 50, отступы по 5.
- Рамка вокруг второй картинки: цвет # 89ffA0, тип outset, толщина 50, отступы по 5.
- Рекомендуется: картинки и кнопки поместить в таблицу, функцию сделать с параметром. Параметр – номер картинки.

3. Создайте документ, в котором рисунок занимает 50% окна и является нижним слоем. На верхнем слое второй рисунок, который двигается по вертикали к нижней границе первого рисунка, затем на столбец (ширину рисунка) вправо и вверх, к верхней границе, на столбец вправо и к нижней границе, и т.д. до правой границы нижнего рисунка.



10. Практическая работа №8 «Слои»

Цель работы: научиться пользоваться z-индексированием.

Для создания слоев следует использовать тег или . Эти теги взаимозаменяемы и различаются лишь внешним видом в браузере.

Если требуются отступы до и после текста, следует использовать элемент . При размещении текста внутри параграфа применяется тег .

В таблице 10.1 перечислены наиболее важные атрибуты.

Таблица 10.1 – Атрибуты слоя

id	Имя слоя, используемое для указания его в <script>
left	Позиция слоя по x координате
top	Позиция слоя по y координате
position	Задает относительную или абсолютную позицию относительно других объектов
z-index	Позиция слоя при наложении нескольких объектов друг на друга
width	Ширина слоя в пикселах или %
height	Высота слоя в пикселах или %
bgColor	Цвет фона слоя
background	Картинка фона
src	Внешний html документ, содержащийся в слое

Пример наложения текста:

```
<html>
<body>
Слой1 наверху
<div style="position:relative; font-size:50px; z-index:2; color: navy">
Слой 1
</div>
<div style="position:relative; top:-55; left:5; color:orange; font-size:80px;
z-index:1">
Слой 2
</div>
Слой 2 наверху
<div style="position:relative; font-size:50px; z-index:3; color: navy">
Слой 1
</div>
<div style="position:relative; top:-55; left:5; color:orange; font-size:80px;
z-index:4">
Слой 2
</div>
</body>
</html>
```

Тип позиционирования слоя определяется параметром position, положение элемента - двумя координатами top и left.

Кроме тегов и абсолютное позиционирование поддерживают следующие элементы: <applet>, <input>, <button>, <object>, <select>, <fieldset>, <iframe>, <table>, <img>, <textarea>.

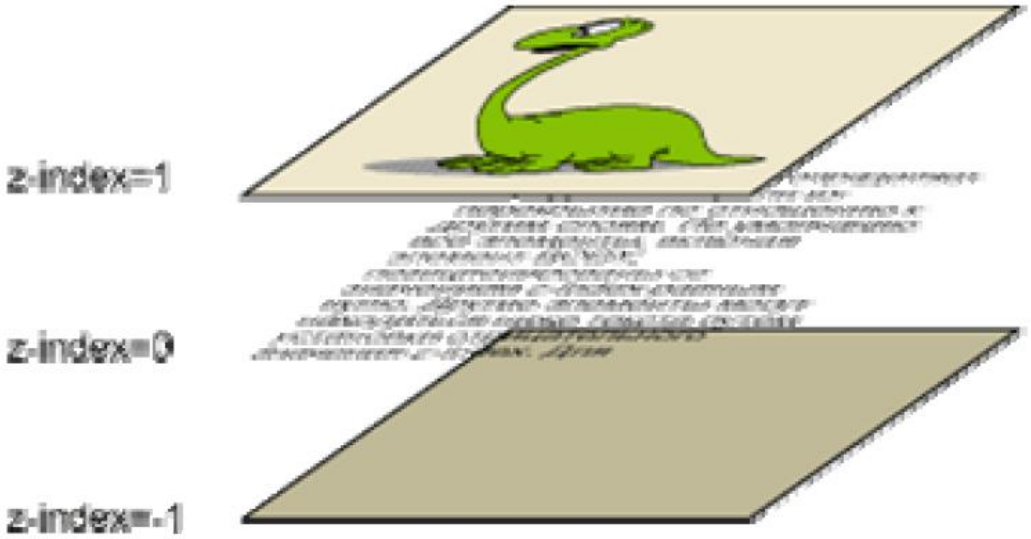
Параметр position:relative используется для смещения слоя относительно родительского элемента. Установка этого значения не изменяет размещение элемента, но если установлены значения свойств top или left, то слой смещается от своего нормального положения в документе.

