JavaScript

JavaScript

1. Синтаксис языка
2. События и функции
3. Встроенные объекты
4. Объект *Window*
5. Обращение к элементам формы
6. Объект *Image*
7. Объект *Style*

https://learn.javascript.ru/

JavaScript

1 Синтаксис языка

Web-документ, отображаемый браузером, – это результат исполнения программ, созданных на языках разметки, стилей и сценариев:

1. Для описания *структуры* документа используется язык разметки (*html*).
2. Для описания *внешнего вида* документа используется язык стилей (*css*).
3. Для описания *поведения* документа, его реакции на действия пользователя используется язык сценариев (*JavaScript*).

Подключение к странице

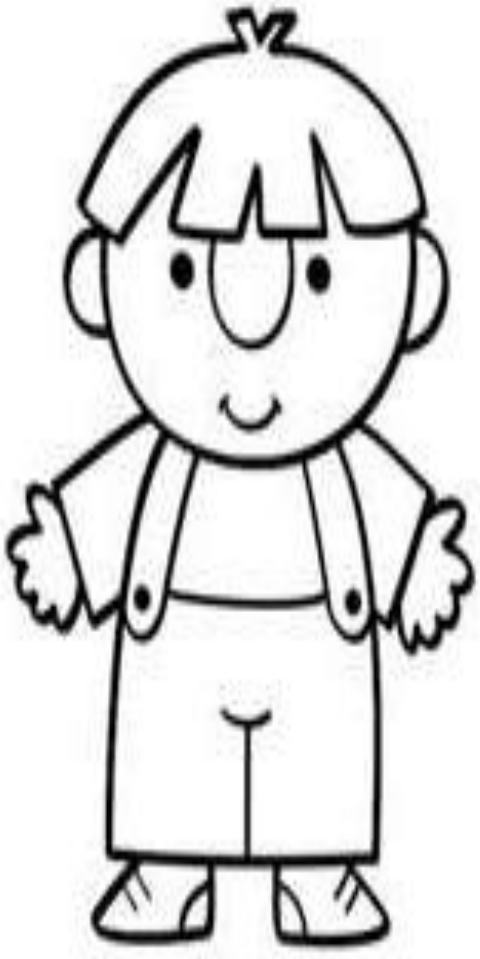
JavaScript-код исполняет браузер. В него встроен интерпретатор JavaScript.

Следовательно, выполнение программы зависит от того, когда этот интерпретатор получает управление.

Существует *четыре способа* размещения кода JavaScript на странице.

Теговые контейнеры Web-документа

<HTML>



<HEAD>

<TITLE> Заголовок документа </TITLE>

</HEAD>

<BODY>

Содержание документа

</BODY>

</HTML>

В теговом контейнере <body>...</body>

1 способ размещения скрипта

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

В теговом контейнере <head>...</head>

2 способ размещения скрипта

```
<head>
…
<script  type=”text/javascript”>
 Команды сценария
</script>
…
</head>
```

Используется, если скрипт представляет собой функцию, которая вызывается в ответ на какое-либо событие.

Во внешнем файле

3 способ размещения скрипта

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

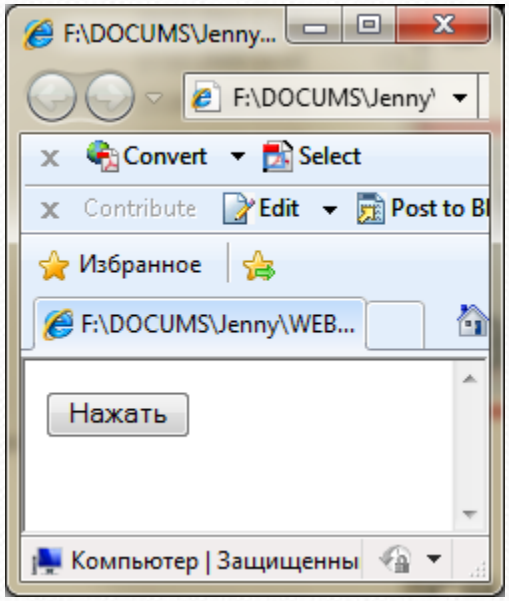
По аналогии с тем, как стили подключаются к странице с помощью элемента link, сценарии подключаются с помощью элемента script, только файл имеет расширение не .css, а .js.

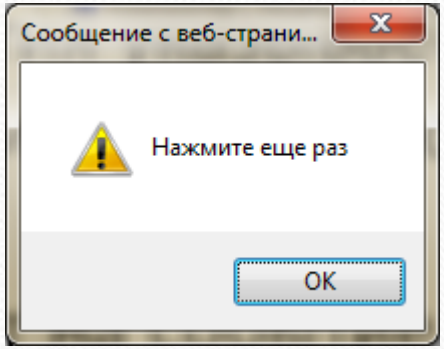
Непосредственно в теге

4 способ размещения скрипта

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

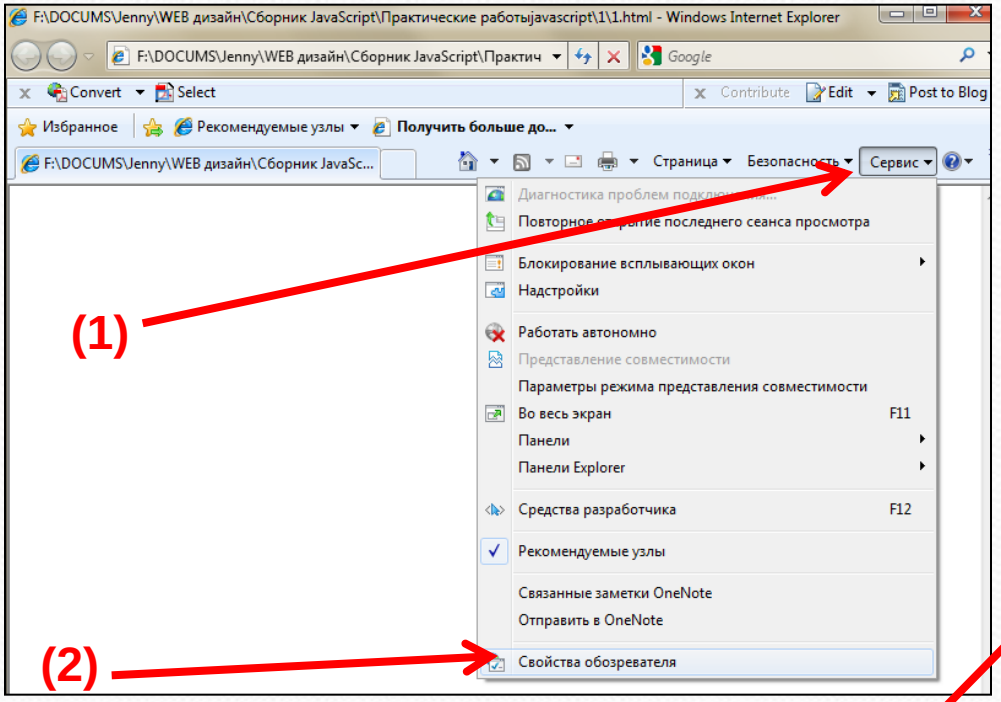
Обработчик события указывается прямо в теге в ответ на какое-либо событие, без заключения в теги <script > </script>.

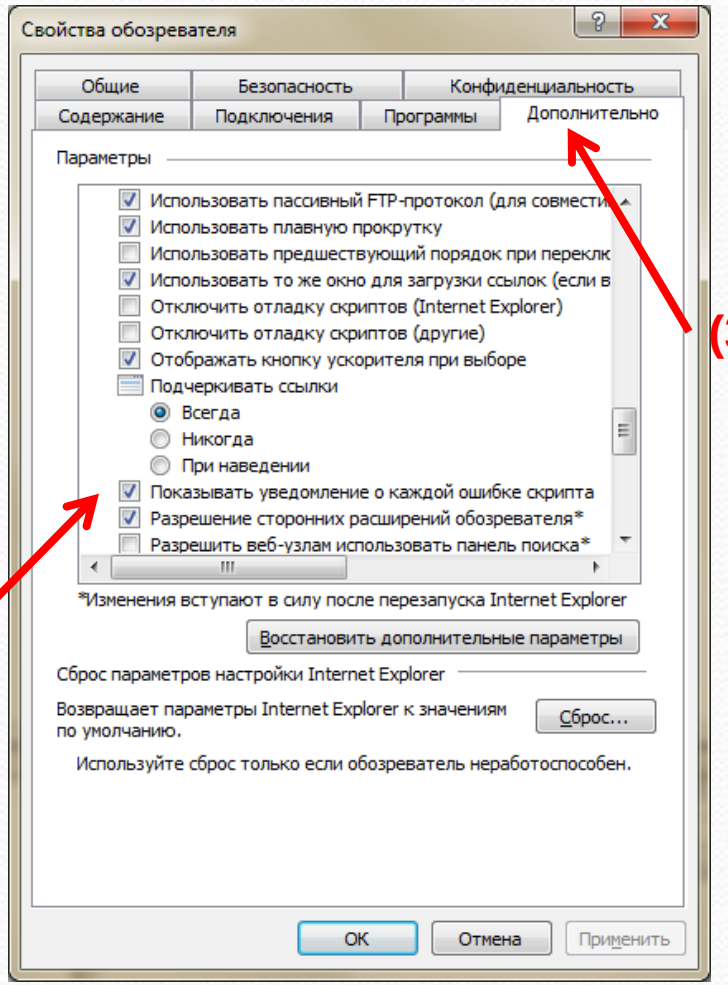




В примере, в ответ на событие **onClick** (кнопку нажали и отпустили) выводится сообщение.

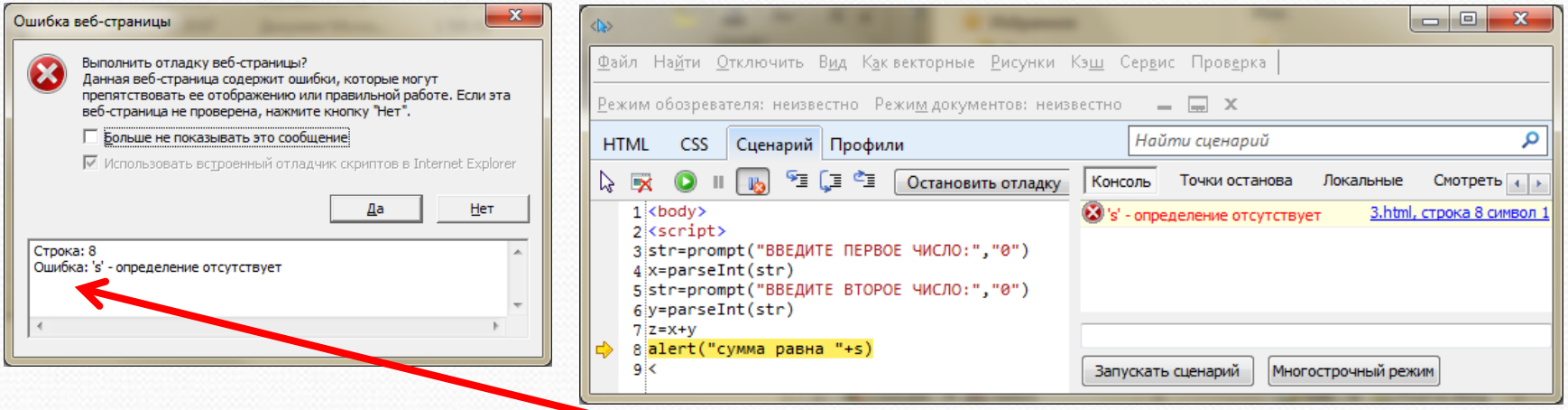
Отладка скрипта





Для того чтобы браузер выводил на экран сообщения об ошибке его необходимо настроить.

Отладка скрипта



В сообщении об ошибке указывается номер строки, в которой обнаружена ошибка, а также ее тип.

