



ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

ОСНОВИ НА WEB ПРОГРАМИРАЊЕ

- JavaScript -

Основни поими за JavaScript

- **JavaScript** е т.н. scripting language (lightweight programming language)
- Овозможува интерактивност на HTML страни
- Една JavaScript се состои од линии на егзекутабилен компјутерски код
- JavaScript е т.н. interpreted language односно скриптите се извршуваат без претходна компилација (сите веб пребарувачи денес поседуваат вграден JavaScript интерпретер)
- Не е потребна лиценца
- Се користи во голем број веб страни денес и е најпопуларниот скриптирачки јазик поддржан од сите поголеми веб пребарувачи

Зошто JavaScript

- JavaScript ги збогатува веб страните со динамички и интерактивни содржини
- Работи во клиентскиот софтвер
- Обезбедува динамичка содржина на клиентска страна
- Нуди поголема респонсивност на корисничкиот веб интерфејс
 - Ја избегнува латентноста од пристап до сервер
- Динамички го менува HTML-от кој се прикажува во веб пребарувач

JavaScript ≠ Java

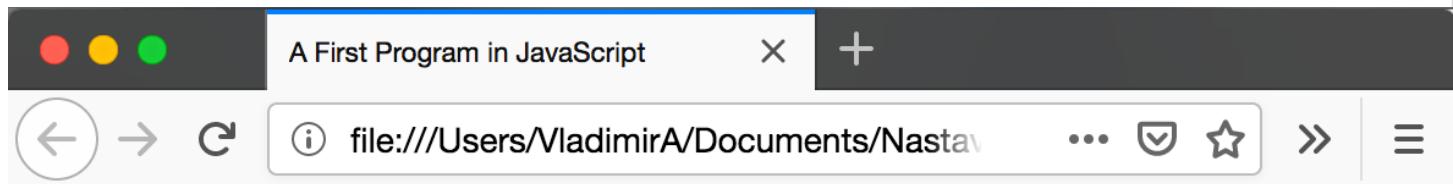
- JavaScript и Java се комплетно различни јазици и по концепт и по дизајн
- Java (развиен од Sun Microsystems) е моќен и комплексен програмски јазик – во иста категорија со C и C++
- JavaScript (развиен од Netscape, потоа поддржан од Microsoft и ECMA) не може да црта, не поддржува multithreading, не може да користи I/O
 - Лесен програм кој може локално да се симне и изврши во пребарувач и е компатибilen на многу платформи

JavaScript	Java
Interpreted by the client-side computer	Compiled on the server before executed on the client machine
Dynamic binding, object references are checked at runtime	Static binding, object references must exist at compile time
No need to declare data types	Data types must be declared
Code is embedded in HTML	Code is not integrated in HTML
Limited by the browser functionality	Java applications are standalone
Can access browser objects	Java has no access to browser objects

Едноставна JavaScript

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.1: welcome.html -->
4 <!-- Displaying a line of text. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>A First Program in JavaScript</title>
9     <script type = "text/javascript">
10
11       document.writeln(
12         "<h1>Welcome to JavaScript Programming!</h1>" );
13
14   </script>
15 </head><body></body>
16 </html>
```

- JavaScript програмата е напишана во <head>
 - Може да биде и во <body> => inline scripting
- Користење на објектот document од веб пребарувачот
- Со објектот се асоцираат атрибути (податоци) и методи
- Скриптата е клиент на објектот



Welcome to JavaScript Programming!

JavaScript source file

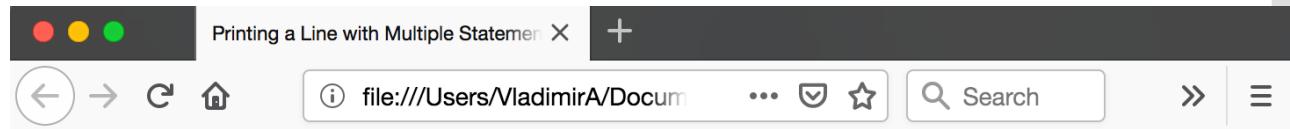
```
<script language="JavaScript"  
src="your_source_file.js"></script>
```

- **SRC** – локацијата на екстерната скрипта
- **LANGUAGE** – спецификација на скриптирачкиот јазик на скриптата

- Згодна опција за споделување на JavaScript код од страна на повеќе HTML документи
- Можност за криење на JavaScript кодот од други корисници

Модификација на примерот

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.2: welcome2.html -->
4 <!-- Printing one line with multiple statements. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Printing a Line with Multiple Statements</title>
9     <script type = "text/javascript">
10       <!--
11         document.write( "<h1 style = 'color: magenta'>" );
12         document.write( "Welcome to JavaScript " +
13           "Programming!</h1>" );
14       // -->
15     </script>
16   </head><body></body>
17 </html>
```

