

ЛАБОРАТОРНАЯ РАБОТА № 2

ТЕМА: 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.

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:

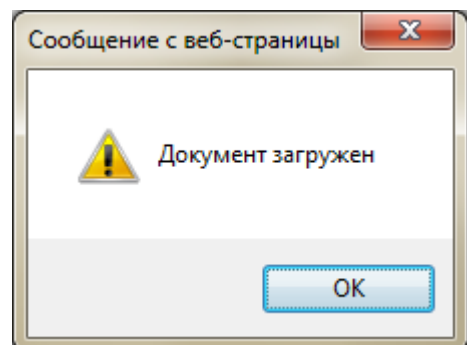
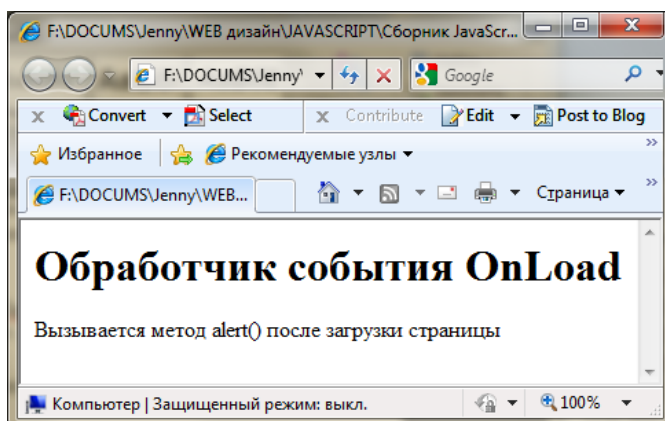
```
<input type="button" value="Нажми Меня" onclick="alert('Спасибо!');" />
```

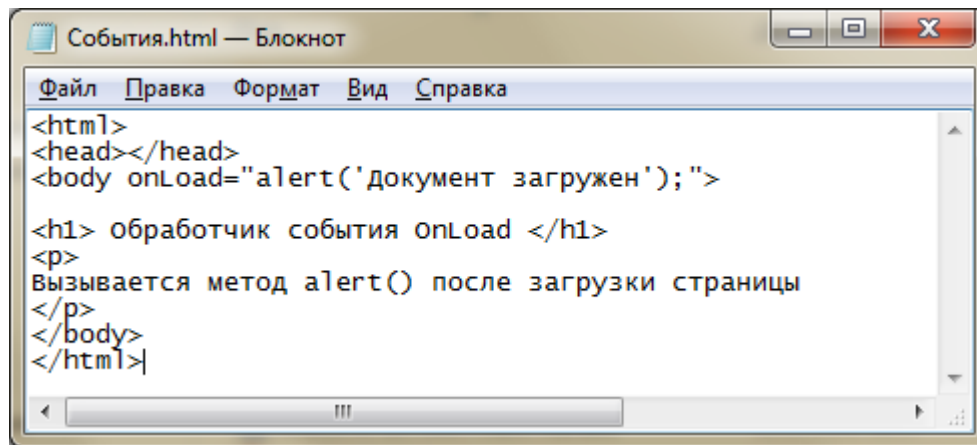
В этом случае JavaScript код пишется в кавычках в одну строку.

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().

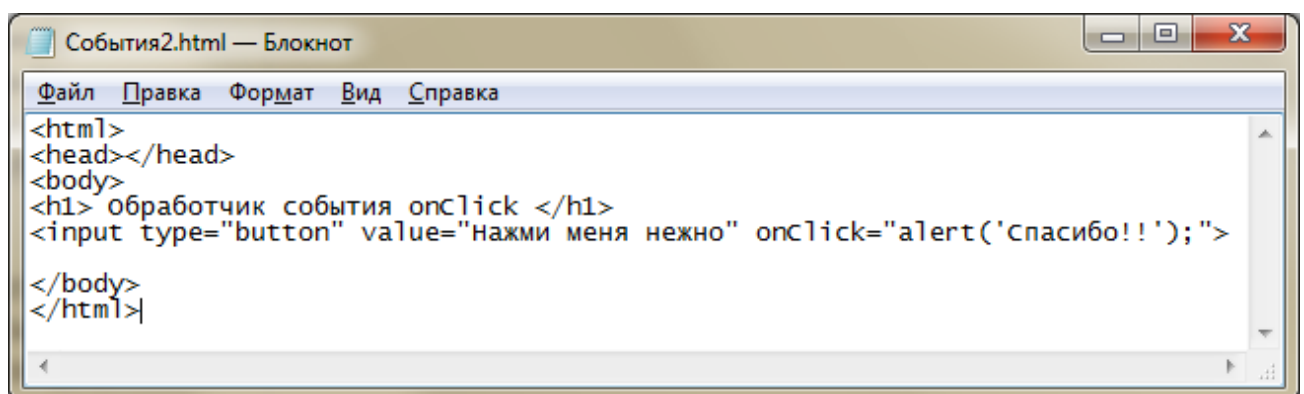
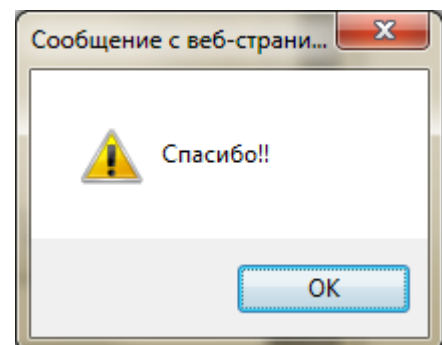
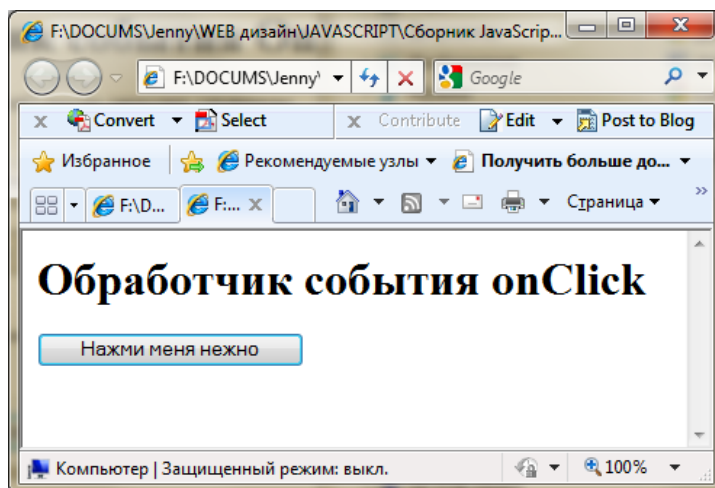