Нумерация строк отсчитывается от первой строки всего документа. Пустые строки тоже считаются.

https://learn.javascript.ru/debugging-chrome

Синтаксис языка

- JavaScript –зависит от регистра.
  Имена JavaScript и Javascript - разные имена!!
  Все ключевые слова - в нижнем регистре.
- Требования к именам переменных как в C++.
- 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:

```
var s = 1203   // целочисленная переменная
var d = 3.14   // переменная с плавающей точкой
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.

Математические операции

Оператор или операция	Действие	Пример	Значение, которое примет x
+	Сложение	x=100+5 str2='Начало' str1=str2+' конец'	105 Начало конец
-	Вычитание	x = 100-5	95
*	Умножение	x = 2*3	6
/	Деление	x = 12/2	6
%	Остаток от деления	x = 16%3	1
++	Инкремент	x = 2; x++;	3
--	Декремент	x = 2; x– –;	1

Операторы присваивания

Опера-тор	Действие	Пример	Значение x	Аналог записи
=	Присваивает значение переменной	**x = 1000**	1000	
+=	Увеличивает значение переменной	x+=100	1100	x = x+100
-=	Уменьшает значение переменной	x -= 12	988	x = x-12
*=	Умножает значение переменной	x *= 2	2000	x = x*2
/=	Делит значение переменной	x /= 2	500	x = x/2
%=	Возвращает остаток от деления	x %= 480	40	x = x%480

Операции сравнения

Операция	Действие	Пример
==	Равно	x == 10
!=	Не равно	x != 5
>	Больше	x > 0
<	Меньше	x < 4
>=	Больше либо равно	x >= y
<=	Меньше либо равно	x <= 5

Логические операции

Операция	Действие	Пример
&&	Логическое «И» (and)	x>=2 && y>=2
||	Логическое «ИЛИ» (or)	x>0 || y>0
!	Логическое «НЕ» (not)	!(1 < x && x < 10)

Условный оператор

if (a == 2) z = 2; else z = 3

if (x >= 2 && x <= 6)
{y = 0; z = 1}
else
{y = 1; z = 0}

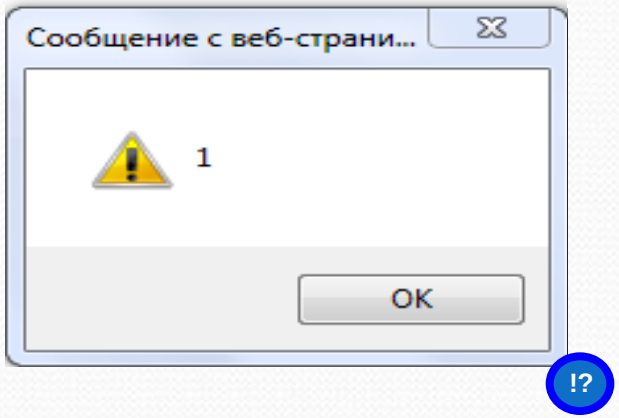
Отличия от C++.

1.Then отсутствует.

2.Точка с запятой перед else ставится, если else находится на той же строке.

Оператор цикла for

```
<script>
var i;
var k=0;
var sum = 0;
for (i=0; i<=2; i=i+0.5)
{
sum=sum+ i;
k=k+0.2;
}
alert(sum); alert(k)
</script>
```



Ввод/вывод данных
Диалоговые окна

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

1. alert
2. confirm
3. prompt

Метод alert(“строка”)

Метод alert используется для вывода простейшего диалогового окна, содержащего текст сообщения и единственную кнопку "Ok".

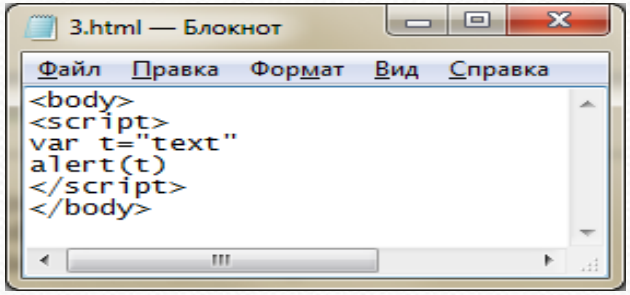
Программа выводит сообщение и ожидает нажатия кнопки. После нажатия на кнопку, программа начинает выполняться дальше.

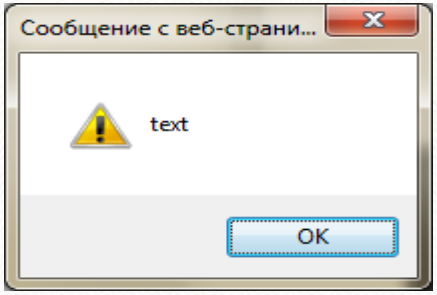
Текст сообщения может сцепляться с любой текстовой переменной с помощью знака «+».

Чтобы текст выводился в несколько строк используют символы «\n»

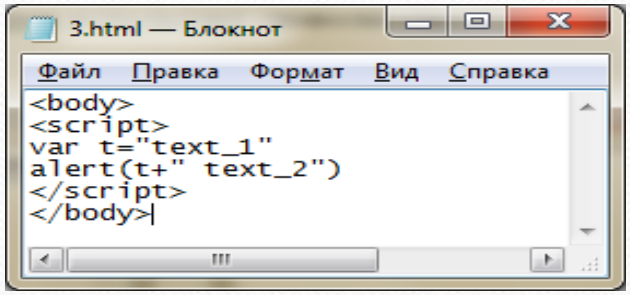
Примеры метода alert

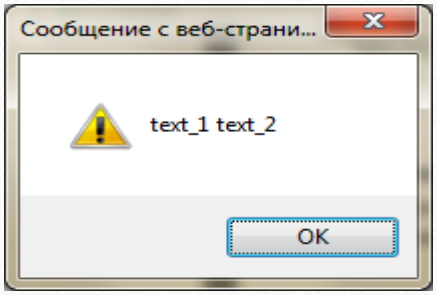
Пример 1



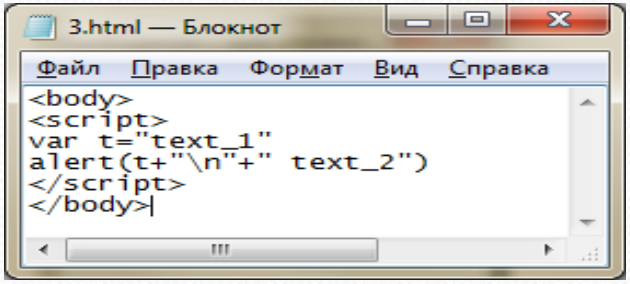


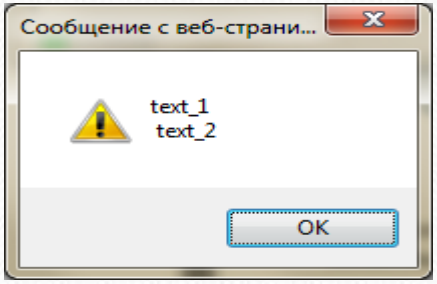
Пример 2





Пример 3





Метод 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.

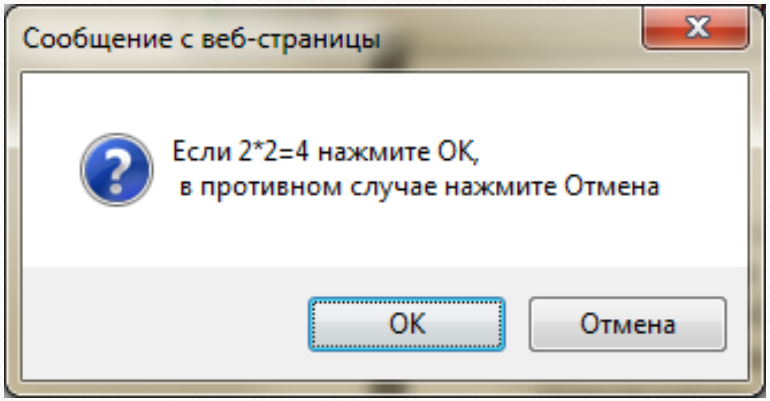
Как правило, результат работы функции присваивают логической переменной, для дальнейшего анализа, как это показано в примере.

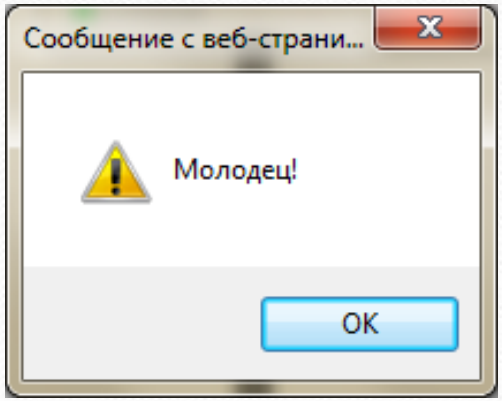
Пример метода confirm

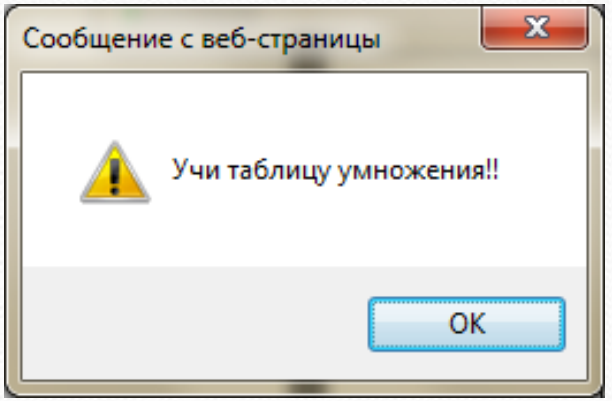
3.html — Блокнот

Файл Правка Формат Вид Справка

```
<body>
<script>
var result=confirm("Если 2*2=4 нажмите ОК,"+
"\n"+" в противном случае нажмите Отмена");