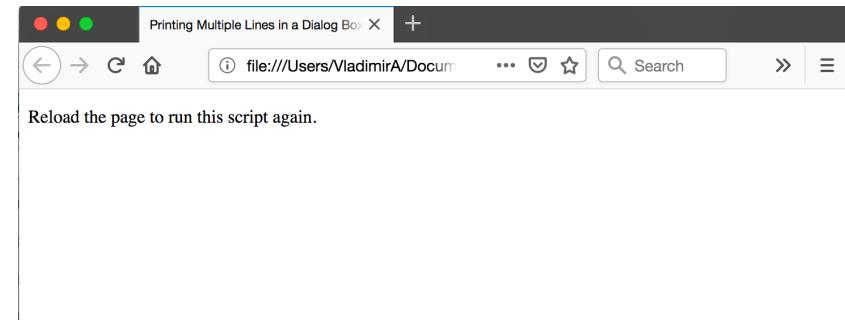
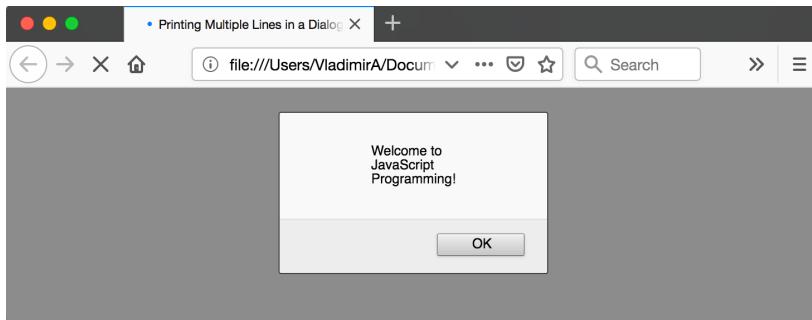


Welcome to JavaScript Programming!

Alert dialog

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.3: welcome3.html -->
4 <!-- Alert dialog displaying multiple lines. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Printing Multiple Lines in a Dialog Box</title>
9     <script type = "text/javascript">
10       <!--
11         window.alert( "Welcome to\nJavaScript\nProgramming!" );
12       // -->
13     </script>
14   </head>
15   <body>
16     <p>Click Refresh (or Reload) to run this script again.</p>
17   </body>
18 </html>
```

- За прикажување на важни податоци на корисниците при прегледување веб страни
- Се користи објектот `window` на пребарувачот и неговиот метод `alert`



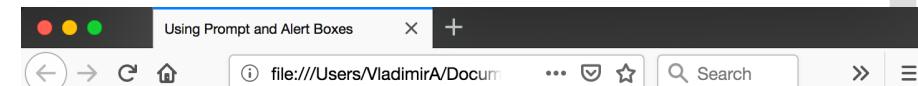
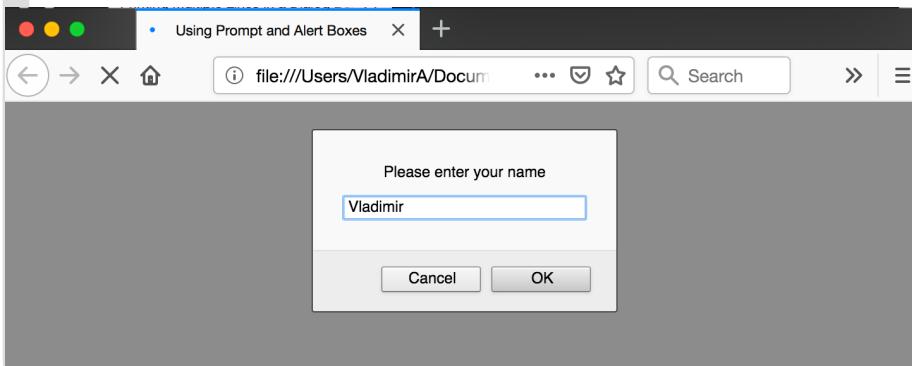
Escape sequences

Escape sequence	Description
\n	<i>New line</i> —position the screen cursor at the beginning of the next line.
\t	<i>Horizontal tab</i> —move the screen cursor to the next tab stop.
\\"	<i>Backslash</i> —used to represent a backslash character in a string.
\"	<i>Double quote</i> —used to represent a double-quote character in a string contained in double quotes. For example, <pre>window.alert(\"in double quotes\");</pre> displays "in double quotes" in an alert dialog.
\'	<i>Single quote</i> —used to represent a single-quote character in a string. For example, <pre>window.alert('\'in single quotes\'');</pre> displays 'in single quotes' in an alert dialog.

Prompt dialog

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.5: welcome4.html -->
4 <!-- Prompt box used on a welcome screen -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Using Prompt and Alert Boxes</title>
9     <script type = "text/javascript">
10       <!--
11         var name; // string entered by the user
12
13         // read the name from the prompt box as a string
14         name = window.prompt( "Please enter your name" );
15
16         document.writeln( "<h1>Hello " + name +
17           ", welcome to JavaScript programming!</h1>" );
18         // -->
19       </script>
20     </head><body>
21   </html>
```

