Тема 1.6 Функции и события

Функции

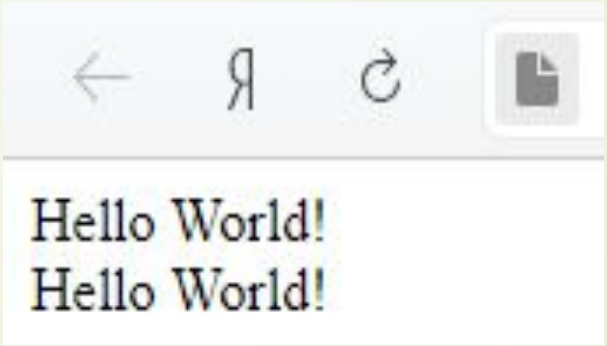
– это именованная последовательность действий (инструкций).

```
function Имя_Функции() {
оператор;
.....
оператор;
}
```

Пример: необходимо вывести текстовое сообщение Hello World! несколько раз

```
<script>
document.write(“Hello World!”);
document.write(“Hello World!”);
</script>
```

Мы можем оформить этот скрипт через функцию:

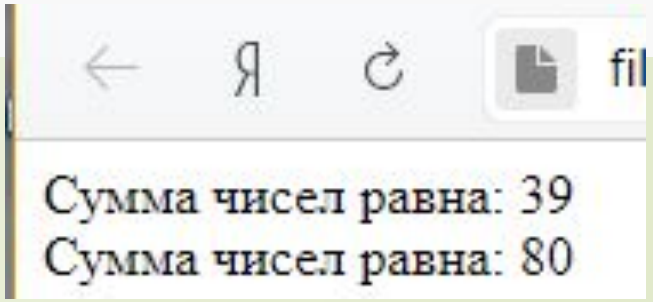



```
<script>
function print() {
   document.write(“Hello World! <br>”);
  }
    print();// вывод сообщения 1 раз
    print();// вывод сообщения 2-ой раз
</script>
```

Параметры функции:

Напишем функцию, которая будет суммировать 2 числа и выводить результат на экран.

```
<script>
    function summ(a,b){
        var itog = a + b;
        document.write("Сумма чисел равна: " + itog + "<br>");
    }
 summ(25, 14);
 summ(13,67)
</script>
```

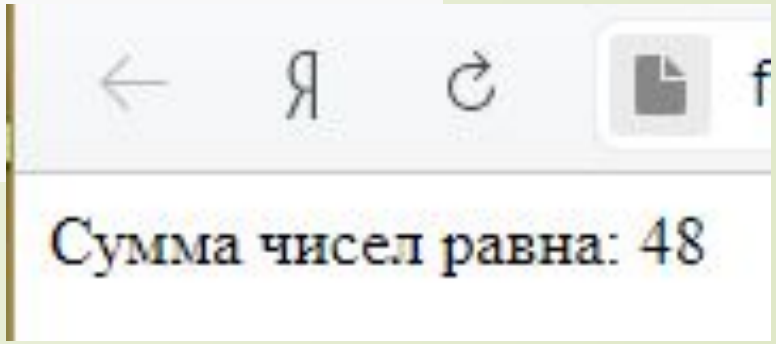


Параметры функции:

В качестве параметров функций могут выступать не только числа, но и какие – либо переменные.

Например, у вас птичий двор. Надо узнать количество птиц.

```
<script>
    function summ(a,b){
        var itog = a + b;
        document.write("Сумма чисел равна: " + itog + "<br>");
    }
 var chicken = 32;
 var duck = 16;
 summ(chicken, duck);
</script>
```



Область видимости переменных

Переменные бывают глобальные и локальные.

Глобальные переменные создаются один раз (вне функции) и потом используются в коде программы там где это необходимо (в массивах, функциях и т.д.)

Локальные переменные создаются внутри функции и только внутри нее могут использоваться.

Пример:

```
<script type="text/javascript">
 function showMessage() {
        var message = 'Привет, я - Вася!'; // локальная переменная
        alert( message );
        }
  showMessage(); // 'Привет, я - Вася!'
  alert( message ); // Ничего не получим
</script>
```

Возврат значений функцией:

Во всех рассмотренных примерах результат работы функции был сразу выведен на экран.