if(result) alert("Молодец!")
else alert("Учи таблицу умножения!!");
</script>
</body>
```







Метод prompt(“строка1”, “строка2”)

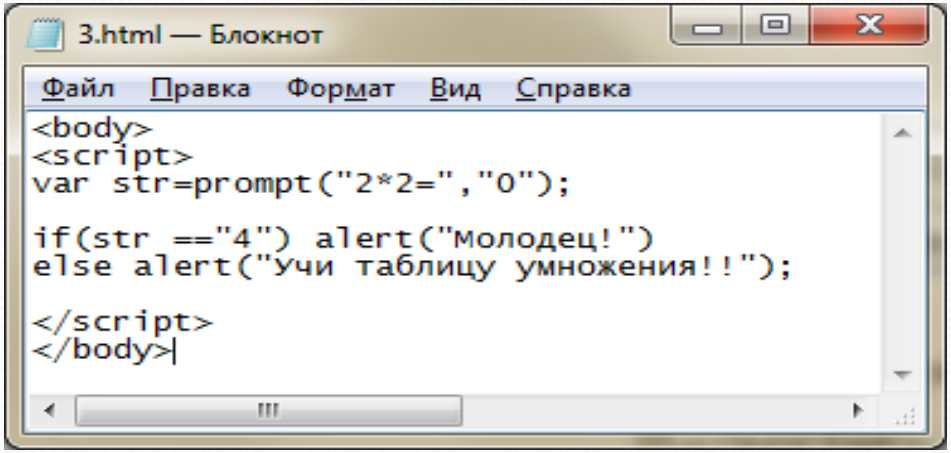
Метод prompt используется в тех случаях, когда пользователю нужно ввести значение в переменную.
В окно выводится сообщение «строка1», в поле ввода помещается умалчиваемое значение «строка2».

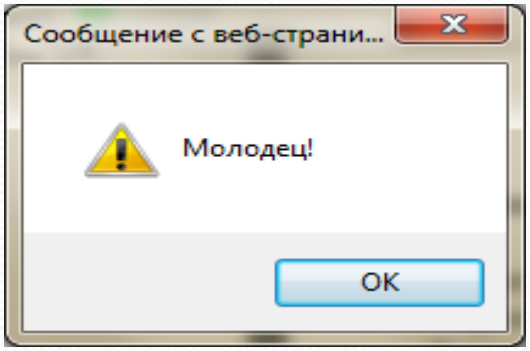
Результат работы функции присваивают переменной строкового типа.

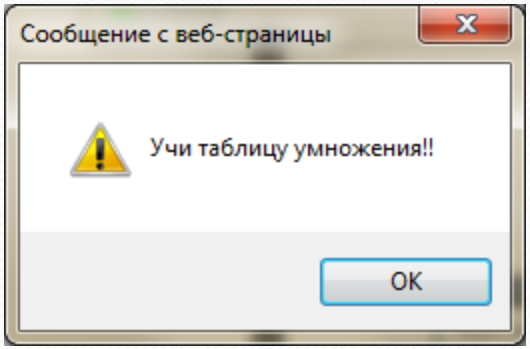
Если введенные данные нужно использовать в арифметических выражениях, необходимо выполнить преобразование введенной строки к числовому типу. Это можно сделать при помощи следующих функций:

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

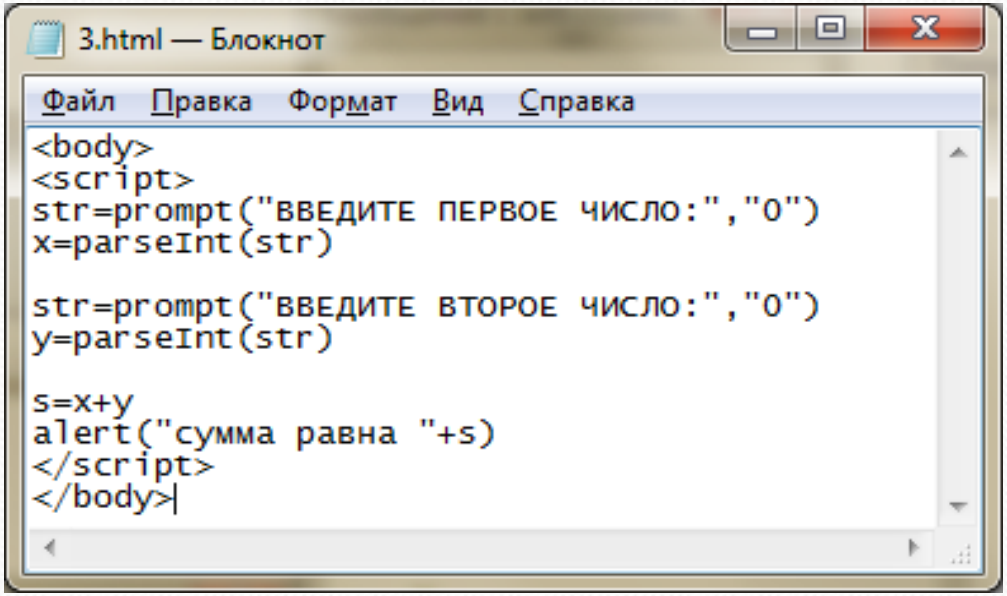
Пример 1 метода prompt







Пример 2 метода prompt



В данном скрипте переменные не описываются, что допускается.

Переменная str будет иметь строковый тип, так как результат функции prompt должен быть строкового типа.

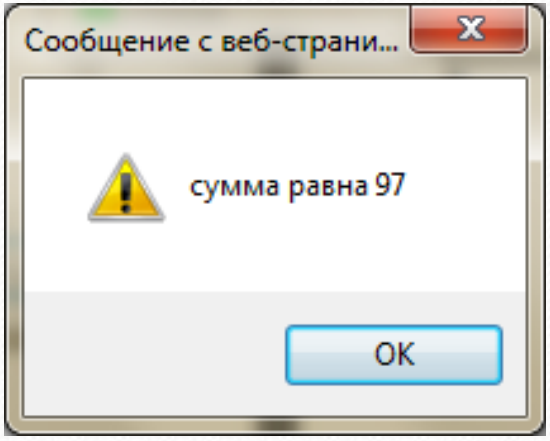
Функция parseInt преобразует переменную str, которая имеет строковый тип в переменную x числового типа.

Переменная s в операторе присваивания имеет числовой тип, так как переменные x и y имеют числовой тип.

Переменная s в функции alert автоматически преобразуется в строковый тип, так как параметр этой функции должен быть строкой.

Результат 1 примера 2

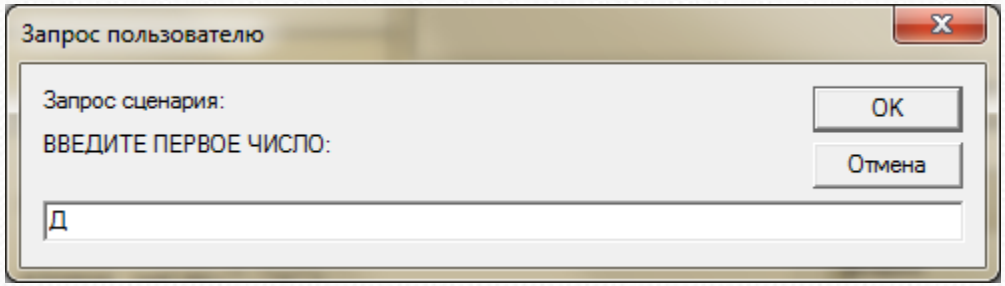
Введем два числа, найдем их сумму и выведем ее на экран.

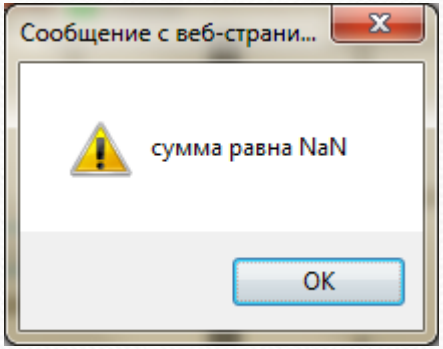


Результат 2 примера 2

Введем вместо чисел буквы. 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 (Not a Number)**.





Объекты в JavaScript

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».

Про JavaScript говорят, что в нем все объект.

А именно: объектами является окно, в котором открывается документ, сам документ, все элементы документа и даже свойства этих элементов. Для упорядочивания всех объектов используется **DOM** – (**D**ocument **O**bject **M**odel) объектная модель документа.

Объект document и его метод document.write

Объект document соответствует всему HTML-документу.

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".

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.

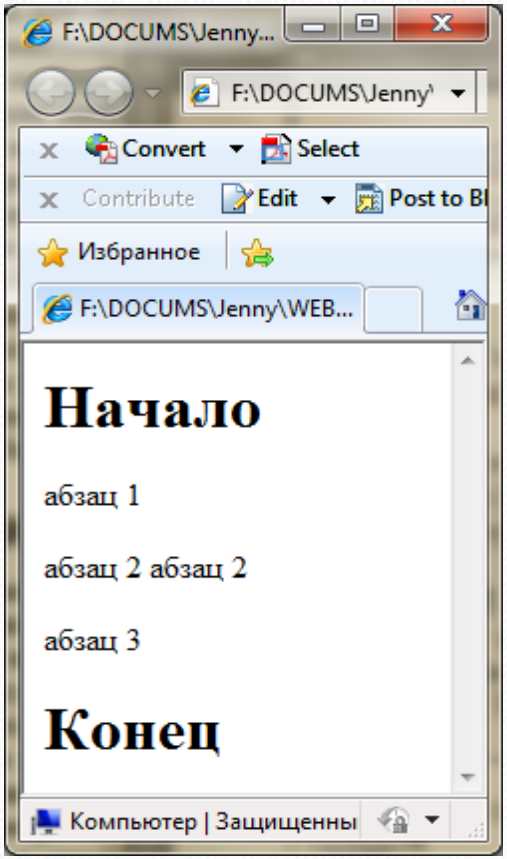
Пример 1

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

4.html — Блокнот

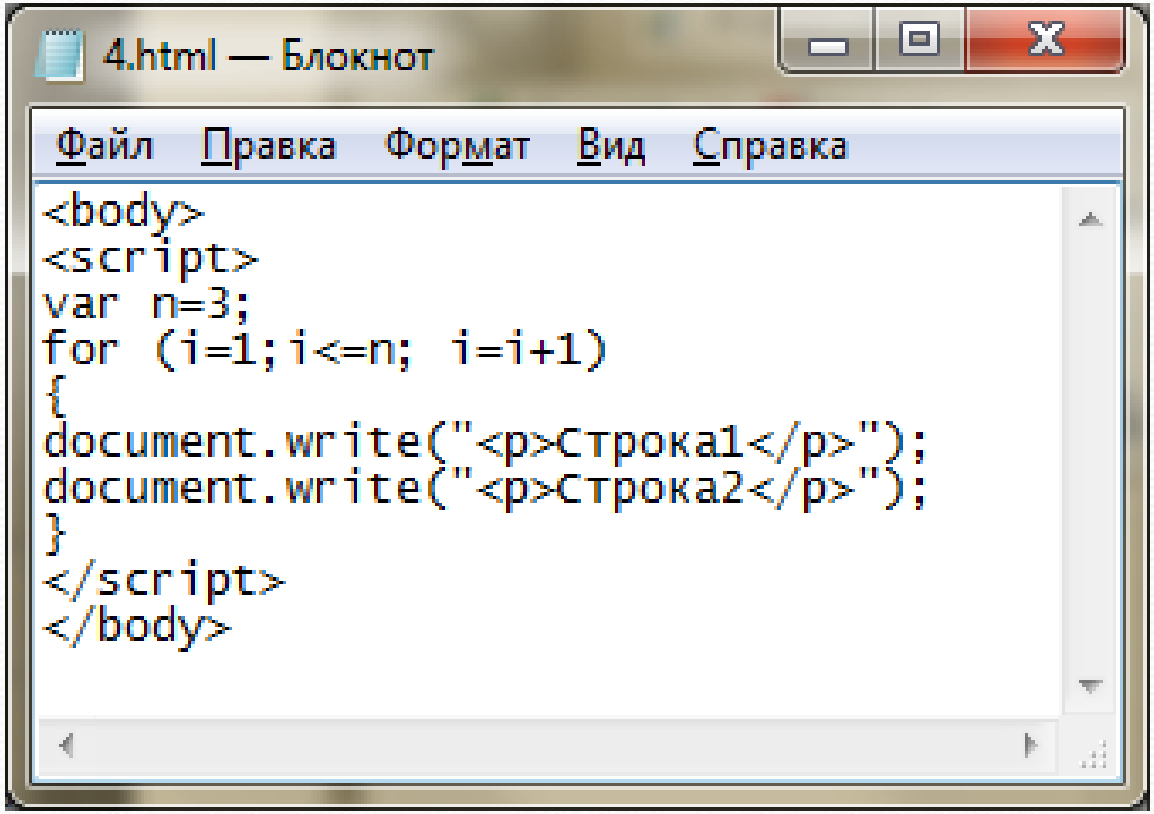
Файл Правка Формат Вид Справка

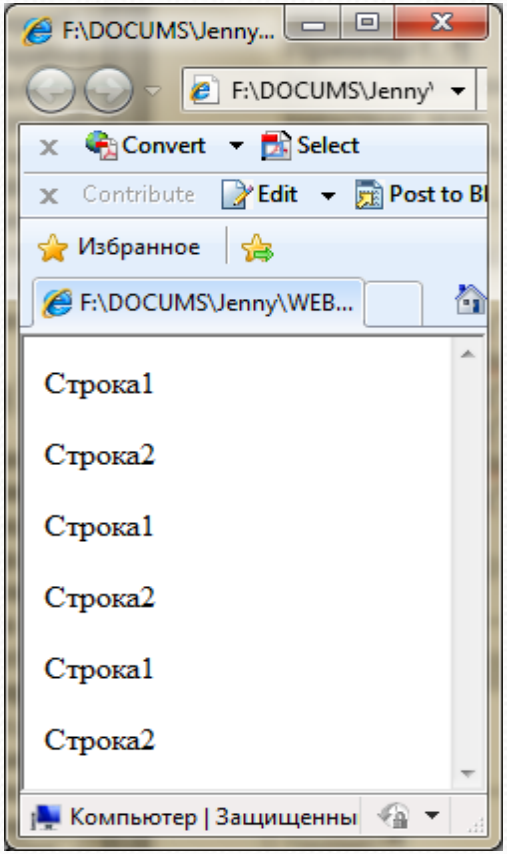
```
<body>
<h1>Начало</h1>
<script>
document.write("<p>абзац 1</p>");

document.writeln("<p> абзац 2");
document.write("абзац 2</p>");