- Скриптирањето овозможува да се генерира дел или целосно една веб страна во моментот кога му се прикажува на корисникот => динамичка веб страна



**Hello Vladimir, welcome to JavaScript
programming!**

Променливи (1/2)

- Променливите се користат за зачувување податоци
- Вредноста на една променлива може да се промени за време на извршување на скриптата
- Правила за имиња на променливи:
 - Case-sensitive
 - Мора да почнуваат со буква, underscore character (_) или dollar sign (\$)
 - strname – STRNAME (not same)
- **Декларација**
 - **Explicit:** var i = 12; // 'var' in declaration
 - **Implicit:** i = 12;
- **Домен на важност (scope)**
 - **Глобален**
 - Декларирани надвор од функции
 - Било која имплицитно дефинирана променлива
 - **Локален**
 - Експлицитни декларации внатре во функции

```
// local variable  
var x = 12;  
// global variable  
y = 12; Prompt Box
```

Променливи (2/2)

- **Dynamic Typing** – променливите може да чуваат било каков валиден тип на вредност:
 - **Number** ... var myInt = 7;
 - **Boolean** ... var myBool = true;
 - **Function** ... [Discussed Later]
 - **Object** ... [Discussed Later]
 - **Array** ... var myArr = new Array();
 - **String** ... var myString = "abc";
- **Специјални податочни типови:**
 - **Null**: иницијална вредност е зададена
 - **Undefined**: променливата е дефинирана, но не и е дадена вредност
 - ... и може да чува вредности од различни типови во различни времиња на извршување на скриптата

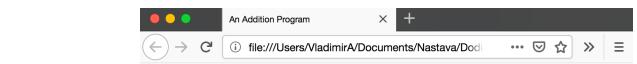
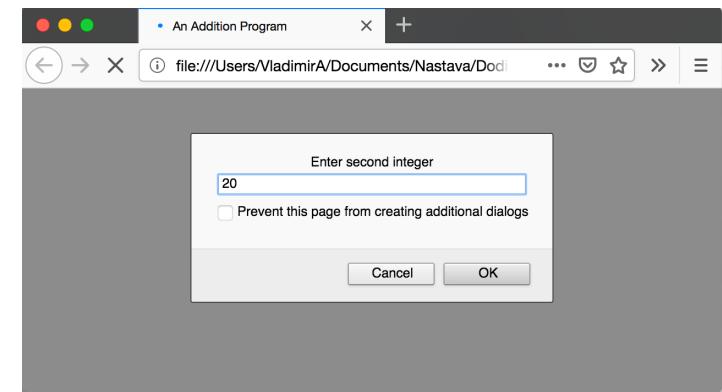
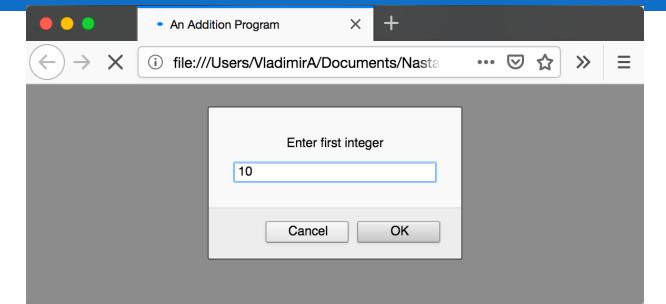
Strings

