
Уеб технологии

4 и 5 к., спец. БИС,
3к., спец. “Информатика”
2016-2017 уч. година



Програмен език - JavaScript

Масиви

- ❖ инициализация

```
var M;  
M = [5, 3, 4, 1, 2];
```

- ❖ номерирането на елементите от масива започва от 0

```
num0 = M[0]  
j = M[3]
```

- ❖ промяна и добавяне на елементи в масив

```
M[0] = 7;  
M[7] = 11;  
[7, 3, 4, 1, 2, undefined, 11]
```

- ❖ многомерни масиви

```
M[0] = [3, 2, 1];  
j = M[0][1];
```

Пример (1)

```
<html>
<head>
<meta charset="utf-8">
  <title>Намиране на Min елемент на масив</title>
</head>
<body>
  <script>
var M = [7, 13, 9, 3, 88, 6, 1, 10, 13];
Min = M[0];    // Minimum
Min_ind = 0;   // Index of Min
document.write('Масив: <br>' + Min + " ");
for (i=1; i<M.length; i++) {
  if (M[i] < Min) {
    Min = M[i];
    Min_ind = i;
  } // if
  document.write(M[i] + " ");
} //for
document.write(' <br>Минимален: ' + Min);
document.write(' <br>Индекс на елемента: ' + Min_ind);
</script>
</body>
</html>
```

Масив:
7 13 9 3 88 6 1 10 13
Минимален: 1
Индекс на елемента: 6

Пример (2)

```
<html>
<head>
  <meta charset="utf-8">
  <title>Намиране на максимален елемент в двумерен масив</title>
</head>
<body>
  <script>
    var M = [0, 0, 0];
    M[0] = [3, 2, 1];
    M[1] = [7, 8, 9];
    M[2] = [5, 6, 7];
    Max = M[0][0];    // Max
    Ind = [0, 0];     // Ind of Max
    for (i=0; i<M.length; i++) {
      for (j=0; j<M[i].length; j++) {
        if (M[i][j] > Max) {
          Max = M[i][j];
          Ind[0] = i;
          Ind[1] = j;
        }
      }
    }
    document.write('Максималният елемент е ' + Max);
    document.write('<br>Индексът на максималния елемент е [' + Ind[0] + '][' + Ind[1] + ']');
  </script>
</body>
</html>
```

Максималният елемент е 9
Индексът на максималния елемент е [1][2]

Функции

- ❖ фрагмент от JavaScript код, който може да бъде извикан от произволно място в сценария.
- ❖ синтаксис

```
function <име на функция> ([параметри]) {  
  <тяло на функцията>  
  [return <стойност>]  
}
```

Функции

- ❖ JavaScript допуска използване на анонимни функции

```
var x = function() {  
    window.alert( 'Test' );  
}  
x( );
```