document.write("<p> абзац 3</p>");
</script>
<h1>Конец</h1>
</body>
```



Пример 2

В данном примере весь документ формируется динамически.





JavaScript

② События и функции

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.

Категории событий

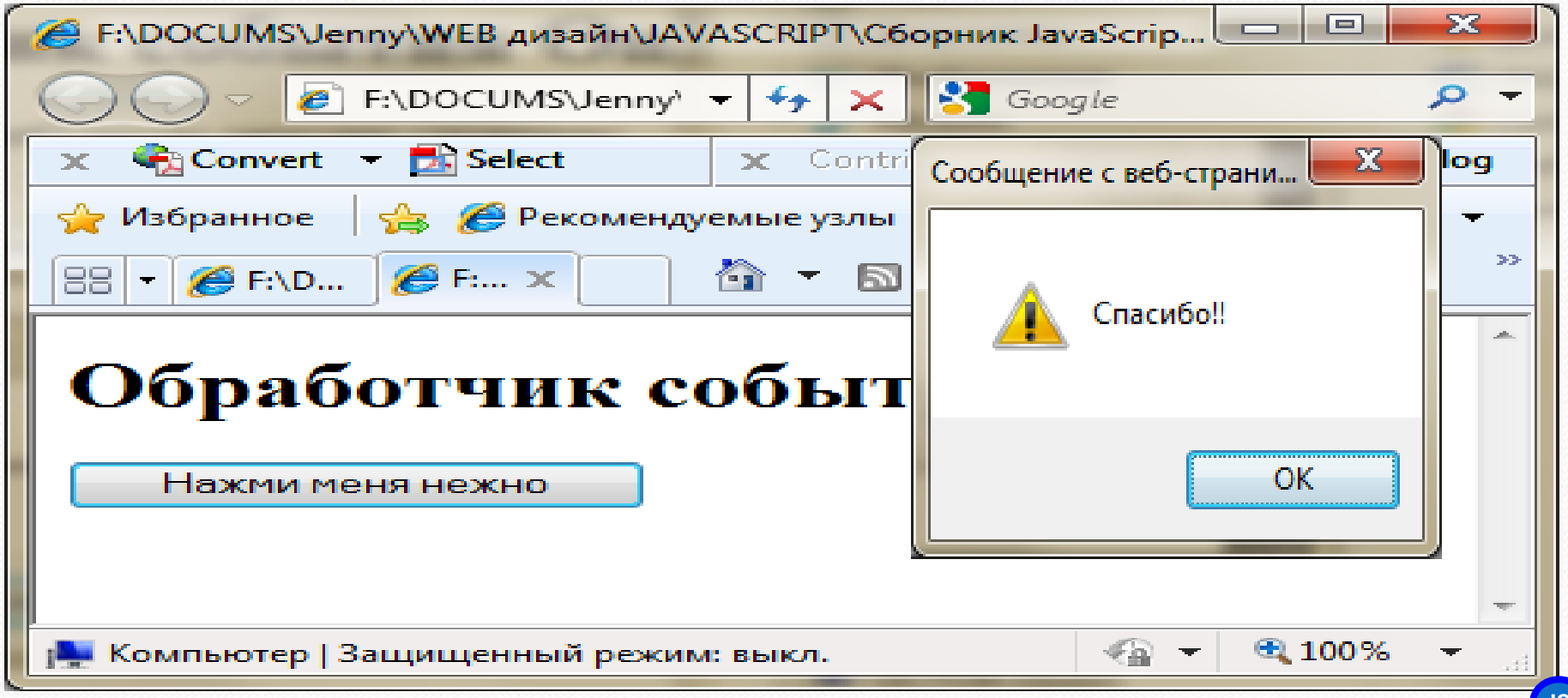
1. События, связанные с документом в целом.
2. События, связанные с элементами документа.
3. События, связанные с окнами.

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

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

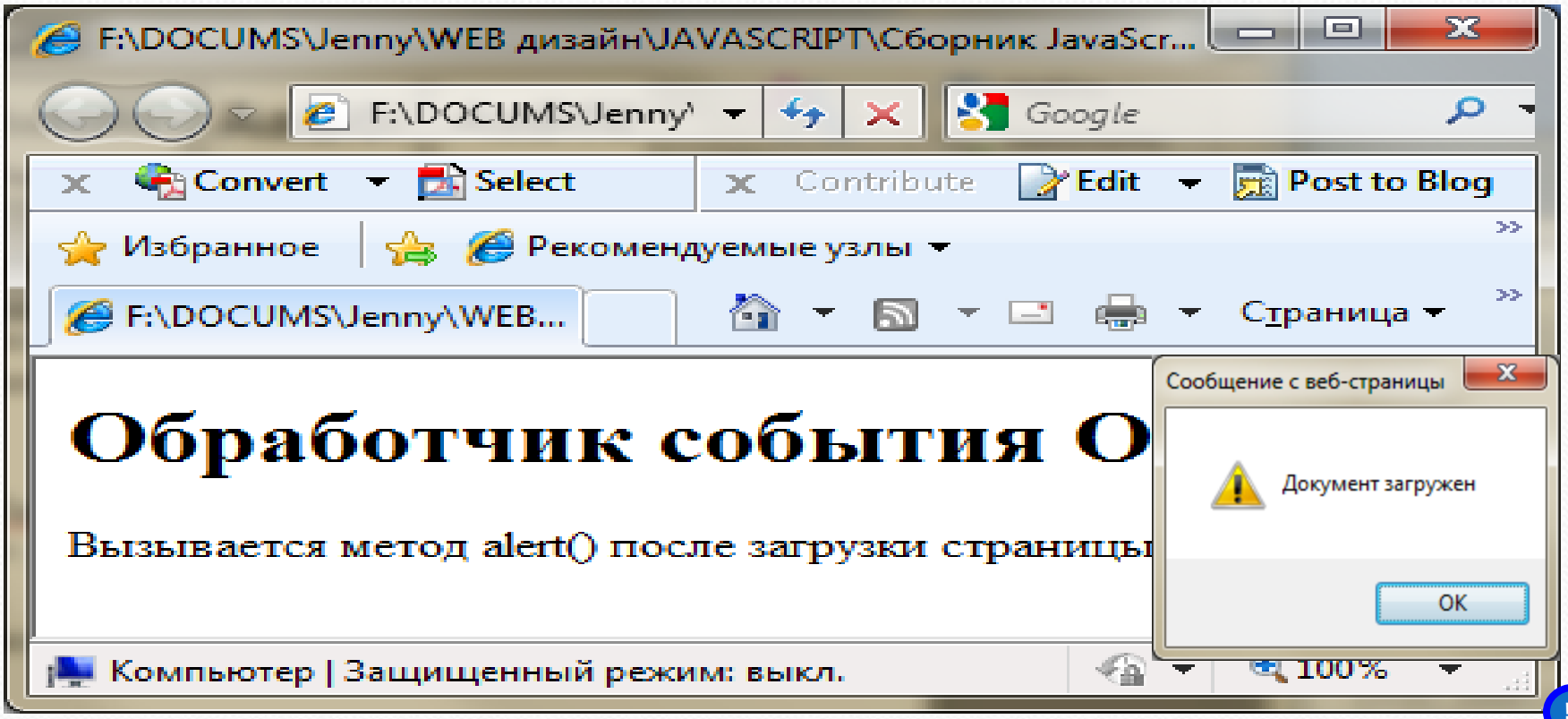
Обработчик onClick

Если нажать на кнопку происходит событие **Click**, по которому вызывается метод alert().



Обработчик onLoad

Событие **Load** возникает когда закончена загрузка документа. Например, в данном примере, после загрузки страницы вызывается метод alert().



Назначение обработчиков событий

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**:

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

Назначение обработчиков событий

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

```
function Имя_Функции (список формальных аргументов через запятые)
{
  ...
  операторы
  ...
  return значение;
}
```

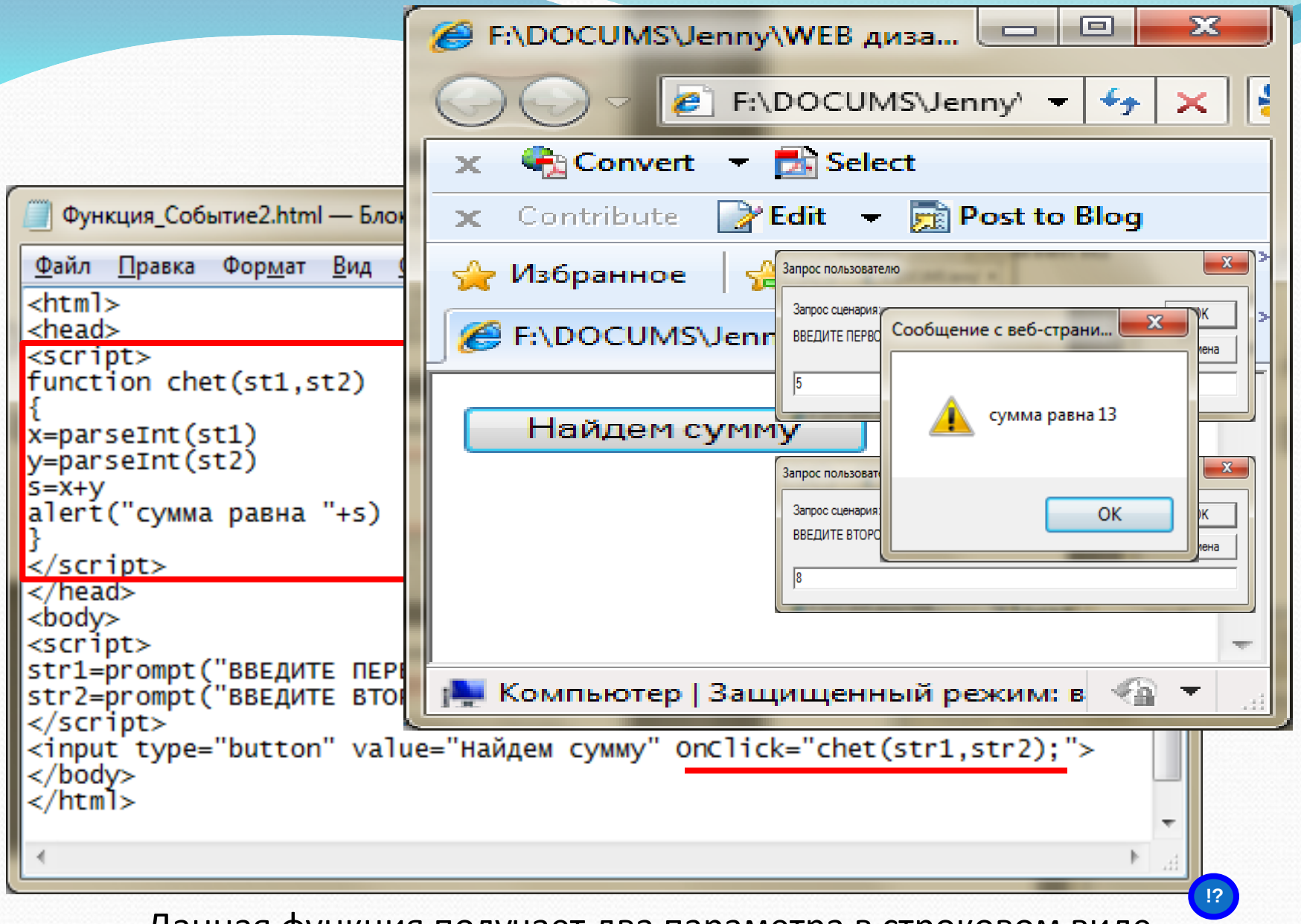
Команда **return**, возвращающая значение функции, может быть не одна, может и вовсе отсутствовать. В последнем случае функция не возвращает никакого значения и ее вызов нельзя использовать в выражениях. Если в функцию или из нее не передаются параметры - то после имени функции ставятся круглые скобки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

Вызов функции

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.



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

JavaScript

③ Встроенные объекты

Классификация объектов

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

Все объекты, которые используются в языке **JavaScript,** делятся на несколько групп.

Познакомимся с:

- некоторыми встроенными объектами базового языка;
- некоторыми объектами 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.

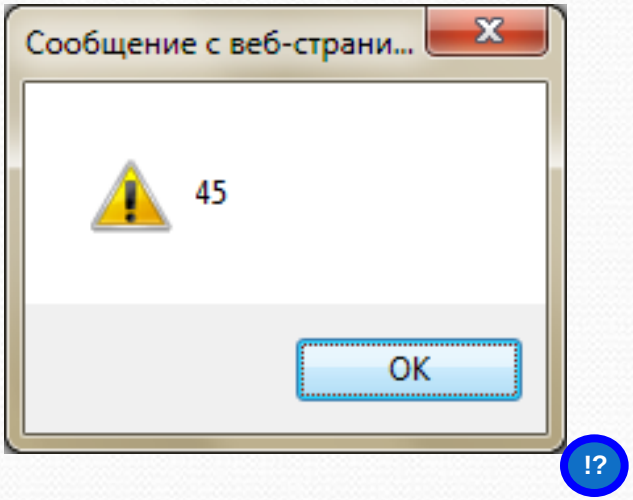
В таблице приведён список некоторых из его методов.

Методы объекта Math

Метод	Описание
`abs(число)`	Возвращает модуль (абсолютную величину) числа.
`sqrt(число)`	Возвращает квадратный корень из числа.
`pow(основа, степень)`	Возвращает результат возведения основания в указанную степень. Например, Math.pow(5, 3) вернёт 5^3=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.

Пример

```
<script>
d1=45.9;
d2=Math.floor(d1);
alert(d2);
</script>
```



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)**.

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 86 400 000 мсек.

Объект и экземпляр объекта

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

Например, **объектом** является **комплект документации** на заводе, по которой изготовляются компьютеры.

Сам **компьютер** является **экземпляром** этого объекта.

Все компьютеры, которые сходят с конвейера, имеют одни и те же свойства и одни и те же методы управления этими свойствами.

Создание экземпляра объекта

Для создания экземпляра объекта используется конструктор **new**.

Объект **Math** статический, то есть он существует в единственном экземпляре. Поэтому для объекта **Math** экземпляр не создавался.

Для объекта **Date** можно создавать экземпляры, которые содержат текущие дату и время, а можно создавать экземпляры с указанной датой и временем.

Создание экземпляра объекта Date с текущей датой и временем

```
var a = new Date();
```

В переменной «**a**» текущая системная дата и системное время.

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

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

• год — числовое выражение, задающее полный номер года;
• 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.

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

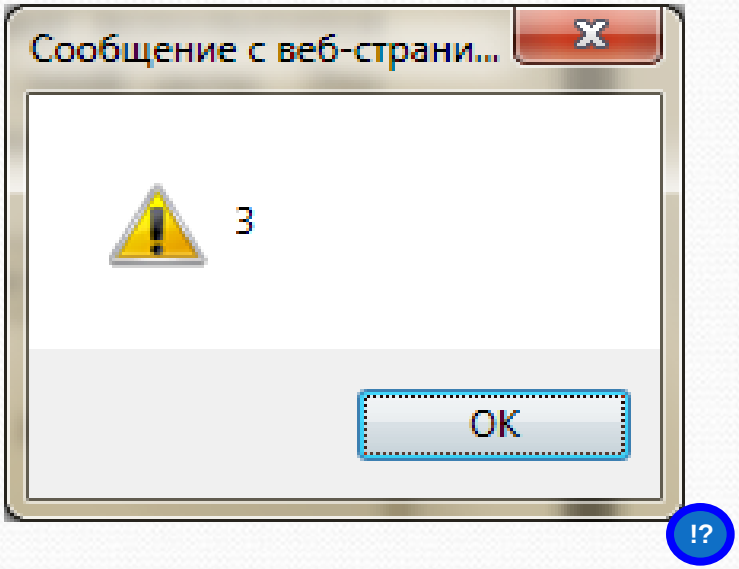
Методы объекта Date

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.

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

```
<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)**.