- Променлива од тип **string** може да чува секвенца од алфанимерички карактери, празни места и специјални карактери
- String може да биде сместен или во единечни или во двојни наводници - **single quotation marks ('), double quotation marks (")**
- Кој е податочниот тип на “100”? String (not number)
- JavaScript **не поседува char податочен тип**

```
<head>
<script language="JavaScript">
    document.write("This is a string.");
    document.write("This string contains 'quote'.");
    var myString = "My testing string";
    alert(myString);
</script>
</head>
```

Собирање цели броеви

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.7: addition.html -->
4 <!-- Addition script. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>An Addition Program</title>
9     <script type = "text/javascript">
10       <!--
11         var firstNumber; // first string entered by user
12         var secondNumber; // second string entered by user
13         var number1; // first number to add
14         var number2; // second number to add
15         var sum; // sum of number1 and number2
16
17         // read in first number from user as a string
18         firstNumber = window.prompt( "Enter first integer" );
19
20         // read in second number from user as a string
21         secondNumber = window.prompt( "Enter second integer" );
22
23         // convert numbers from strings to integers
24         number1 = parseInt( firstNumber );
25         number2 = parseInt( secondNumber );
26
27         sum = number1 + number2; // add the numbers
28
29         // display the results
30         document.writeln( "<h1>The sum is " + sum + "</h1>" );
31         // -->
32     </script>
33   </head><body></body>
34 </html>
```



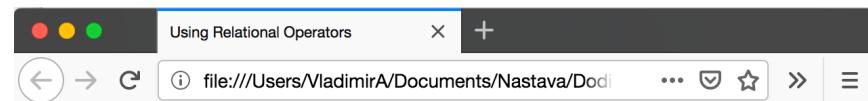
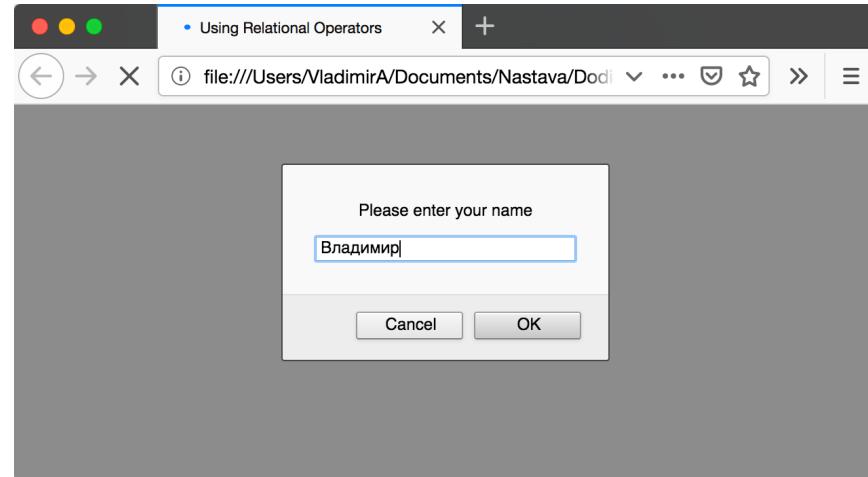
The sum is 30

Equality and relational operators

Standard algebraic equality operator or relational operator	JavaScript equality or relational operator	Sample JavaScript condition	Meaning of JavaScript condition
<i>Equality operators</i>			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
<i>Relational operators</i>			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	≥	x ≥ y	x is greater than or equal to y
≤	≤	x ≤ y	x is less than or equal to y

Decision-making: if услов

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 6.14: welcome5.html -->
4 <!-- Using equality and relational operators. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Using Relational Operators</title>
9     <script type = "text/javascript">
10    <!--
11    var name; // string entered by the user
12    var now = new Date();      // current date and time
13    var hour = now.getHours(); // current hour (0-23)
14
15    // read the name from the prompt box as a string
16    name = window.prompt( "Please enter your name" );
17
18    // determine whether it's morning
19    if ( hour < 12 )
20      document.write( "<h1>Good Morning, " );
21
22    // determine whether the time is PM
23    if ( hour >= 12 )
24    {
25      // convert to a 12-hour clock
26      hour = hour - 12;
27
28      // determine whether it is before 6 PM
29      if ( hour < 6 )
30        document.write( "<h1>Good Afternoon, " );
31
32      // determine whether it is after 6 PM
33      if ( hour >= 6 )
34        document.write( "<h1>Good Evening, " );
35    } // end if
36
37    document.writeln( name +
38      ", welcome to JavaScript programming!</h1>" );
39    // -->
40  </script>
41  </head><body></body>
42 </html>
```