- ❖ в произволна променлива може да се съхранява указател към функция

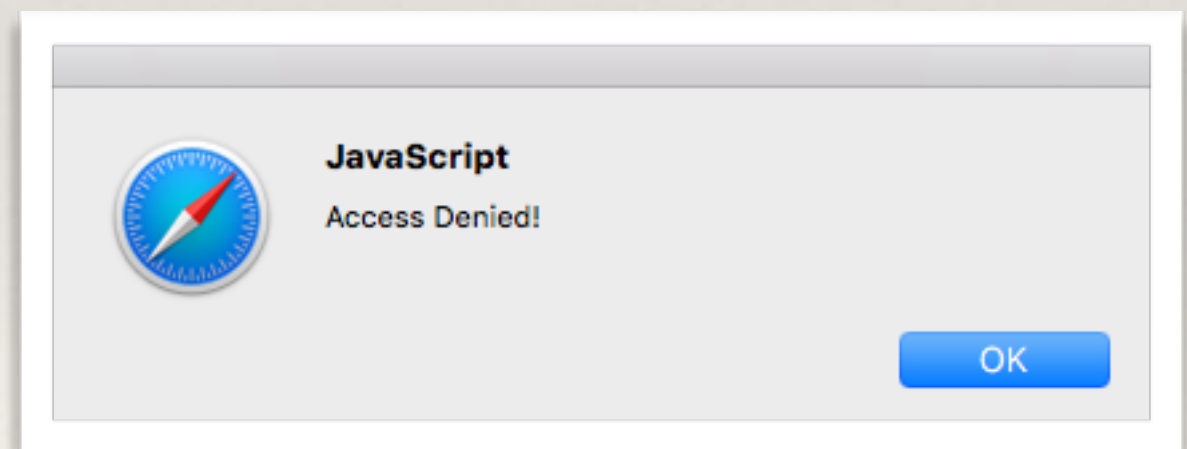
```
var d = denied;  
d( );
```

Разполагане на функциите

- ❖ функцията може да се намира на произволно място в сценария, но това място трябва да е преди мястото, на което функцията се използва за първи път, често се поставят в `head` (заглавната част)
- ❖ ако функциите са голям брой могат да се поставят в отделен `.js` файл

Пример (1) - функция в тага <head>

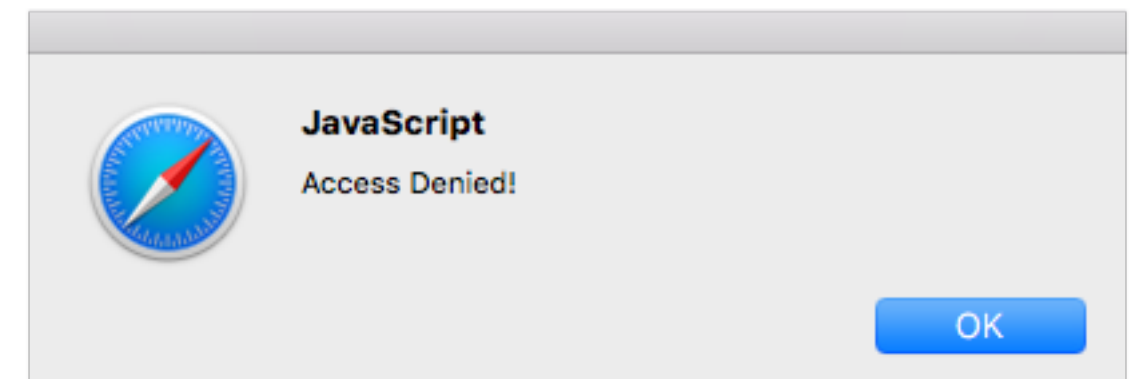
```
<html>
<head>
  <meta charset="utf-8">
  <title>ФУНКЦИЯ, КОЯТО ПОКАЗВА ДИАЛОГОВ ПРОЗОРЕЦ 'Access
denied'</title>
  <script>
    function denied() {
      window.alert('Access Denied!');
    }
  </script>
</head>
<body>
  <script>
    denied();
  </script>
</body>
</html>
```



Пример (2) - функцията е във външен файл

```
<html>
<head>
  <meta charset="utf-8">
  <title>ФУНКЦИЯ ВЪВ ВЪНШЕН ФАЙЛ</title>
  <script type="text/javascript"
src="functions.js"></script>
</head>
<body>
  <script>
    denied( );
  </script>
</body>
</html>
```

```
function denied() {
  window.alert('Access Denied!');
}
```



Пример (3)

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <meta charset="utf-8">
```

```
  <title>Пример за функция, която връща резултат</title>
```

```
</head>
```

```
<body>
```

```
<p>Това е функция с два параметъра, която прави изчисление и  
връща резултат:</p>
```

```
<p id="demo"></p>
```

```
<script>
```

```
function myFunction(a, b) {  
  return a * b;
```

```
}
```

```
document.getElementById("demo").innerHTML = myFunction(4, 3);
```

```
</script>
```

```
</body>
```

```
</html>
```

Това е функция с два параметъра, която прави изчисление и връща резултат:

12

Пример (4)