В программировании, в основном, нет необходимости сразу выводить результат работы функции.

Функция может **возвращать** результаты своей работы, а программист **получать** его где это необходимо.

Для возврата значений функцией используется оператор **return**.

Пример: Создадим функцию, которая будет возвращать дискриминант квадратного уравнения по формуле – **b^2 – 4ac**.

```
 <script type="text/javascript">
function calcD(a, b, c) {
   return b*b - 4*a*c;
}

var test = calcD(-4, 2, 1);
alert(test); // 20
 </script>
```

1. Создаем функцию + возвращаем ее значение (результат работы) – оператор **return**;
2. Вызываем функцию в любом месте + заносим результат ее работы в переменную;
3. Используем переменную, содержащую значение функции по своему усмотрению.

Оператор **return** может

- находиться в любом месте функции;
- может быть осуществлен несколько раз;
- использоваться без значения, для того чтобы прекратить выполнение и выйти из функции.

```
<script type="text/javascript">
  function checkAge(age) {
  if (age > 18) {
    return true;
  } else {
    return confirm('Родители разрешили?');
  }
}

var age = prompt('Ваш возраст?');

if (checkAge(age)) {
  alert( 'Доступ разрешен' );
} else {
  alert( 'В доступе отказано' );
}
</script>
```

Выбор имени функции:

Как правило, при назначении имени функции используют **глагольные префиксы**, обозначающие общий характер действия, после которых следует уточнение.

show – что-то показывают (показать сообщение).

get – получают;

calc – вычисляют;

create – создают;

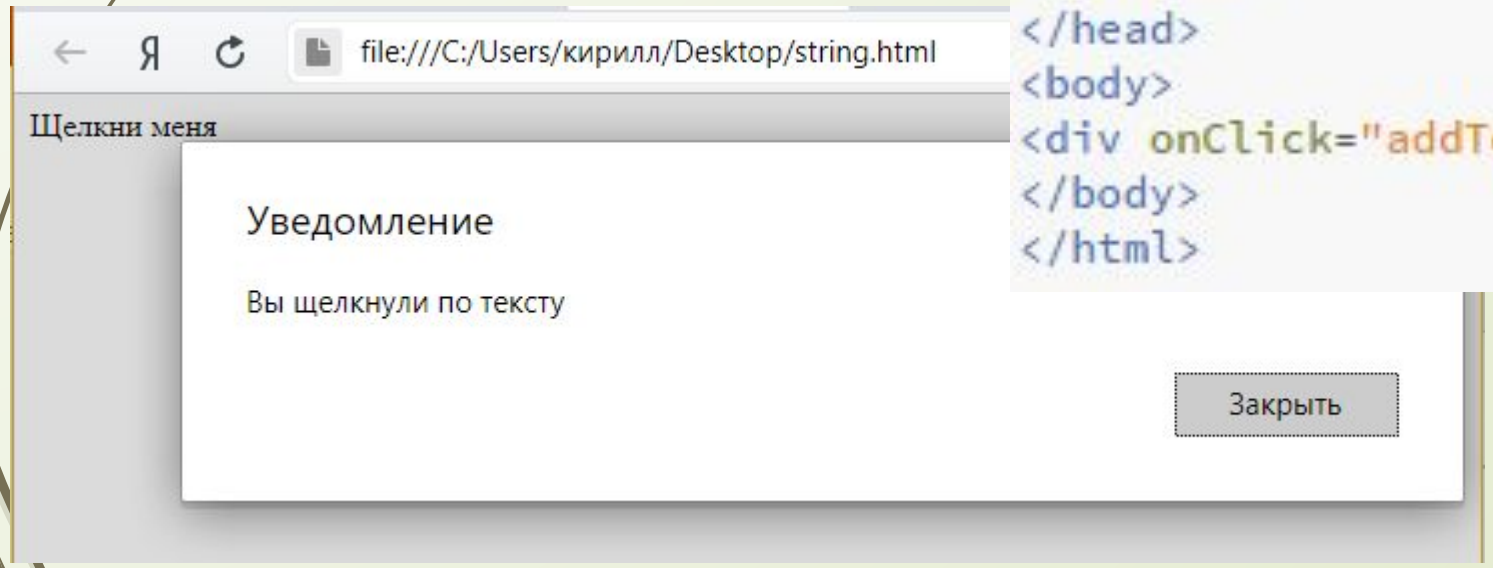
check – проверяют.