**Good Evening, Владимир, welcome to
JavaScript programming!**

Control statements

- JavaScript има вкупно 8 изјави за контрола (control statements: sequence, selection – 3, repetition - 4)
 - Изјавите (statements) во една скрипта се изведуваат една по друга според редоследот на нивното појавување => sequential execution
 - Поим за transfer of control без goto statement
 - Три изјави за селекција:
 - IF – single-selection statement
 - IF ... ELSE – double-selection statement
 - SWITCH – multiple-selection statement
 - Четири повторувачки изјави:
 - WHILE
 - DO ... WHILE
 - FOR
 - FOR ... IN
- JavaScript има 2 типа на комбинација на изјави за контрола: control-statement stacking и control-statement nesting

JavaScript клучни зборови

JavaScript reserved keywords

break	case	catch	continue	default
delete	do	else	false	finally
for	function	if	in	instanceof
new	null	return	switch	this
throw	true	try	typeof	var
void	while	with		

Keywords that are reserved but not used by JavaScript

class	const	enum	export	extends
implements	import	interface	let	package
private	protected	public	static	super
yield				

Тернарен оператор

```
<script language="JavaScript">  
If (3 > 2) {  
    alert("True");  
} else {  
    alert("False");  
}  
  
(3 > 2) ? alert("True") : alert("False");  
</script>
```

- Замена за едноставна “if/else” изјава

Dangling-else проблем

```
if ( x > 5 )
    if ( y > 5 )
        document.writeln( "<p>x and y are > 5</p>" );
else
    document.writeln( "<p>x is <= 5</p>" );
```

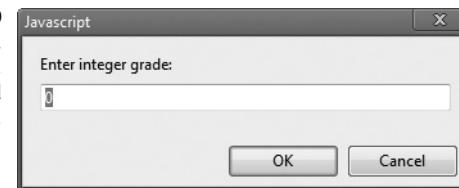
```
if ( x > 5 )
    if ( y > 5 )
        document.writeln( "<p>x and y are > 5</p>" );
else
    document.writeln( "<p>x is <= 5</p>" );
```

```
if ( x > 5 )
{
    if ( y > 5 )
        document.writeln( "<p>x and y are > 5</p>" );
}
else
    document.writeln( "<p>x is <= 5</p>" );
```