```
<html>
<head>
<meta charset="utf-8">
  <title>Глобални и локални променливи</title>
  <script>
    // Глобални променливи
    var A = 10;
    var B = 20;
    function F1() {
      // Локални променливи
      var X = 10;
      var B = 5;
      document.write("<br>A = " + A);
      document.write("<br>B = " + B);
      document.write("<br>X = " + X);
    }
  </script>
</head>
<body>
  <script>
    F1();
    document.write("<HR>");
    document.write("A = " + A);
    document.write("<br>B = " + B);
  </script>
</body>
</html>
```

Глобални и локални променливи
A = 10 B = 5 X = 10
A = 10 B = 20

Пример (5) - рекурсивна функция

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Рекурсивна функция</title>
<script type="text/javascript" src="fact.js">
</script>
</head>
<body>
  <p>Това е рекурсивна функция, която изчислява факториел</p>
  <p id="demo"></p>
<script>
  document.getElementById("demo").innerHTML = fact(5);
</script>
</body>
</html>
```

```
// JavaScript Document
function fact(x) {
  if (x == 0 || x == 1) return 1;
  else return (x * fact(x - 1));
}
```

Рекурсивна функция

Това е рекурсивна функция, която изчислява факториел

120

Вградени класове

❖ клас Global

Методи на класа:

- parseInt(<низ> <бройна система>) - преобразува низ в цяло число
- parseFloat(<низ>) - преобразува низ в число с плаваща запетая
- eval(<низ>) - изчислява израза, зададен в низа
- isNaN(<израз>) - проверява дали изразът е число

...

Примери:

```
var str = "100";  
var x = 50 + parseInt(str);      //150  
var strf = "100.52";  
var y = parseFloat(strf);        //100.52  
var z = eval("2 + 2");           //4
```

Вградени класове

❖ **клас Number** - за работа с числа

`var <обект> new Number(начална стойност)`

`var x = new Number(1) // създаване на екземпляр на класа`

Методи на класа:

- `valueOf()` - връща числовата стойност на екземпляр от класа

- `toString()` - връща низовото представяне на число

Свойства:

`MAX_VALUE`, `MIN_VALUE`, `NaN`, `NEGATIVE_INFINITY`, `POSITIVE_INFINITY`

Примери:

`var x = Number.MAX_VALUE;`

`var y = new Number(100);`

`var str = y.toString(); // "100"`

Вградени класове

- ❖ клас **String** - за обработка на низове

```
var s = new String("Hello");
```

СВОЙСТВО:

length

Пример:

```
var so = new String("Hello");  
document.write(so.length); // 5
```

Методи:

toString(), valueOf(), charAt(), ...

Вградени класове

- ❖ клас **Array** - за обработка на масиви

`var <обект> = new Array(<брой елементи в масива>);`

`var <обект> = new Array(<елементи, отделени със запетая>);`

`var M = new Array(1, 2, 3);`

Свойство:

`length`

Пример:

`document.write(M.length);`

Методи:

`push(<елементи>), unshift(<елементи>), pop() ...`

Вградени класове

- ❖ клас **Math** - съдържа математически функции

Методи

`abs()`, `log()`, `sqrt()`, `round()`, ...

- ❖ клас **Date** - за работа с дата и време

Методи

`getDate()`, `getDay()`, `getHours()`, ...