Событие Load и его обработчик onLoad

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:

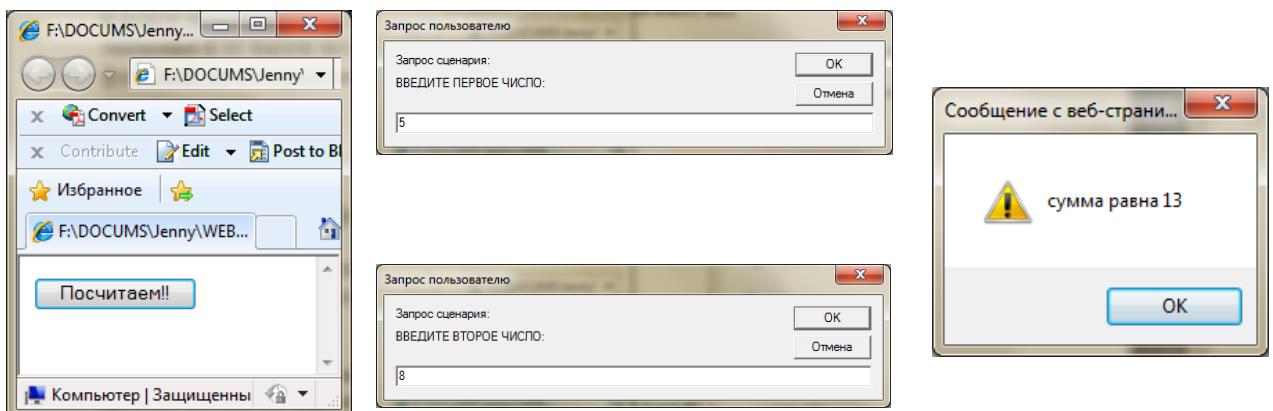
```
Имя_функции(список фактических аргументов через запятые)
```

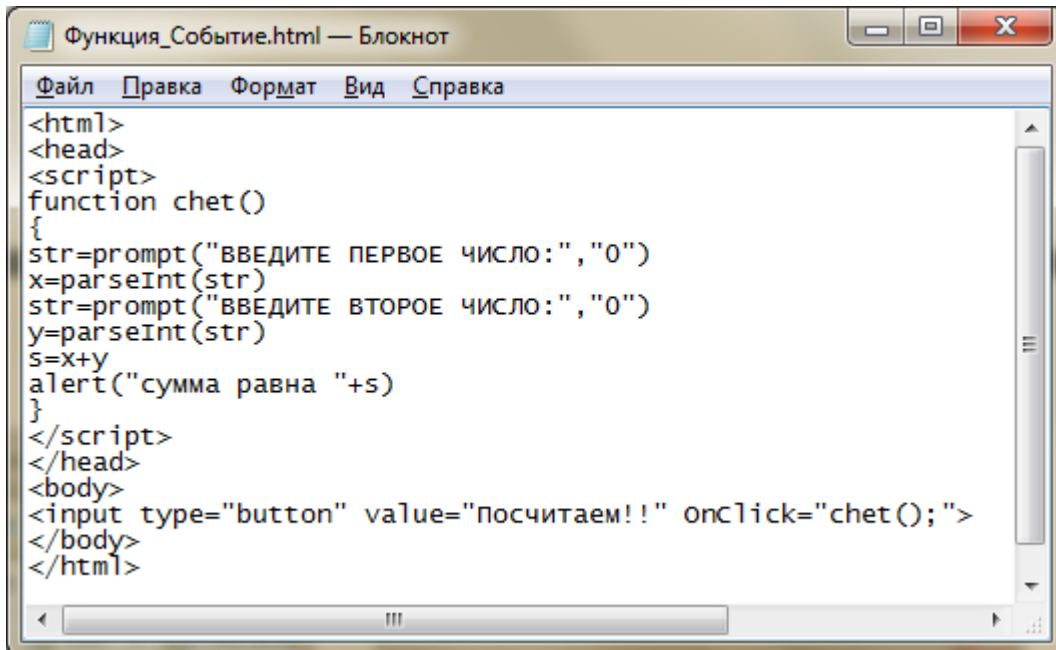
Фактическим аргументом функции может быть константа, переменная, выражение и, в частности, вызов другой функции.