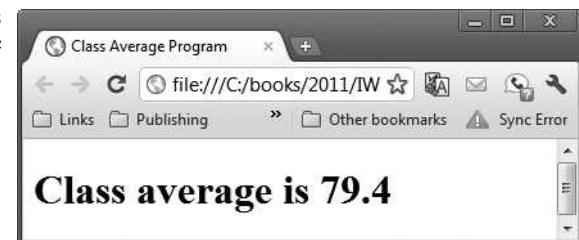
Counter-controlled repetition

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 7.7: average.html -->
4 <!-- Counter-controlled repetition to calculate a class average. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Class Average Program</title>
9     <script>
10
11       var total; // sum of grades
12       var gradeCounter; // number of grades entered
13       var grade; // grade typed by user (as a string)
14       var gradeValue; // grade value (converted to integer)
15       var average; // average of all grades
16
17       // initialization phase
18       total = 0; // clear total
19       gradeCounter = 1; // prepare to loop
20
21       // processing phase
22       while ( gradeCounter <= 10 ) // loop 10 times
23     {
24
25         // prompt for input and read grade from user
26         grade = window.prompt( "Enter integer grade:", "0" );
27
28         // convert grade from a string to an integer
29         gradeValue = parseInt( grade );
30
31         // add gradeValue to total
32         total = total + gradeValue;
33
34         // add 1 to gradeCounter
35         gradeCounter = gradeCounter + 1;
36     } // end while
37
38     // termination phase
39     average = total / 10; // calculate the average
40
41     // display average of exam grades
42     document.writeln(
43       "<h1>Class average is " + average + "</h1>" );
44
45   </script>
46   </head><body></body>
47 </html>
```

a) This dialog is displayed 10 times. User input is 100, 88, 93, 55, 68, 77, 83, 95, 73 and 62. User enters each grade and presses OK.

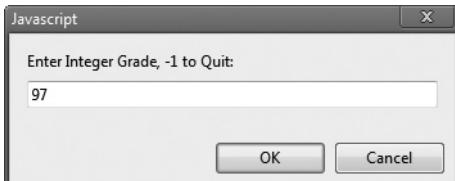


b) The class average is displayed in a web page



Sentinel-controlled repetition

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 7.9: average2.html -->
4 <!-- Sentinel-controlled repetition to calculate a class average. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Class Average Program: Sentinel-controlled Repetition</title>
9     <script>
10
11       var total; // sum of grades
12       var gradeCounter; // number of grades entered
13       var grade; // grade typed by user (as a string)
14       var gradeValue; // grade value (converted to integer)
15       var average; // average of all grades
16
17       // initialization phase
18       total = 0; // clear total
19       gradeCounter = 0; // prepare to loop
20
21       // processing phase
22       // prompt for input and read grade from user
23       grade = window.prompt(
24         "Enter Integer Grade, -1 to Quit:", "0" );
25
26       // convert grade from a string to an integer
27       gradeValue = parseInt( grade );
28
```



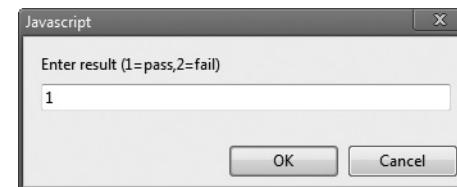
```
29   while ( gradeValue != -1 )
30   {
31     // add gradeValue to total
32     total = total + gradeValue;
33
34     // add 1 to gradeCounter
35     gradeCounter = gradeCounter + 1;
36
37     // prompt for input and read grade from user
38     grade = window.prompt(
39       "Enter Integer Grade, -1 to Quit:", "0" );
40
41     // convert grade from a string to an integer
42     gradeValue = parseInt( grade );
43   } // end while
44
45   // termination phase
46   if ( gradeCounter != 0 )
47   {
48     average = total / gradeCounter;
49
50     // display average of exam grades
51     document.writeln(
52       "<h1>Class average is " + average + "</h1>" );
53   } // end if
54
55   else
56     document.writeln( "<p>No grades were entered</p>" );
57
58   </script>
59 </head><body></body>
</html>
```