Събития

- ❖ възникват в резултат на взаимодействието на потребителя с веб страницата



Събития, свързани с мишката

Събитие	Описание
onmousedown	при натискане на бутон на мишката
onmouseup	при отпускане на натиснат бутон на мишката
onclick	при натискане и пускане (кликване) на бутон на мишката
ondblclick	при двойно кликване
onmouseover	позициониране на мишката над елемент
onmouseout	когато курсорът излезе извън очертанията на елемент
onmousemove	при преместване на мишката
onselect	при избор на елемент
oncontextmenu	при натискане десен бутон на мишката и показване на контекстното меню

Събития, свързани с клавиатурата

Събитие	Описание
<code>onkeydown</code>	при натискане на клавиш от клавиатурата
<code>onkeypress</code>	същото, но връща кода на натиснатия символ
<code>onkeyup</code>	когато се отпусне натиснат клавиш
<code>onhelp</code>	при натискане на клавиш F1

Събития, свързани с уеб страницата

Събитие	Описание
onload	веднага след зареждане на уеб страницата
onscroll	при скролиране
onresize	при промяна размера на прозореца
onunload	при напускане на текущата страница
onbeforeunload	при напускане на текущата страница, преди събитието onload
onbeforeprint	при разпечатване на страницата
onafterprint	след разпечатване на страницата

Събития, свързани с формите

Събитие	Описание
onsubmit	при изпращане на формата, натискане на бутон submit
onreset	при изчистване на формата, натискане на бутон reset
onblur	когато елемент от формата губи фокус
onfocus	когато елемент от формата получава фокус
onchange	когато фокусът се премества върху друг елемент

Пример - обработка на събития от мишката

```
<html>
<head>
<meta charset="utf-8">
<title>Обработка на събития от мишката</title>
</head>
<body onload="window.alert( 'OnLoad. Моля натиснете бутон
на мишката' );"
  onmousedown="document.write( 'OnMouseDown' );"
  <h1>Събития, свързани с мишката</h1>
</body>
</html>
```



JavaScript

OnLoad. Моля натиснете бутон на мишката

ОК

Обработка на събития от мишката

Събития, свързани с мишката

Обработка на събития от мишката

OnMouseDown

Работа с форми

- ❖ тагът `<form>` и `</form>` в HTML документ създава елемент в масив `forms` в реда, в който се появяват в документа.
- ❖ достъпът до първата форма в документа става чрез `document.forms[0]` (номерирането започва както при всички масиви от 0)
- ❖ формата може да се достъпва и чрез името, което е зададено в отварящия таг `<form>` - `document.ime_na_forma`
- ❖ `document.forms[i].length` – намира броя елементи на формата с индекс `i`;
- ❖ `document.forms.length` – намира броя форми в страницата

Свойства на обекта на формата

- ❖ `length` - брой елементи във формата
- ❖ `action` - URL адрес на сценария, който обработи формата
- ❖ `elements` - връзка към съвкупността `elements`
- ❖ `encoding` - тип на предаваните данни
- ❖ `method` - режим на предаване на данните
- ❖ `enctype` - метод на кодиране на данните от формата
- ❖ `name` - името на формата

Свойства на обекта forms

- ❖ action – задава се изпратят данните

```
document.getElementById("myForm").action = "form_action.asp";
```

Работа с форми

- ❖ на всеки един елемент от формата се присвоява име, по което след това става обръщение към елемента

Пример:

// обръщение към текстово поле от форма

```
document.forms["form1"].firstname.value
```

```
document.form1.firstname.value
```

// обръщение по индекс, номер на формата

```
document.forms[0].firstname.value
```

// чрез метода getElementById()

```
document.getElementById("firstname").value
```

// чрез elements

```
documents.forms["form1"].elements["firstname"].value
```

```
documents.forms["form1"].elements[0].value
```

```
documents.forms[0].elements[0].value
```

Методи на обекта на формата

- ❖ `submit()` - изпраща данните
- ❖ `reset()` - изчиства формата

Текстови полета

- ❖ `<input type = “text”>`
- ❖ `<input type> = “password”>`

Свойство	Описание
value	стойност на елемента на формата
defaultValue	начална стойност на елемента
disabled	ако е true полето е неактивно
form	връзка към формата
maxLength	максималният брой символи, които могат да се въведат в полето
name	име на елемента
readOnly	ако е true не може да се редактира
type	тип на елемента на формата