Объект Array

Объект **Array** предназначен для хранения массивов данных.

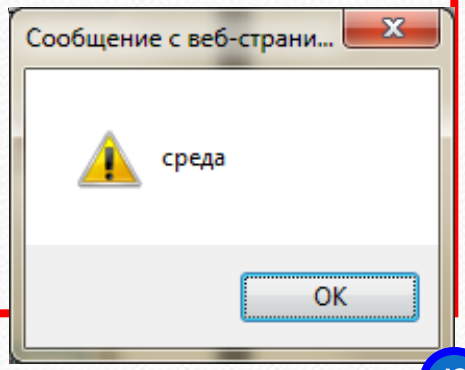
Массив - это упорядоченный набор элементов.

Доступ к отдельному элементу производится по имени массива и индексу (номеру).

Нумерация элементов массива в JavaScript всегда начинается с нуля.

Пример

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



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**.

http://www.hunlock.com/blogs/mastering_javascript_arrays#concat

Именование объектов документа

Каждый из элементов на Web–странице является объектом. Необходимо уметь обращаться к этим объектам. Есть несколько способов.

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**.

Ссылка на объект документа

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")
```

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='Name?'>
```

Ссылка на 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 описать переменную **ob** типа объект и присвоить свойству **value** этого объекта новое значение **'Введите имя'**.

```
var ob=document.getElementById("entry")
ob.value='Введите имя';
```

JavaScript

④ Объект Window

Объект Window

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

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);
```

- 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.

Открыть новое окно и закрыть его

```
var переменная = open(файл, имя_окна);
var переменная = open(файл, имя_окна, параметры_окна);
переменная.close();
```

имя_окна- строка, имя окна (значение свойства "id" объекта **window**).
параметры_окна - строка, имя параметра (значение свойства "id" объекта **параметр**).

Параметры окна задаются в строке параметры_окна в виде:

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

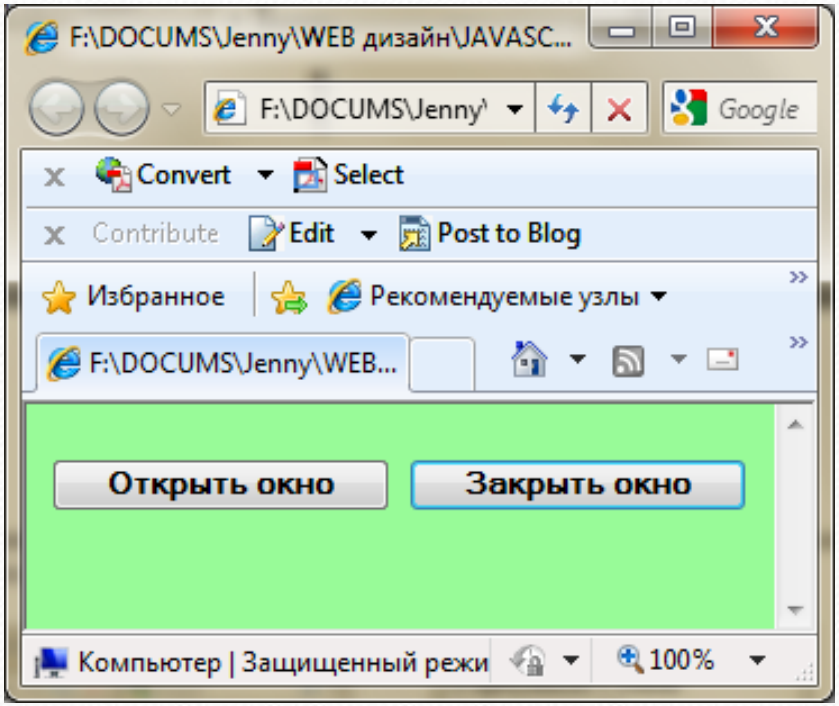
Замечание
Не рекомендуется оставлять пробел после разделительной запятой.

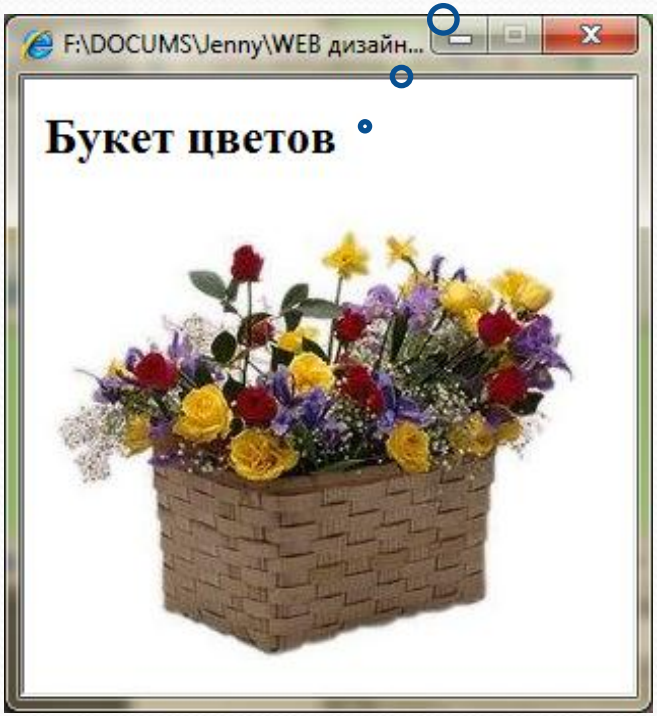
Описание некоторых параметров окна

Параметр	Описание
`width`	Ширина окна в пикселах. Минимальное значение - 100.
`height`	Высота окна в пикселах. Минимальное значение - 100.
`left`	Задает горизонтальную координату левого верхнего угла окна в пикселах.
`top`	Задает вертикальную координату левого верхнего угла окна в пикселах.

Пример

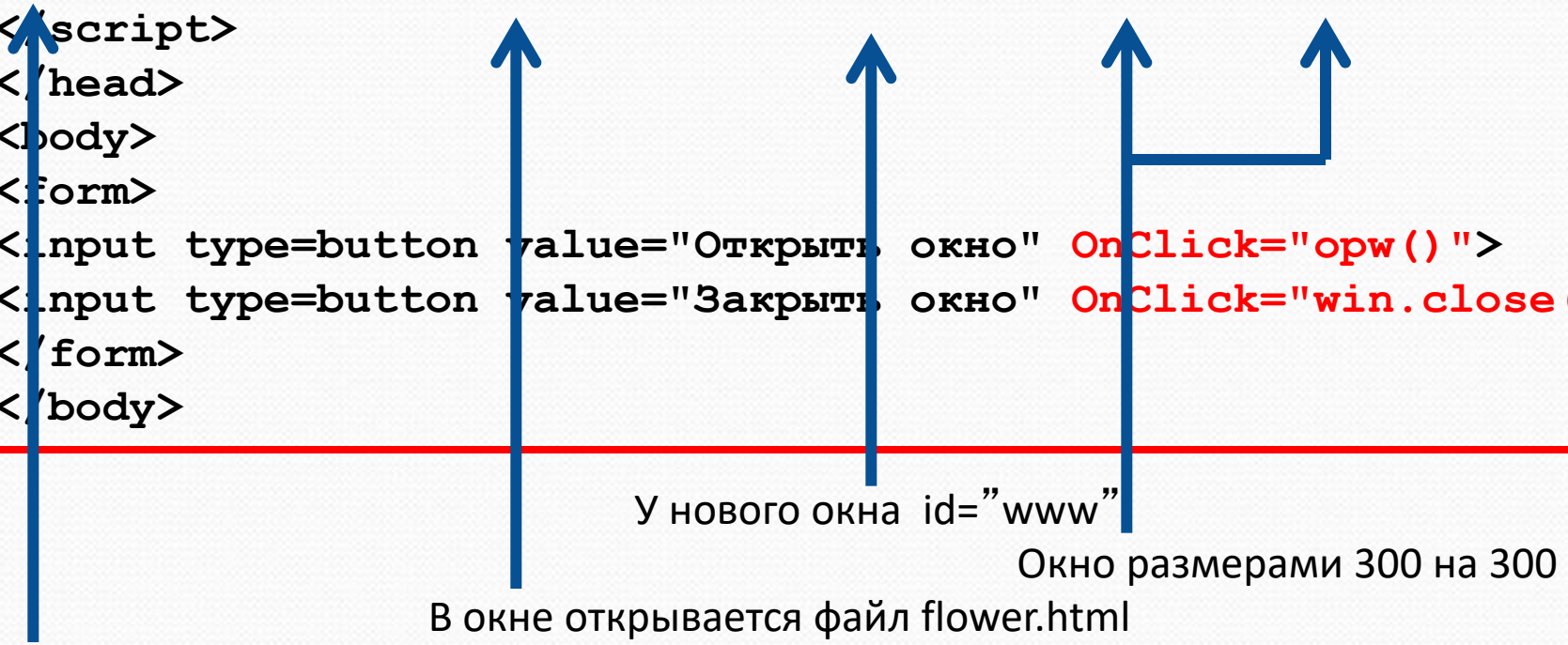
Написать приложение, которое по нажатию кнопки создает новое окно, в которое выводится существующий html-документ (flowers.html), а по нажатию другой кнопки, закрывает созданное окно.





Решение примера

```
<head>
<script>
function opw()
{win=window.open("flower.html","www","width=300,height=300");}
</script>
</head>
<body>
<form>
<input type=button value="Открыть окно" OnClick="opw()">
<input type=button value="Закрыть окно" OnClick="win.close()">
</form>
</body>
```



JavaScript

⑤ Обращение к элементам формы

Разновидности 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.

Свойство checked объекта input

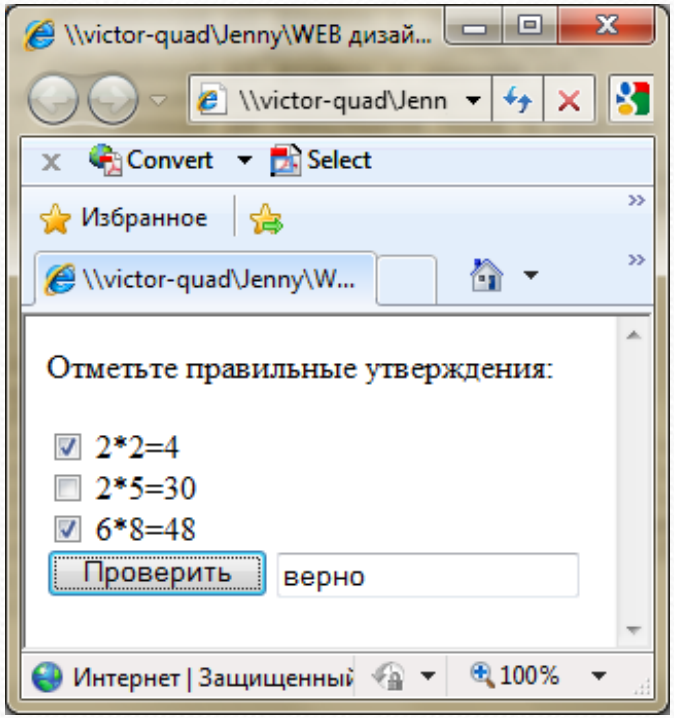
Флажки и радиокнопки разновидности объекта **input**. У объекта **input** есть свойство **checked**.

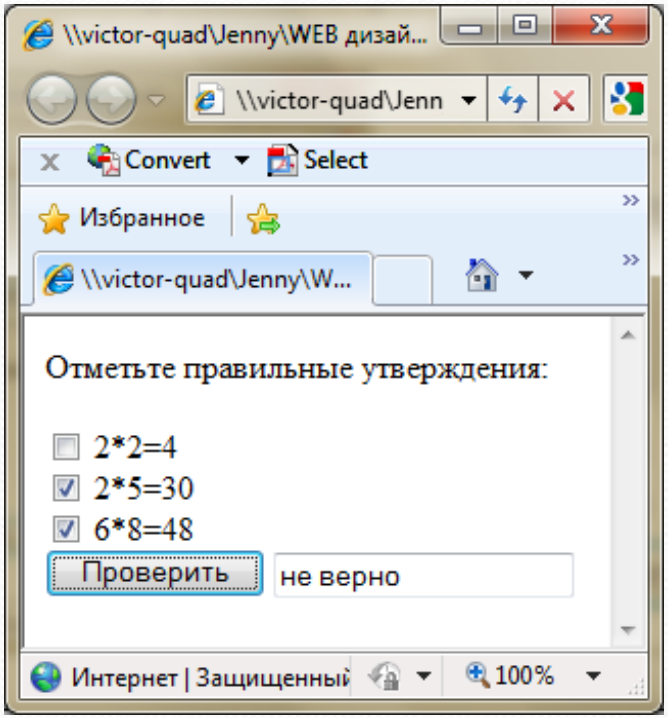
Это свойство имеет значение **true**, когда флажок установлен (радиокнопка выбрана), и **false** - в противном случае.

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

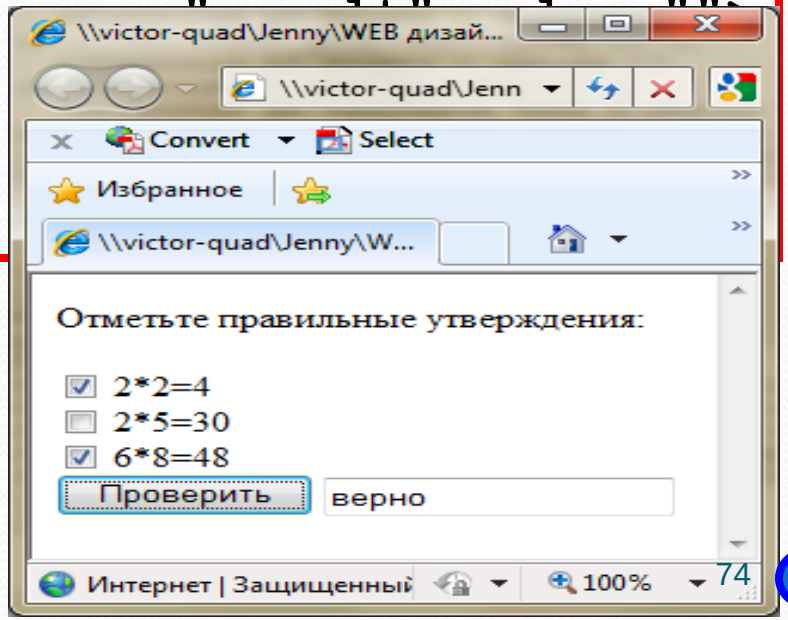
Написать приложение для проверки таблицы умножения. Правильные примеры необходимо отметить флажками. Если ошибок нет, в поле формы вывести слово «верно», в противном случае «неверно».





Тело документа