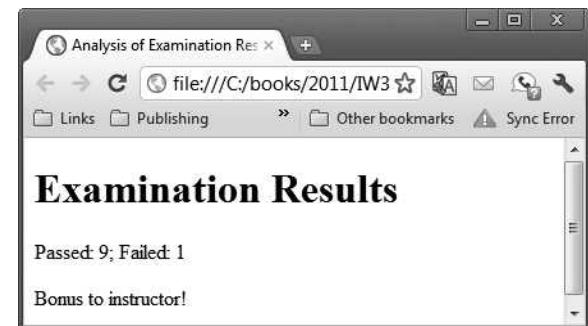
Вгнездени контролни изјави

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 7.11: analysis.html -->
4 <!-- Examination-results calculation. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Analysis of Examination Results</title>
9     <script>
10
11       // initializing variables in declarations
12       var passes = 0; // number of passes
13       var failures = 0; // number of failures
14       var student = 1; // student counter
15       var result: // an exam result
16
17       // process 10 students; counter-controlled loop
18       while ( student <= 10 )
19       {
20         result = window.prompt( "Enter result (1=pass,2=fail)", "0" );
21
22         if ( result == "1" )
23           passes = passes + 1;
24         else
25           failures = failures + 1;
26
27         student = student + 1;
28     } // end while
29
30     // termination phase
31     document.writeln( "<h1>Examination Results</h1>" );
32     document.writeln( "<p>Passed: " + passes +
33                     " Failed: " + failures + "</p>" );
34
35     if ( passes > 8 )
36       document.writeln( "<p>Bonus to instructor!</p>" );
37
38   </script>
39   </head><body></body>
40 </html>
```

- a) This dialog is displayed 10 times. User input is 1, 2, 1, 1, 1, 1, 1, 1 and 1.



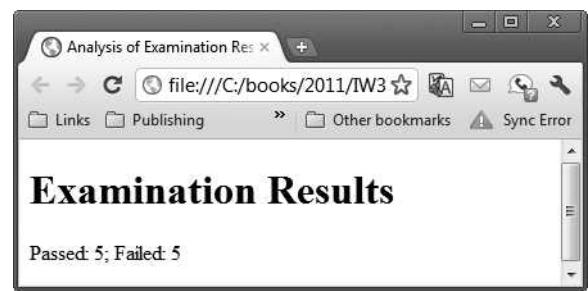
- b) Nine students passed and one failed, therefore "Bonus to instructor!" is printed.



- c) This dialog is displayed 10 times. User input is 1, 2, 1, 2, 2, 1 and 1.



- d) Five students passed and five failed, so no bonus is paid to the instructor.



Оператори

Assignment operator	Initial value of variable	Sample expression	Explanation	Assigns
<code>+=</code>	<code>c = 3</code>	<code>c += 7</code>	<code>c = c + 7</code>	10 to c
<code>-=</code>	<code>d = 5</code>	<code>d -= 4</code>	<code>d = d - 4</code>	1 to d
<code>*=</code>	<code>e = 4</code>	<code>e *= 5</code>	<code>e = e * 5</code>	20 to e
<code>/=</code>	<code>f = 6</code>	<code>f /= 3</code>	<code>f = f / 3</code>	2 to f
<code>%=</code>	<code>g = 12</code>	<code>g %= 9</code>	<code>g = g % 9</code>	3 to g

Operator	Example	Called	Explanation
<code>++</code>	<code>++a</code>	preincrement	Increment a by 1, then use the new value of a in the expression in which a resides.
<code>++</code>	<code>a++</code>	postincrement	Use the current value of a in the expression in which a resides, then increment a by 1.
<code>--</code>	<code>--b</code>	predecrement	Decrement b by 1, then use the new value of b in the expression in which b resides.
<code>--</code>	<code>b--</code>	postdecrement	Use the current value of b in the expression in which b resides, then decrement b by 1.

Пример за incrementing

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 7.14: increment.html --&gt;
4  <!-- Preincrementing and Postincrementing. --&gt;
5  &lt;html&gt;
6      &lt;head&gt;
7          &lt;meta charset = "utf-8"&gt;
8          &lt;title&gt;Preincrementing and Postincrementing&lt;/title&gt;
9          &lt;script&gt;
10
11          var c;
12
13          c = 5;
14          document.writeln( "&lt;h3&gt;Postincrementing&lt;/h3&gt;" );
15          document.writeln( "&lt;p&gt;" + c ); // prints 5
16          // prints 5 then increments
17          document.writeln( " " + c++ );
18          document.writeln( " " + c + "&lt;/p&gt;" ); // prints 6
19
20          c = 5;
21          document.writeln( "&lt;h3&gt;Preincrementing&lt;/h3&gt;" );
22          document.writeln( "&lt;p&gt;" + c ); // prints 5
23          // increments then prints 6
24          document.writeln( " " + ++c );
25          document.writeln( " " + c + "&lt;/p&gt;" ); // prints 6
26
27      &lt;/script&gt;
28      &lt;/head&gt;&lt;body&gt;&lt;/body&gt;
29  &lt;/html&gt;</pre>
```