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.





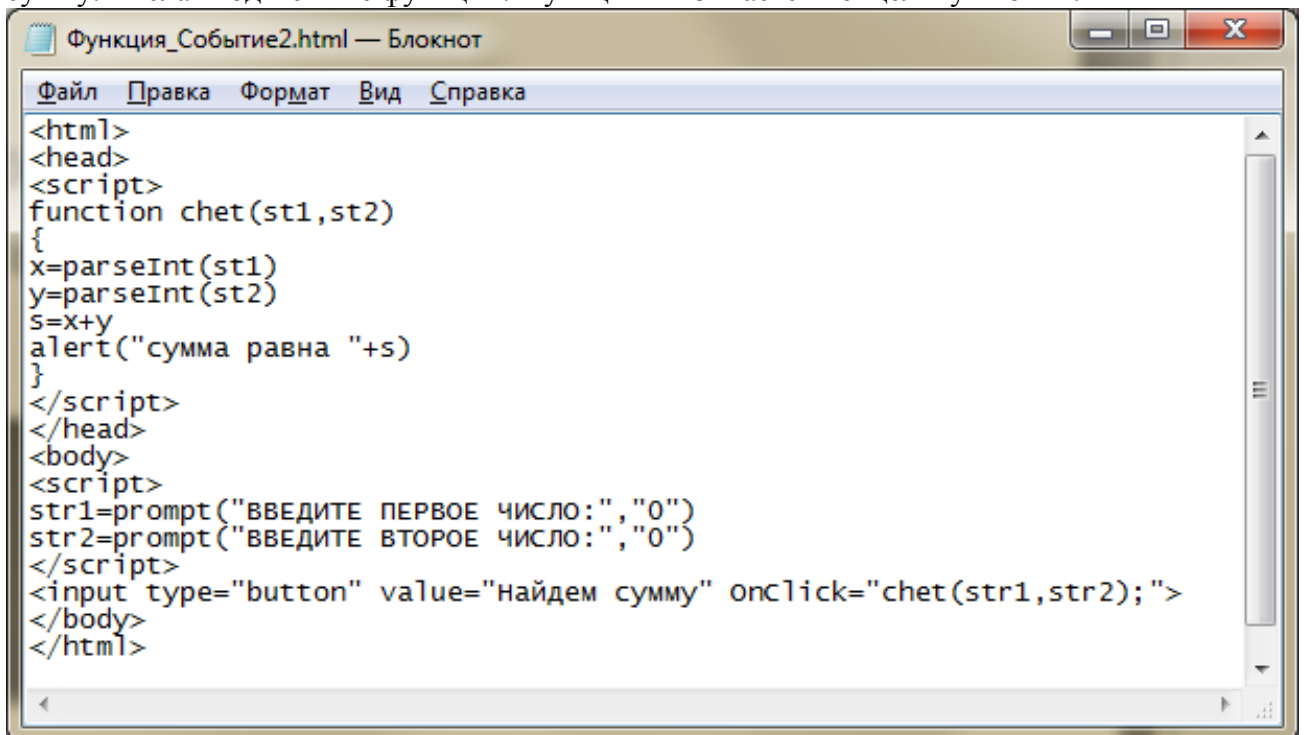
```

<html>
<head>
<script>
function chet()
{
str=prompt("ВВЕДИТЕ ПЕРВОЕ ЧИСЛО:", "0")
x=parseInt(str)
str=prompt("ВВЕДИТЕ ВТОРОЕ ЧИСЛО:", "0")
y=parseInt(str)
s=x+y
alert("сумма равна "+s)
}
</script>
</head>
<body>
<input type="button" value="посчитаем!!" onClick="chet();" >
</body>
</html>

```

Пример 2

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



```

<html>
<head>
<script>
function chet(st1,st2)
{
x=parseInt(st1)
y=parseInt(st2)
s=x+y
alert("сумма равна "+s)
}
</script>
</head>
<body>
<script>
str1=prompt("ВВЕДИТЕ ПЕРВОЕ ЧИСЛО:", "0")
str2=prompt("ВВЕДИТЕ ВТОРОЕ ЧИСЛО:", "0")
</script>
<input type="button" value="найдем сумму" onClick="chet(str1,str2);" >
</body>
</html>

```

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

Задание

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