```
<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"
</form>
</body>
```

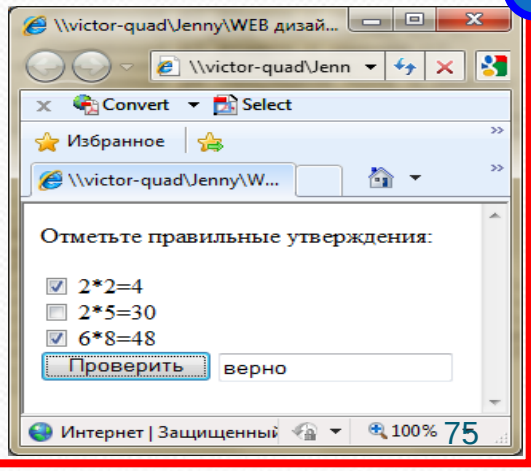


Флажок с именем c1
Флажок с именем c2
Флажок с именем c3
Текстовое поле с именем result

Функция

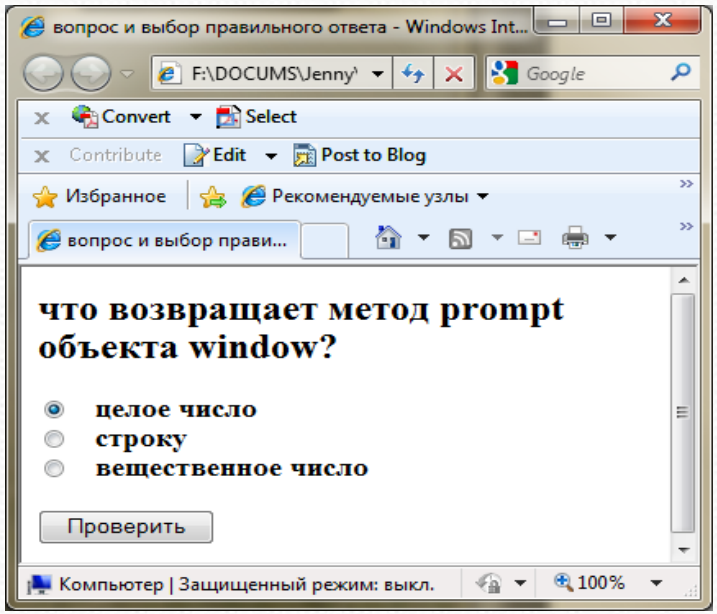
```
<head>
<script>
function checkAll()
{
  var ans='не верно';
  var p1=document.getElementById('c1');
  var p2=document.getElementById('c2');
  var p3=document.getElementById('c3');
  var p4=document.getElementById('result');
  // если флажок c1 поднят, флажок c2 не поднят и флажок c3 поднят
  if (p1.checked && !p2.checked && p3.checked)
  ans ='верно';
  else ans = `неверно';
  // вывод результата в текстовое поле
  p4.value=ans;
}
</script>
</head>
```

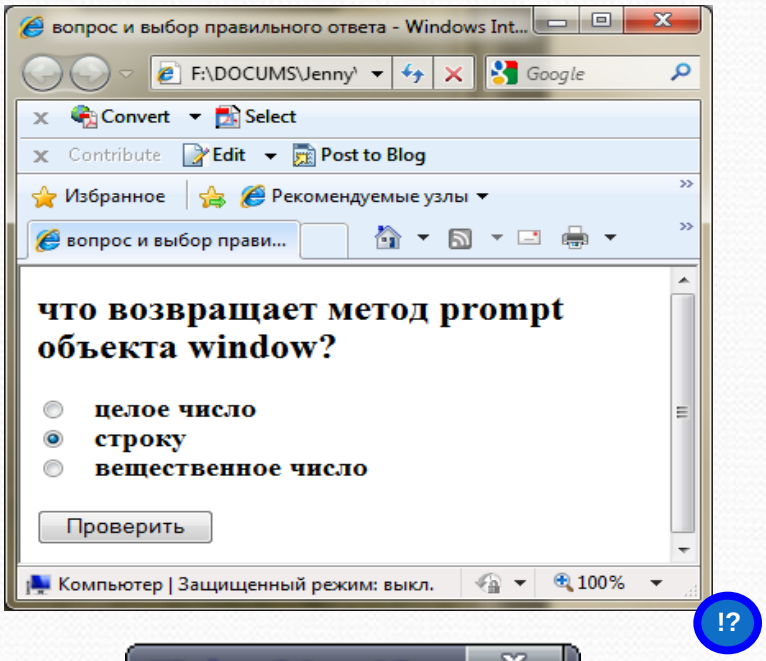
p1 флажок с именем c1
p2 флажок с именем c2
p3 флажок с именем c3
p4 текстовое поле с именем result

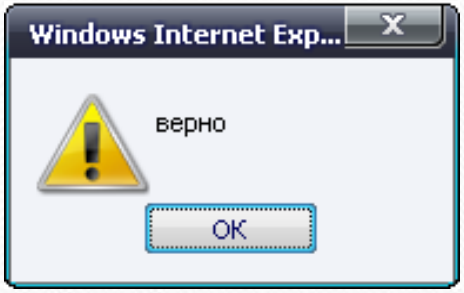


Пример

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





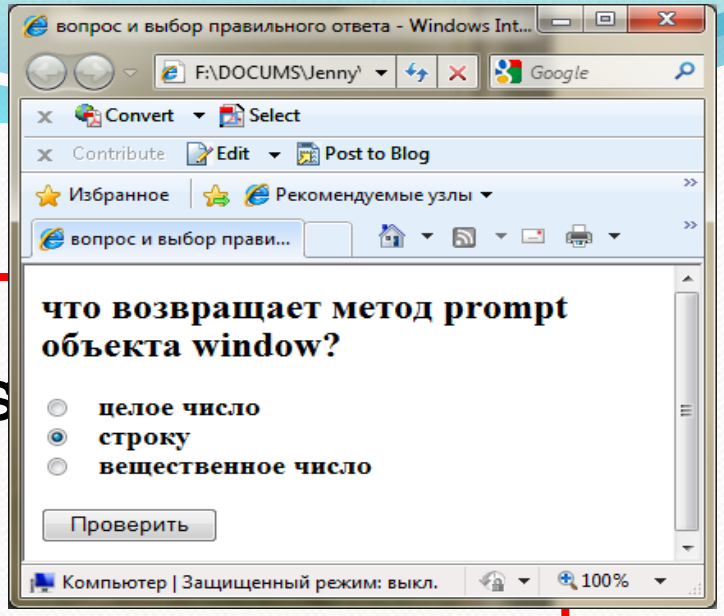


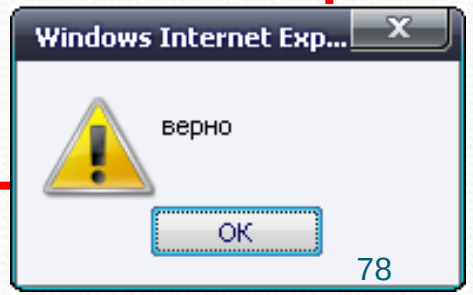
Тело документа

```
<body>
<h2>что возвращает  метод prompt объекта window? </h2>
<form>
<h3>
<input type='radio' id='k1' name='v'/>  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>
```

Функция