В то время как свойство position указывает тип системы координат, параметры top и left определяют точную позицию слоя. Значения этих параметров могут определяться в процентном отношении или пикселах, принимать положительные и отрицательные

величины. Это дает возможность размещать контент выше или ниже на странице независимо от физической позиции кода HTML. То есть, в верхней части страницы можно поместить слой, который описан внизу HTML-документа.

Свойство z-index

Свойство z-index определяет порядок слоев, или их перекрытие по отношению к другим слоям. По умолчанию все слои позиционированы со значением z-index равным нулю. Другие слои могут размещаться ниже путем установки отрицательного значения z-index. Для слоев, у которых z-index не установлен, это значение назначается неявно в соответствии с их положением в документе. Поэтому слой, который помещен в документ позже, размещается выше остальных элементов, позиционированных ранее.



Свойства visibility и display

Для отображения или скрытия слоя используется свойство visibility. Он может принимать значения visible, установленное по умолчанию, для показа слоя, и hidden, которое его прячет.

Например, скрытый блок текста можно оформить следующим образом:

Спрятанный слой

При этом, когда используется данное свойство для скрытия элемента, соотвествующий данному элементу блок занимает прежнее положение на странице, но сома содержимо не отображается.

Чтобы на странице не оставалось пустого блока, соотвествующего скрываемому элементу, можно использовать свойство display со значением none. Для отображения элемента display равно block.

Динамическое управление слоями

Сценарии JavaScript позволяют динамически управлять параметрами установленных слоев. Это позволяет получить такие эффекты, как скрытие и отображение слоя, изменение порядка отображения, перемещение слоя в окне браузера. Все эти эффекты достигаются с помощью изменения соответствующих стилевых параметров установленных слоев.

Для обращения к слоям из сценариев JavaScript, удобнее всего каждому слою дать собственное имя при помощи параметра id. Например:

...

Для того, чтобы скрыть отображение слоя div1, можно использовать следующую команду:

div1.style.visibility='hidden';

Для повторного отображения слоя следует выполнить следующее присвоение:

```
div1.style.visibility='visible';
```

Пример динамической смены слоев: в данном примере для отображения некоторого слоя следует нажать на соответствующую ссылку. Эту идею можно применить и для организации выпадающих меню.

```
<html>
<head>
<style type="text/css">
div {
position: absolute;
top: 20;
left: 160;
visibility: hidden;
}
</style>
<script language="JavaScript">
function show_d(d)
{
div1.style.visibility='hidden';
div2.style.visibility='hidden';
div3.style.visibility='hidden';
div4.style.visibility='hidden';
div5.style.visibility='hidden';
d.style.visibility='visible';
}
</script>
</head>
<body>
<a href="javascript:void(0)" onClick="show_d(div1);">
показать слой 1
</a><br>
<a href="javascript:void(0)" onClick="show_d(div2);">
показать слой 2
</a><br>
<a href="javascript:void(0)" onClick="show_d(div3);">
показать слой 3
</a><br>
<a href="javascript:void(0)" onClick="show_d(div4);">
показать слой 4
</a><br>
<a href="javascript:void(0)" onClick="show_d(div5);">
показать слой 5
</a><br>
<div id="div1">
<h3>Слой номер один</h3>
```

Некоторый текст, на слое расположенный. Его можно скрыть и показать. Текст может содержать несколько строк.

```
</div>
<div id="div2">
<h3>Слой номер два</h3>
```

Содержит свой текст. Если показывается, то текст на других слоях не виден.

```
</div>
<div id="div3">
```

<h3>Слой номер три</h3>

Тоже текст. При работе со слоями надо следить, чтобы текст одного слоя не "выглядывал" из-под другого слоя при самых различных размерах окна браузера и используемых шрифтах.