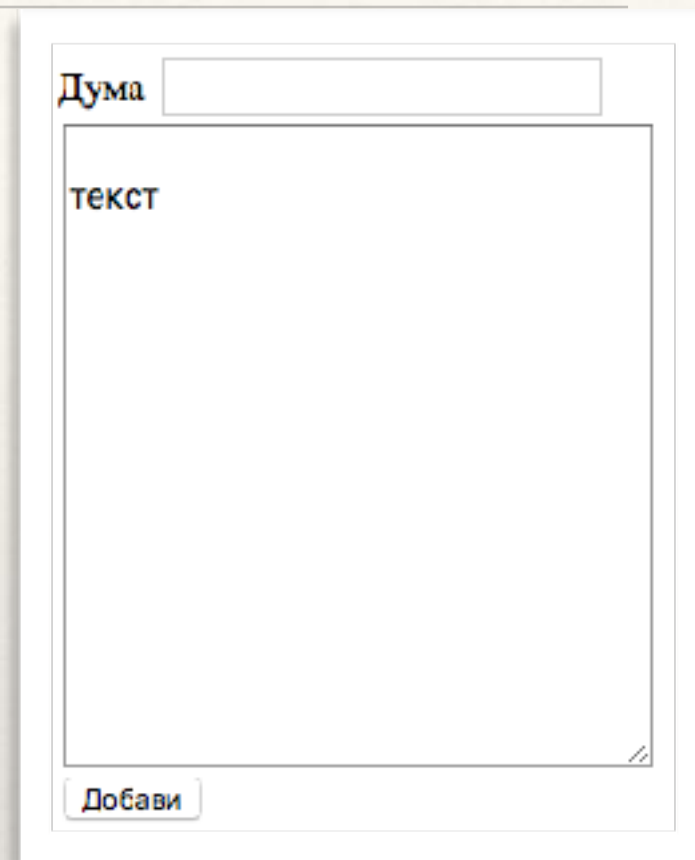
Текстови полета

Метод	Събитие	Описание
blur()	onblur	маха фокуса от текущия елемент на формата
focus()	onfocus, onchange	премества фокуса на текущия елемент на формата
select()		селектира текста в полето

- ❖ текстовите полета `textarea` имат свойство `wrap` със стойности - `off`, `physical`, `virtual`

Пример (1)