```
<head>
<script type="text/JavaS
// получить значение из
function validate()
{
// переменная p1 радиокнопка с id k2
var p1=document.getElementById('k2');
// если выбрана вторая радиокнопка
if (p1.checked) alert("верно");
else alert("неверно");
}
</script>
</head>
```





Свойства объекта option

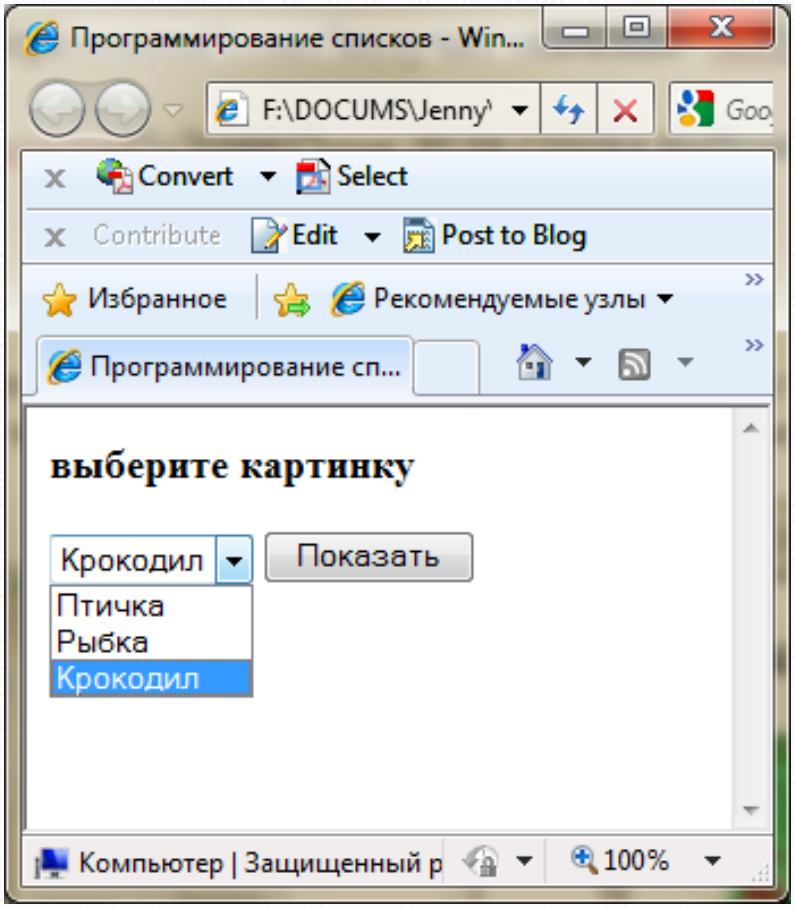
Для создания раскрывающегося списка используется элемент **<select>,** который включает в себя несколько элементов **<option>,** по одному элементу на каждую строку списка.

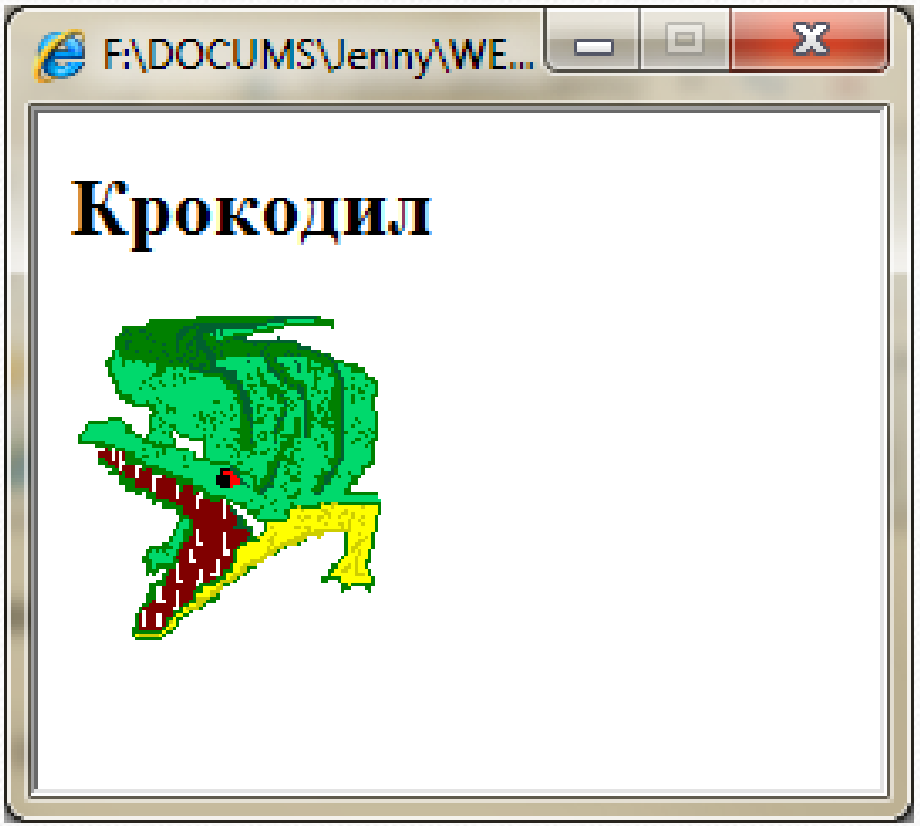
Каждый из этих объектов **option** имеет следующие свойства:

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

Пример

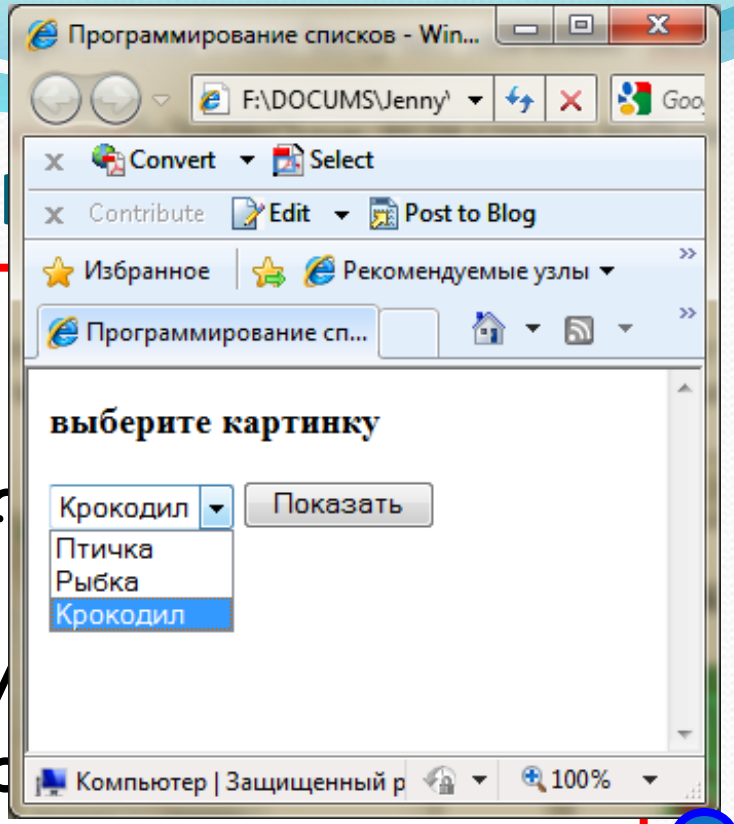
Написать приложение, которое выбирает картинку из списка и при нажатии на кнопку выводит в новое окно выбранное название в виде заголовка и саму картинку.



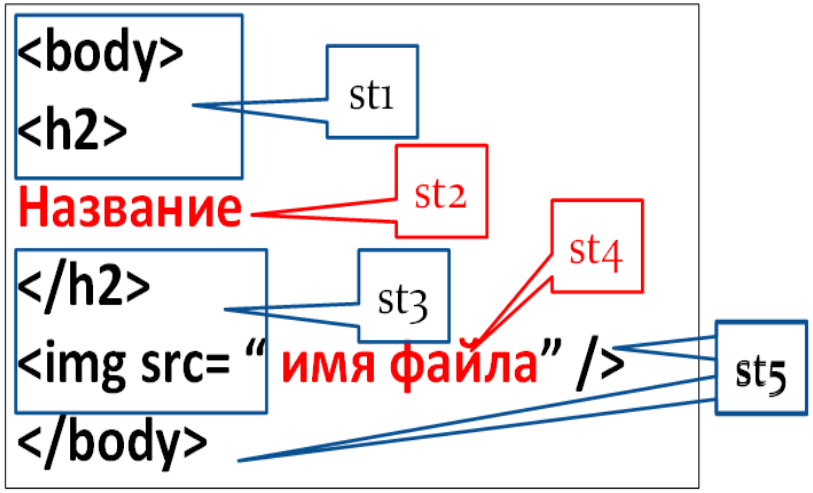


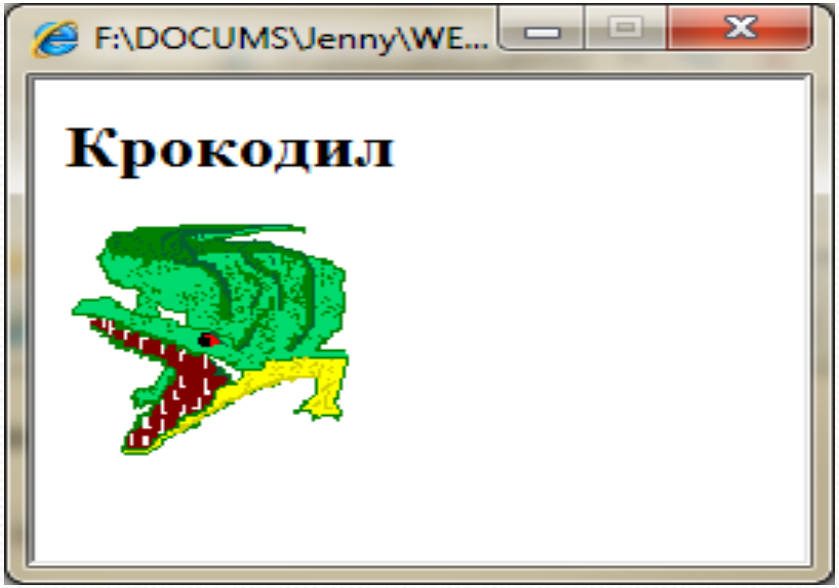
Тело докуме

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



Функция






```
<head>
<script type="text/JavaScript">
function GO()
{
var p1=document.getElementById('k1');
var p2=document.getElementById('k2');
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="'BIRD.GIF'" }
if (p2.selected)
{ st2=p2.text; st4="'FISH.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>
```

JavaScript

⑥ Объект Image

Свойство объекта 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 будем изменять свойство src в зависимости от того, какое наступает событие MouseOver или MouseOut.

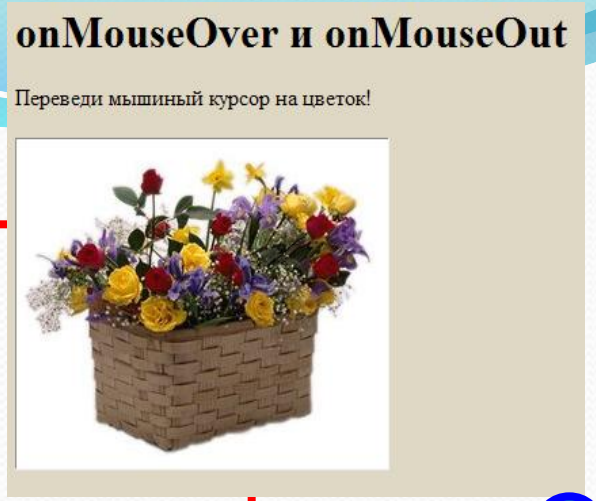
Пример

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

Для обработки событий создадим функцию **doEvent** с логическим параметром.

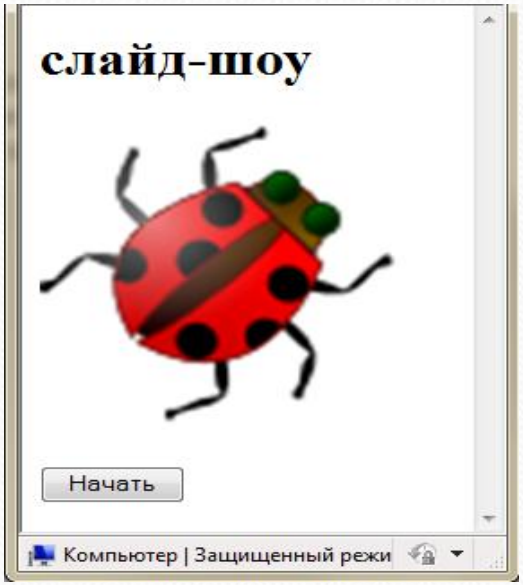
Пример

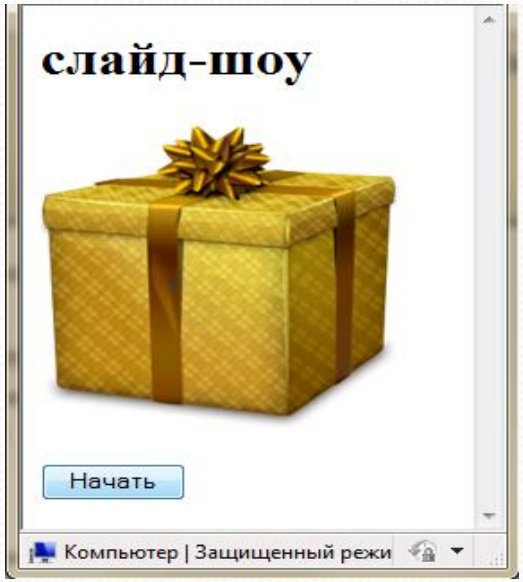
```
<head>
<script>
function doEvent(type)
{
var cv=document.getElementById('pic');
if(type) cv.src="flower.jpg";
else cv.src="bigdaisy.jpg";
}
</script>
</head>
<body>
<h1>onMouseOver и onMouseOut</H1>
<p>Переведи мышиный курсор на цветок!</p>
<img name ="pic" id="pic"  src="bigdaisy.jpg"
onMouseOver="doEvent(true);"
onMouseOut="doEvent(false);">
</body>
```

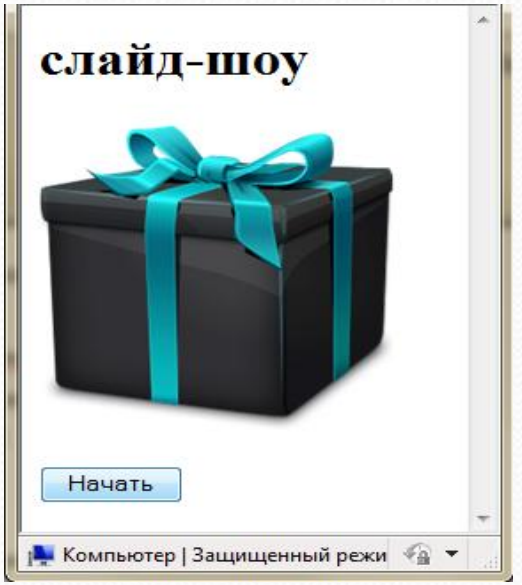


Функция slideShow()

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



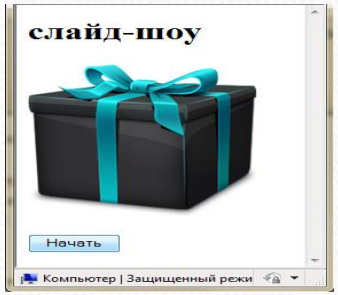




Суть в том, чтобы через каждые несколько секунд у объекта **slide** менялось свойство **src** на значение из массива картинок.

Пример реализации функции