Далее следуют поясняющие слова, которые принято начинать каждое с большой буквы.

События и обработчик события

Сценарий может быть выполнен после определенных действий пользователя. Такие действия называют **событиями**.

```
<html>
<head>
<script>
  function addText() {
    alert("Вы щелкнули по тексту");
  }
</script>
</head>
<body>
<div onClick="addText();">Щелкни меня</div>
</body>
</html>
```

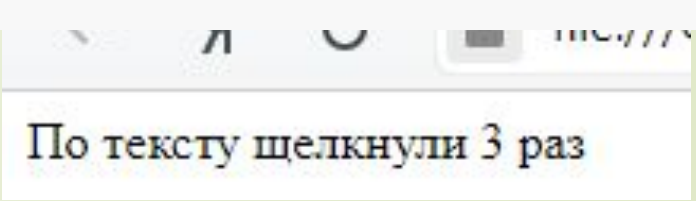


События:

Событие	Когда происходит	Обработчик события
Blur	Потеря объектом фокуса	onBlur
Change	Пользователь изменяет значение элемента	onChange
Click	Щелчок мышью по объекту	onClick
DblClick	Двойной щелчок мышью по объекту	onDblClick
DragDrop	Перетаскивание мышью объект	onDragDrop
Error	Возникновение javascript- ошибки	onError
Focus	Окно или элемент формы получает фокус	onFocus
KeyDown	Нажатие клавиши клавиатуры	onKeyDown
KeyPress	Удержание нажатой клавиши клавиатуры	onKeyPress
KeyUp	Отпускание клавиши клавиатуры	onKeyUp
Load	Документ загружается в браузер	onLoad
MouseDown	Нажатие кнопки мыши	onMouseDown
MouseOut	Указатель мыши выходит за пределы элемента	onMouseOut
MouseOver	Указатель мыши помещается над элементом	onMouseOver
MouseUp	Пользователь отпускает кнопку мыши	onMouseUp
Move	Пользователь перемещает окно	onMove
Reset	Нажатие кнопки сброса в форме	onReset
Resize	Изменение размера окна или элемента	onResize
Select	Выбор элемента формы	onSelect
Submit	Нажатие кнопки Отправить в форме	onSubmit
Unload	Закрытие документа	onUnload

Пример: напишем функцию, которая будет подсчитывать количество кликов по тексту.

```
<html>
<head>
<meta charset="utf-8"/>
<title>Изучение JavaScript</title>
<script type="text/javascript">
  var counter = 0;// объявляем глобальную переменную
  function ClickCounter(element){
     counter++;
     element.innerHTML = "По тексту щелкнули " + counter+ " раз";// функция innerHTML позволяет
     перезаписать информацию в элементе к которому мы ссылаемся
  }
</script>
</head>
<body>
    <div onClick="ClickCounter(this)">По тексту щелкнули 0 раз</div>
</body>
</html>
```



Обработка HTML – форм:

Формы предоставляют возможность сгруппировать элементы HTML, пре

Для создания формы используются теги <**form**> и </**form**>.