<h3>Слой номер четыре</h3>
Здесь нет текста.

<h3>Слой номер пять</h3>
И тут тем более нет.

</body>
</html>

Динамическое изменение цвета фона ячеек

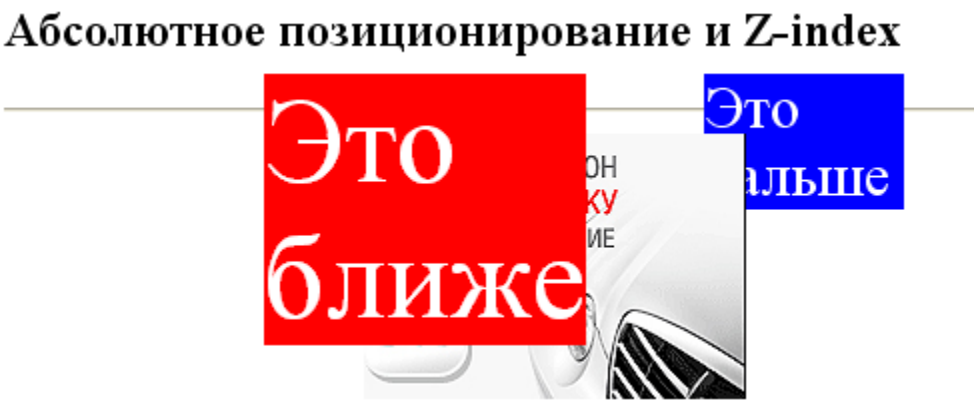
Использование стилей и управление ими с помощью JavaScript позволяет менять вид ячейки "на ходу", при выполнении определенных условий, таких как наведение курсора на ссылку или саму ячейку.

Рассмотрим самый простой прием - цвет фона ячейки меняется, когда курсор мыши наводится на нее. Наведение мыши на область отслеживается событием onMouseover, а вывод мыши за ее пределы - событием onMouseout. Поскольку цвет фона меняется у той же самой ячейки, на которую наводим курсор мыши, то изменение стиля делается с помощью метода this.style.background.

…
<table width=60% border=1 cellspacing=0 cellpadding=4
bordercolor=#333333 align=center>
<tr>
<td align=center bgcolor=#cccccc
onMouseover="this.style.background='#ffcc33'"
onMouseout="this.style.background='#cccccc'">Пункт
1</td>
<td align=center bgcolor=#cccccc>Пункт 2</td>
</tr>
</table>

Задания на практическую работу

Создайте HTML-страницу, на которой будет три слоя. Верхний и нижней представляют из себя статичные квадраты разного цвета с текстом, а меду ними должна проплывать любая картинка слева направо.



Список литературы

1. Пол Вилтон, Джереми МакПик. JavaScript. Руководство программиста. СПб: Питер, 2009-720 с.

2. Стоян Стефанов. JavaScript. Шаблоны. СПб: Символ-плюс, 2011-272 с.

3. Дунаев В.. HTML, скрипты и стили. СПб: БХВ-Петербург, 2011- 816 с.

4. Дронов В.. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. СПб: БХВ-Петербург, 2011- 416 с.

5. Джон Поллок. JavaScript. Руководство разработчика. СПб: Питер, 2011-544 с.

6. Дэвид Макфарланд. JavaScript. Подробное руководство. Эксмо, 2009- 608 с.

7. Климов А. JavaScript на примерах. СПб: БХВ-Петербург, 2009-336 с.

8. Шафер С., HTML, XHTMLи CSS. Библия пользователя. М.: Вильямс, 2010 – 656 с.

9. Лабберс К., Олберс Н., Салим К.. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений.М.: Вильямс, 2011 – 272 с.

10. http://www.w3.org