```
<head>
<style> img{width:200;height:200} </style>
<script>
i=0;  // номер картинки в массиве
function slideShow()
 {
  ris = new Array(`12.png', '8.png', '9.png', '6.png')
  if (i >= 4)   i = 0
  r=document.getElementById('slide')
  r.src=ris[i]
  i++
  // вызываем эту же функцию через каждые 1500 мс
  setTimeout("slideShow()",1500)
 }
</script>
<body>
<h1> слайд-шоу </h1>
<img id='slide'  src=«9.png" />
<form>
<input type="button" value=" Начать " onClick="slideShow();">
</form>
</body>
```



JavaScript

⑦ Объект Style

Свойства объекта Style

Свойства объекта **style** позволяют 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**.

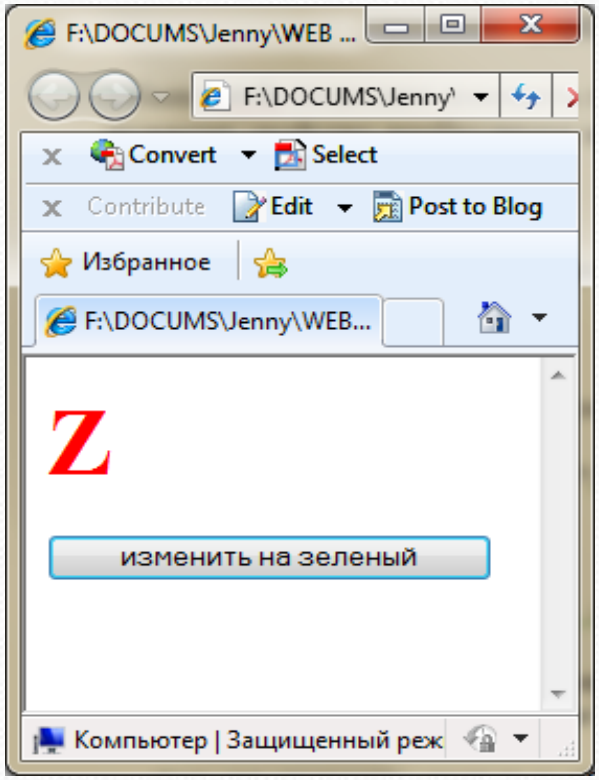
Соответствие между названиями свойств в css и в Javascript

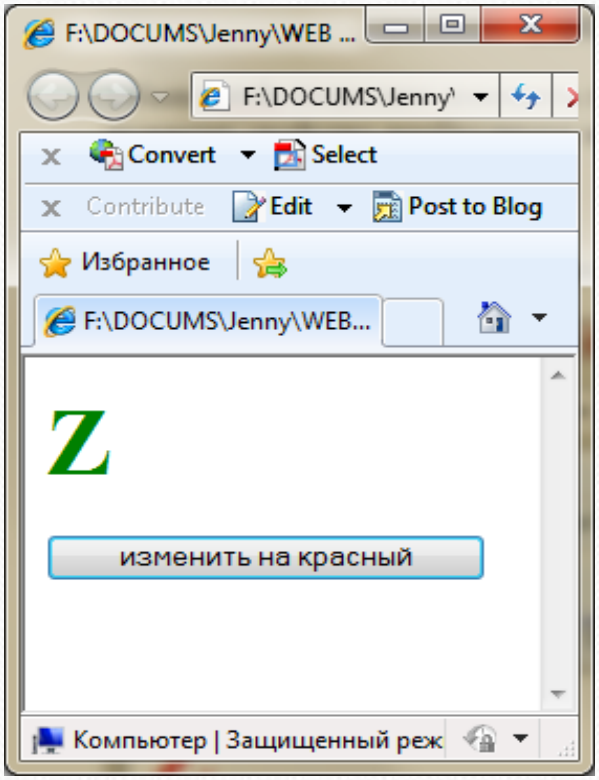
Названия свойств объекта **style** в **JavaScript** почти такие же что и в **CSS** за тем исключением, что символы "-" убираются, а первые буквы всех слов, образующих имя атрибута, кроме первого, делаются прописными.

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

Пример

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

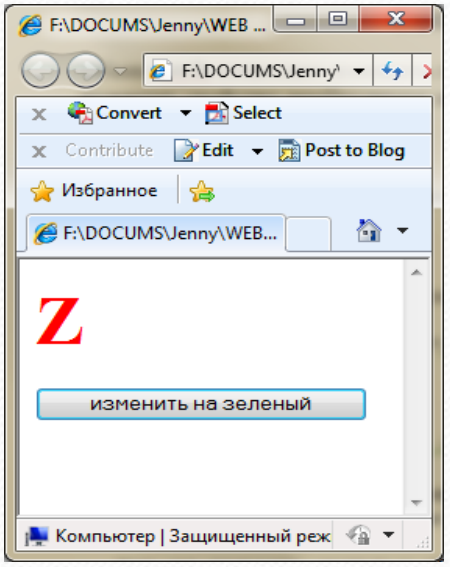




Тело документа

Объект буква

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

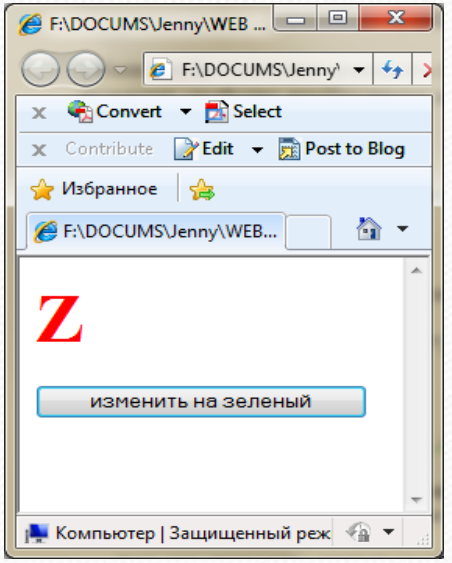


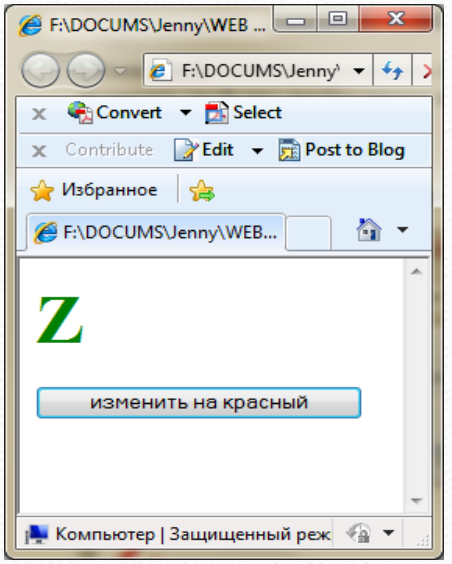
Объект кнопка

При нажатии на кнопку вызывается функция f(), которая будет менять свойство color у буквы и свойство value у кнопки.

Функция

```
<head>
<style> h1 {color:red; font-size:36pt;} </style>
</head>
<script>
i=0;
function f()
{
r1=document.getElementById('e1') // объект буква Z
r2=document.getElementById('e2') // объект кнопка

// поменяем у объекта r2 свойство value (надпись)
if (i==0)
{
i=1;    r1.style.color='green';
r2.value='изменить на красный';
 }
else
{
i=0;    r1.style.color='red';
r2.value='изменить на зеленый';
}
}  // конец функции
</script>
</head>
```





Свойства display

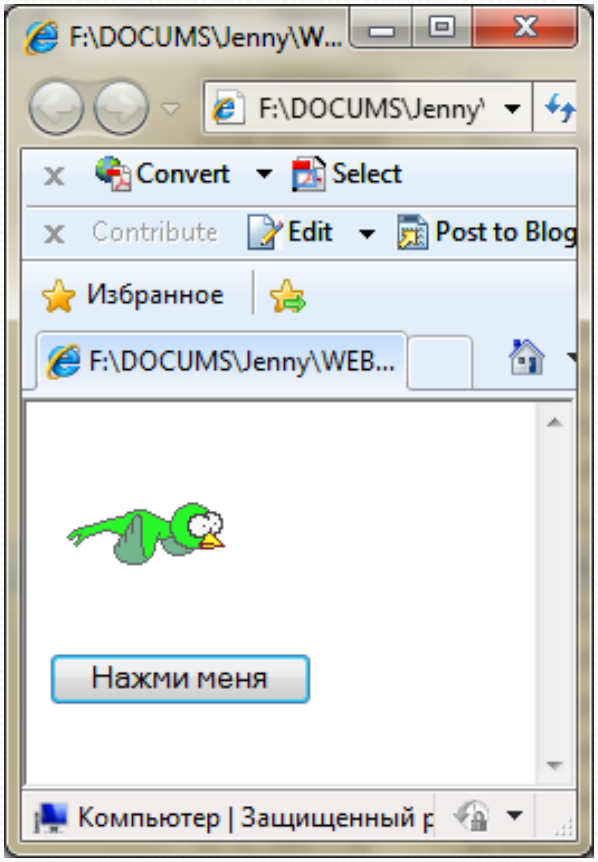
Свойство **display** задает способ отображения элемента на странице.

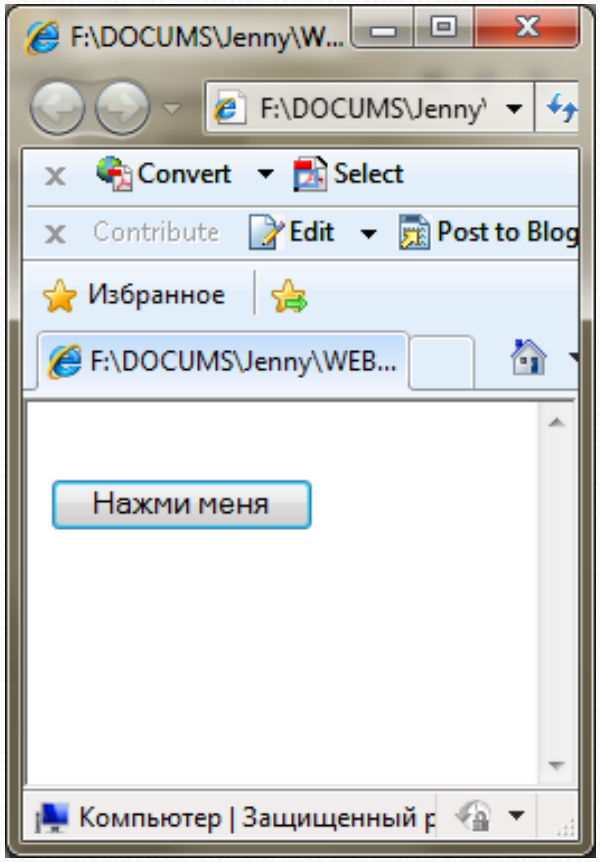
Некоторые значения этого свойства:

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

Пример

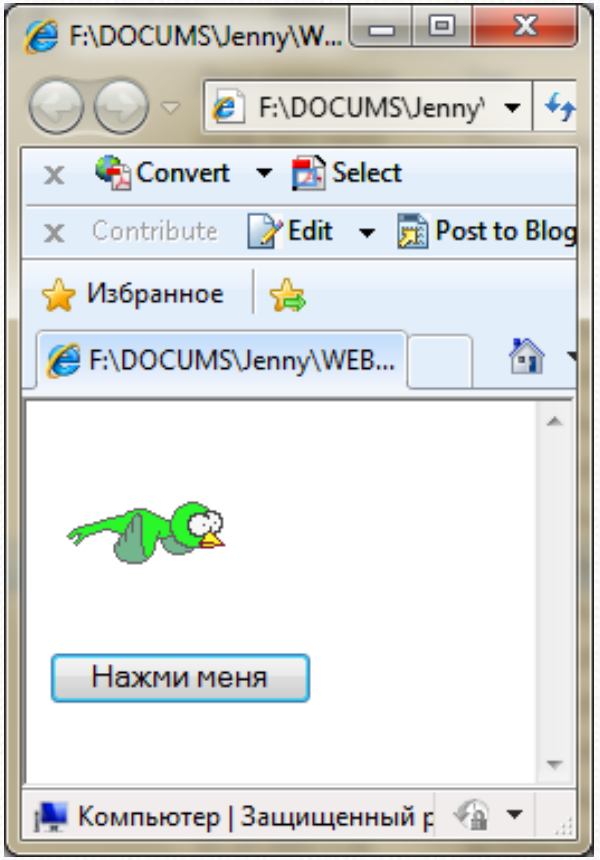
По нажатию кнопки картинка будет появляться и исчезать. Для этого будем пользоваться свойством **display** объекта **style**.





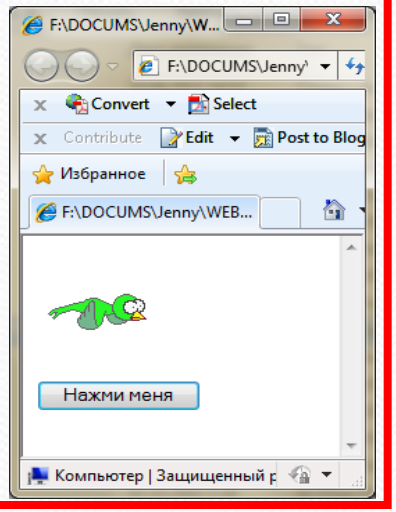
Тело документа

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



Функция

```
<head>
<script>
function displ()
{
// p это объект картинка
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>
```



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.

Пример

Создадим документ, в котором рисунок vallay.jpg занимает 100% окна и является нижним слоем. На верхнем слое второй рисунок bird.png, который двигается вправо. Когда второй рисунок достигает края первого рисунка, он останавливается.

Реализация примера

```
<html>
<head> <style> #im2 {position:absolute;left:0;top:100} </style>
</head>
<body>
<img src="vallay.jpg" id="im1" width="100%" />
<img src="bird.png" id="im2" />
<script type="text/JavaScript">
function f()
{
x = x + 1              // сдвиг рисунка на 1 пиксел
r2.style.left = x      // изменяем свойство left 2-го рисунка
// вызываем каждые 1000 мили сек эту же функцию
if (x < x1 - x2) setTimeout("f()",10)
}
x = 0
r1 = document.getElementById('im1')
r2 = document.getElementById('im2')
x1 = r1.width // ширина фона
x2 = r2.width // ширина птички
f();
</script>
</body>
</html>
```

ПРИМЕР

Спасибо за внимание!