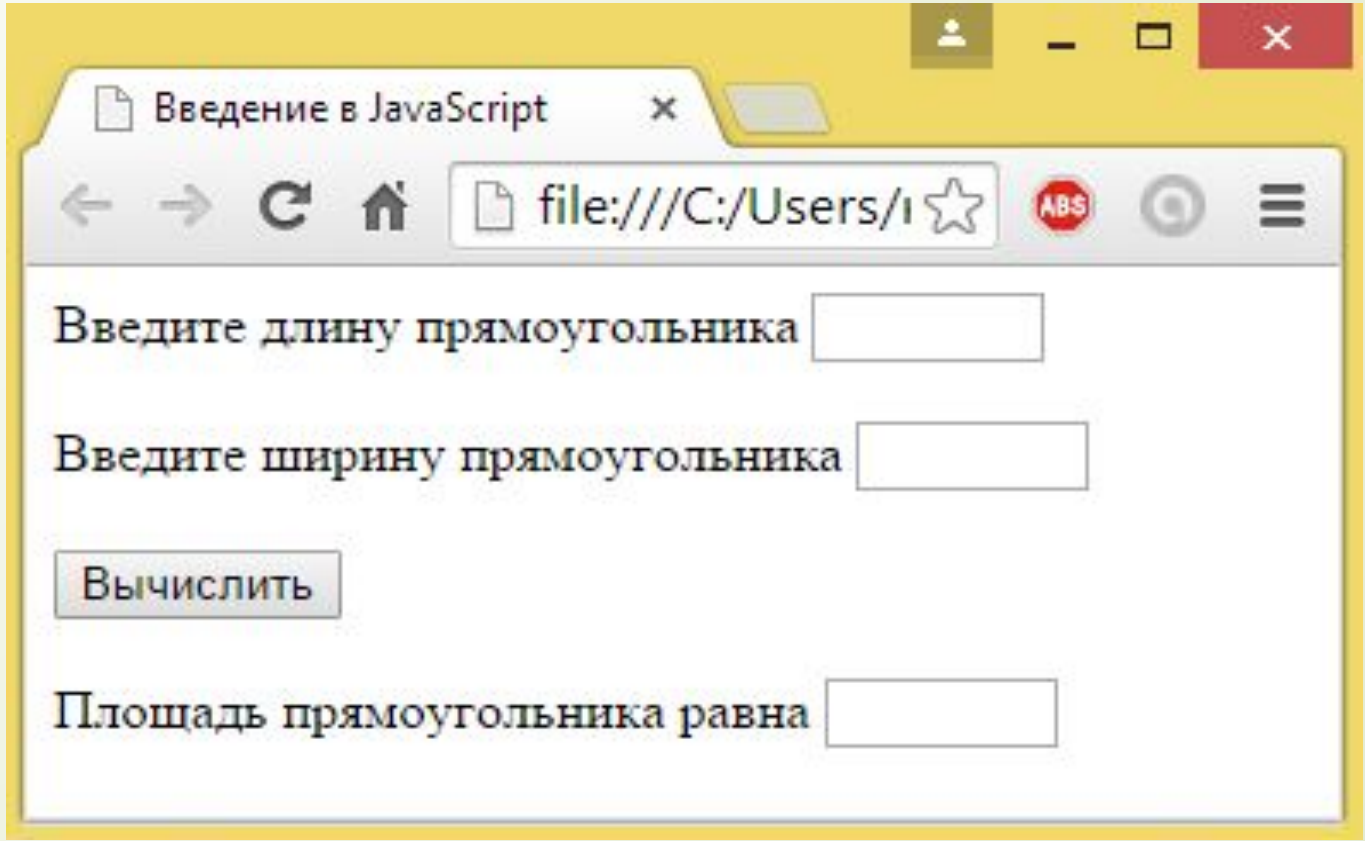
action – определяет, куда передается форма;

method – устанавливает, как будет передаваться информация;

target – определяет фрейм, в который загрузится отклик на передачу формы.

Для **клиентских сценариев** эти атрибуты не обязательны, используется только атрибут **name**, чтобы вы могли ссылаться на нужную форму.

Напишем сценарий, который будет рассчитывать площадь прямоугольника по введенным пользователем длине и ширине.



Код для формы:

```
<html>
<head>
<title> Введение в JavaScript</title>
<script src= "script.js"></script>
</head>
<body>
<form name="form1">
Введите длину прямоугольника <input type="text" name="t1" size="5"> <br><br>
Введите ширину прямоугольника <input type="text" name="t2" size="5"> <br><br>
<input type="button" name="button" value="Вычислить"><br><br>
Площадь прямоугольника равна <input type="text" name="res" size="5">
</form>
</body>
</html>
```

К кнопке привяжем обработчик события:

…

<input type="button" name="button" value="Вычислить" **onClick="plPr ()**">

…

Создаем функцию вычисления площади:

```
function plPr () {
var a = document.form1.t1.value;
var b = document.form1.t2.value;
var s = a*b;
document.form1.res.value = s;
}
```

Параметры функции

Если у нас будет несколько веб – страниц, на которых нам надо вычислить площадь прямоугольника, то нам придется для каждой страницы писать свою функцию.

Разумнее написать один раз функцию и в дальнейшем использовать ее на всех страницах. Для этого HTML – страница должна каким – то образом указать функции, какие именно значения (с какой страницы) брать для вычисления. Здесь нам понадобятся параметры.