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Forms</title>
  </head>
<body>
  <script>
    function AddWord() {
      var text1 = document.form1.text1.value;
      if (text1 == "") { alert('Моля въведете текст'); return ""; }
      var ta1 = document.form1.ta1.value;
      var result = ta1 + "\n" + text1;
      document.form1.ta1.value = result;
      document.form1.text1.value = "";
      return text1;
    }
  </script>
  <form name="form1">
    Дума <input type="text" name="text1" id="text1"><br>
    <textarea name="ta1" id="ta1" cols="25" rows="15"></textarea><br>
    <input type="button" value="Добави" onclick="AddWord();"><br>
  </form>
</body>
</html>
```



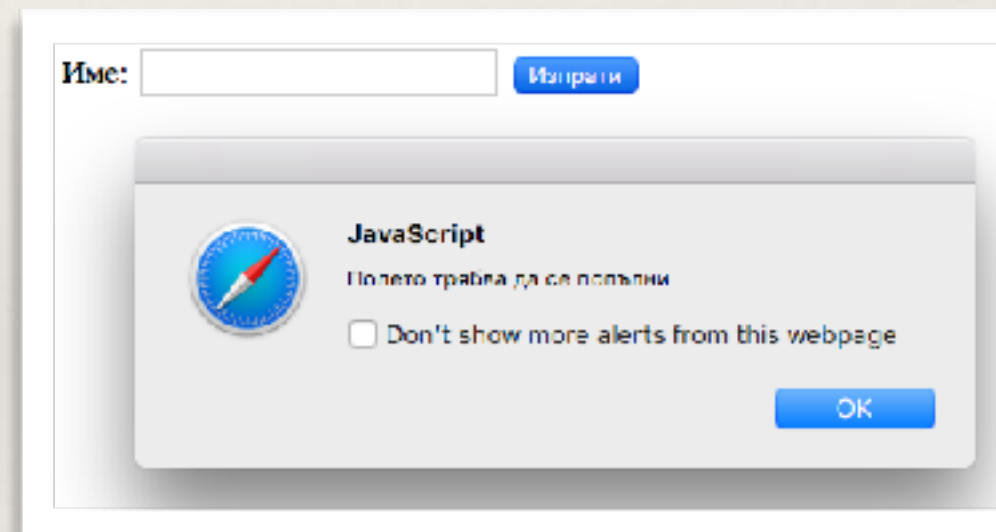
The screenshot shows a web browser window displaying a form. At the top, there is a text input field labeled "Дума" (Word). Below it is a large text area labeled "текст" (text). At the bottom of the form is a button labeled "Добави" (Add).

Валидизиране на форми

- ❖ при избор на бутон `submit`, възниква събитие за изпращане, което може да се прихване от функцията `onSubmit` в отварящия таг `<form>` и там да се извика функция за проверка за валидност на данните
- ❖ във функцията за обработка на събития `onSubmit` трябва да се постави `return` за може изпращането да се осъществи, ако `onSubmit="return true"` и да не се изпълни при `onSubmit="return false"`

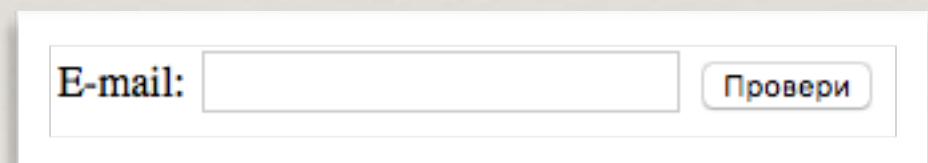
Проверка - дали е попълнено текстово поле

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script>
function validateForm() {
    var x = document.forms[ "myForm" ][ "fname" ].value;
    if (x == null || x == "") {
        alert("Полето трябва да се попълни");
        return false;
    }
}
</script>
</head>
<body>
<form name="myForm" action="demo_form.asp"
onsubmit="return validateForm()" method="post">
Име: <input type="text" name="fname">
<input type="submit" value="Изпрати">
</form>
</body>
</html>
```



Проверка за валиден e-mail

```
<html>
<head>
  <meta charset="utf-8">
<title>Форми</title>
<script>
function validateEmail(email) {
var re = /^(([^<>()[\]\.\\.,;:\s@"]+(\.[^<>()[\]\.\\.,;:\s@"]+)*|("[\]+\.")@((\[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\)|((\[a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,})))$/;
return re.test(email);
}
function check() {
var email = document.form1.email.value;
if (validateEmail(email)) window.alert("OK");
else window.alert("Email is not valid");
}
</script>
<body>
  <form name="form1">
    E-mail: <input type="text" name="email" id="email">
    <input type="button" value="Провери" onclick="check();" >
  </form>
</body>
</html>
```



E-mail:

Проверка за правилно въвеждане

```
<body>
<h1>Проверка дали въведената стойност е между 1 и 10</h1>
<p>Моля въведете число между 1 и 10:</p>
<input id="numb">
<button type="button" onclick="myFunction()">Въведи</button>
<p id="demo"></p>
<script>
function myFunction() {
    var x, text;
    // Вземане на стойността от полето с id="numb"
    x = document.getElementById("numb").value;
    // Ако x не е число или е по-малко от 1 или по-малко от 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Въведеното число не е в интервала между 1 и 10";
    } else {
        text = "Въведеното число е в интервала от 1 до 10";
    }
    document.getElementById("demo").innerHTML = text;
}
</script>
</body>
```

Проверка дали въведената стойност е между 1 и 10

Моля въведете число между 1 и 10:

Пример - галерия

http://www.w3schools.com/bootstrap/bootstrap_carousel.asp