В функции используем имя формы - form1, его и сделаем параметром.

```
function plPr (obj) {
var a=obj.t1.value;
var b=obj.t2.value;
var s=a*b;
obj.res.value=s;
}
```

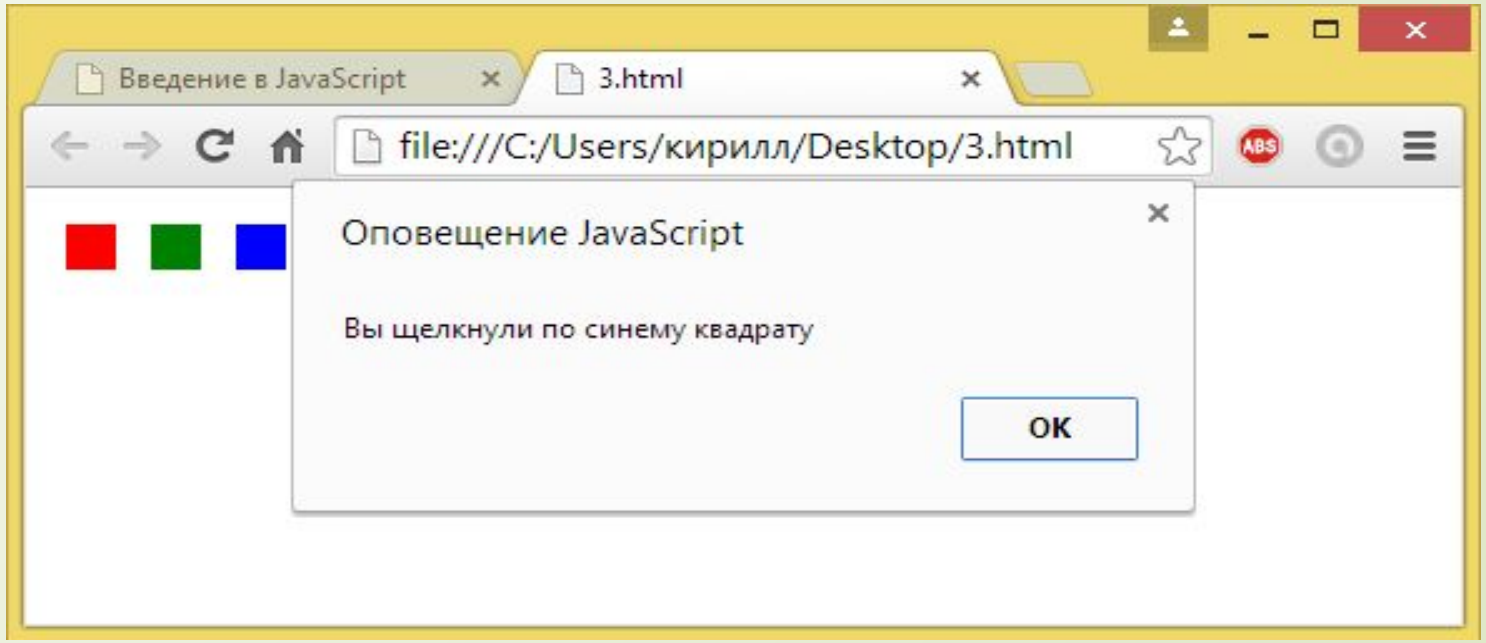
Мы указали, что функция должна принять в качестве параметра какой – то объект (obj) и производить все действия с ним.

```
…
<input type="button" name="button" value="
вычислить" onClick="plPr (form1)">
…
```

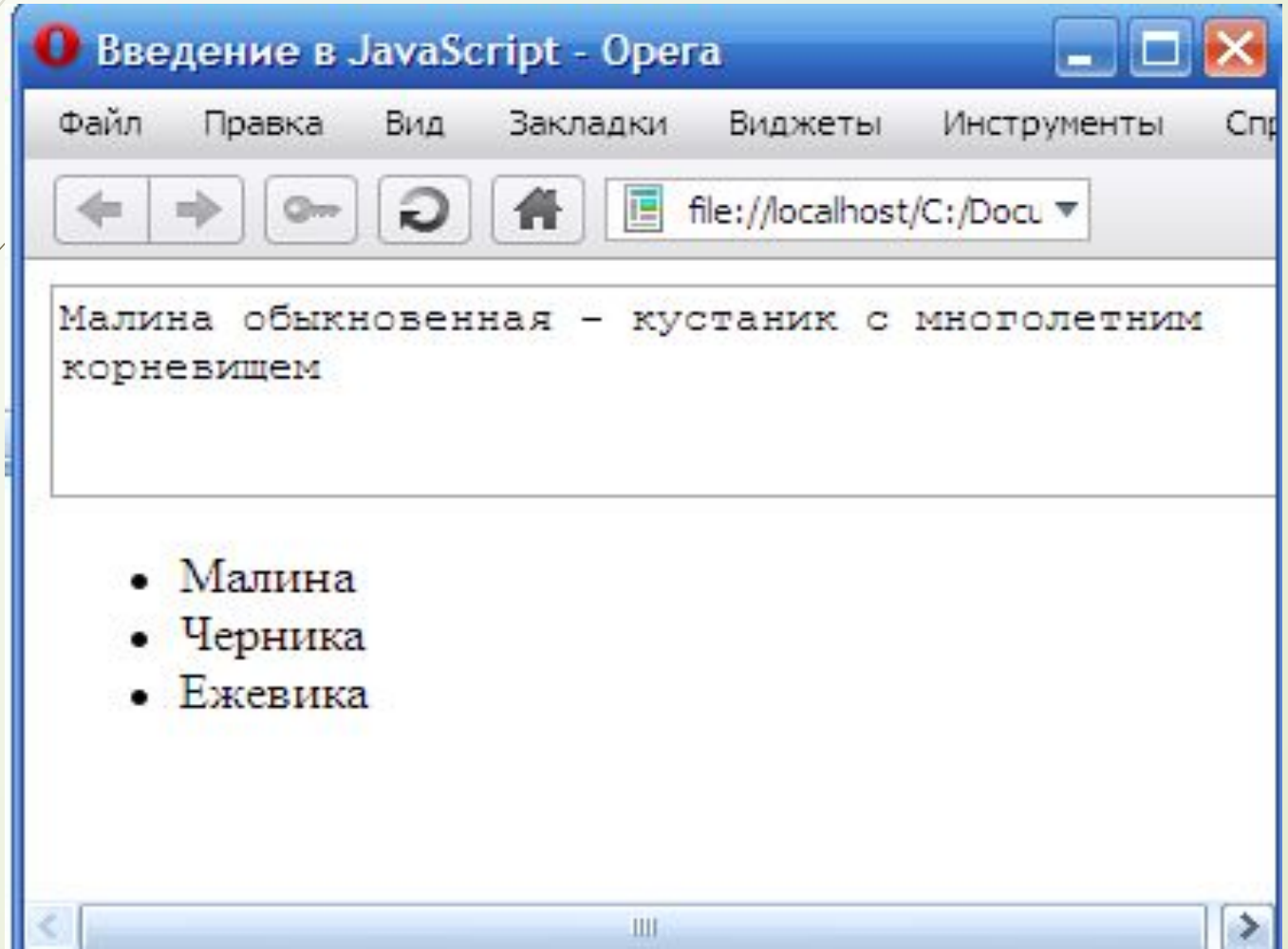
Задача: у нас три квадрата, при щелчке по каждому должно появляться окно с сообщением, где указывается цвет квадрата, по которому щелкнули.

```
function soob (m) {
alert(m);
}
```

```
<table><tr>
<td><div id="red" onClick="soob('Вы щелкнули по красному квадрату')"></div></td>
…
<td><div id="blue" onClick="soob('Вы щелкнули по синему квадрату')"></div></td>
</tr></table>
```



Задача: Пусть у нас будет список ягод, а при наведении мышкой на название ягоды, ее описание появится в текстовом поле.



Функция:

```
function ИмяФункции(obj, n) {
obj.desc.value=n;
}
```

HTML – код:

```
<body>
<form name="ИмяФормы">
<textarea name="desc" cols=   rows= ></textarea>
</form>
<ul>
<li onMouseOver="ИмяФункции(ИмяФормы, 'Малина обыкновенная - кустарник с многолетним корневищем');">Малина</li>
…
</ul>
</body>
```