while counter

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 8.1: WhileCounter.html -->
4 <!-- Counter-controlled repetition. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Counter-Controlled Repetition</title>
9     <script>
10
11       var counter = 1; // initialization
12
13       while ( counter <= 7 ) // repetition condition
14     {
15         document.writeln( "<p style = 'font-size: " +
16                           counter + "ex'>HTML5 font size " + counter + "ex</p>" );
17         ++counter; // increment
18     } //end while
19
20   </script>
21 </head><body></body>
22 </html>
```



HTML5 font size 1ex

HTML5 font size 2ex

HTML5 font size 3ex

HTML5 font size 4ex

HTML5 font size 5ex

HTML5 font size 6ex

HTML5 font size 7ex

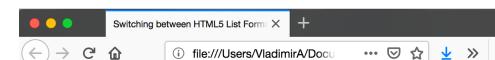
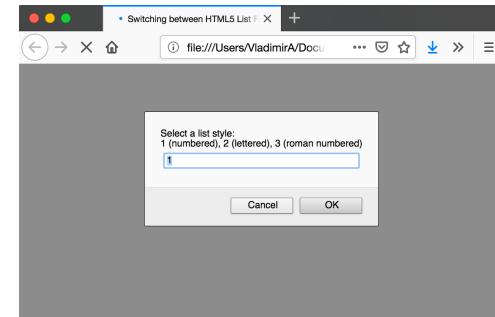
for repetition

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 8.2: ForCounter.html -->
4  <!-- Counter-controlled repetition with the for statement. -->
5  <html>
6      <head>
7          <meta charset="utf-8">
8          <title>Counter-Controlled Repetition</title>
9          <script>
10
11             // Initialization, repetition condition and
12             // incrementing are all included in the for
13             // statement header.
14             for ( var counter = 1; counter <= 7; ++counter )
15                 document.writeln( "<p style = 'font-size: " +
16                     counter + "ex'>HTML5 font size " + counter + "ex</p>" );
17
18             </script>
19         </head><body></body>
20     </html>
```

switch

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 8.7: SwitchTest.html -->
4 <!-- Using the switch multiple-selection statement. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Switching between HTML5 List Formats</title>
9     <script>
10
11       var choice; // user's choice
12       var startTag; // starting list item tag
13       var endTag; // ending list item tag
14       var validInput = true; // true if input valid else false
15       var listType; // type of list as a string
16
17       choice = window.prompt( "Select a list style:\n" +
18         "1 (numbered), 2 (lettered), 3 (roman numbered)", "1" );
19
20       switch ( choice )
21     {
22       case "1":
23         startTag = "<ol>";
24         endTag = "</ol>";
25         listType = "<h1>Numbered List</h1>";
26         break;
27       case "2":
28         startTag = "<ol style = 'list-style-type: upper-alpha'>";
29         endTag = "</ol>";
30         listType = "<h1>Lettered List</h1>";
31         break;
32       case "3":
33         startTag = "<ol style = 'list-style-type: upper-roman'>";
34         endTag = "</ol>";
35         listType = "<h1>Roman Numbered List</h1>";
36         break;
37       default:
38         validInput = false;
39         break;
40     } //end switch
41
42     if ( validInput === true )
43   {
```

```
44       document.writeln( listType + startTag );
45
46       for ( var i = 1; i <= 3; ++i )
47         document.writeln( "<li>List item " + i + "</li>" );
48
49       document.writeln( endTag );
50   } //end if
51   else
52     document.writeln( "Invalid choice: " + choice );
53
54   </script>
55 </head><body></body>
56 </html>
```

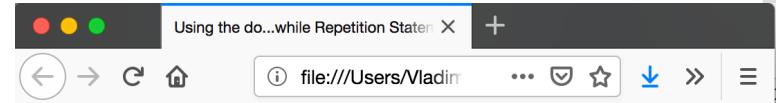


Lettered List

- A. List item 1
- B. List item 2
- C. List item 3

do ... while

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 8.9: DoWhileTest.html -->
4 <!-- Using the do...while repetition statement. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Using the do...while Repetition Statement</title>
9     <script>
10
11       var counter = 1;
12
13       do {
14         document.writeln( "<h" + counter + ">This is " +
15           "an h" + counter + " level head" + "</h" +
16           counter + ">" );
17         ++counter;
18     } while ( counter <= 6 );
19
20   </script>
21
22   </head><body></body>
23 </html>
```



This is an h1 level head

This is an h2 level head

This is an h3 level head

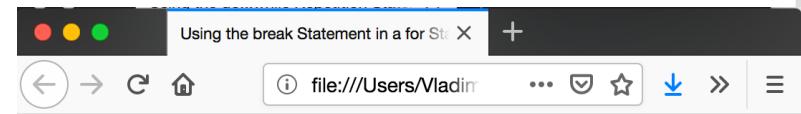
This is an h4 level head

This is an h5 level head

This is an h6 level head

break

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 8.11: BreakTest.html --&gt;
4  &lt;!-- Using the break statement in a for statement. --&gt;
5  &lt;html&gt;
6      &lt;head&gt;
7          &lt;meta charset = "utf-8"&gt;
8          &lt;title&gt;
9              Using the break Statement in a for Statement
10         &lt;/title&gt;
11         &lt;script&gt;
12
13             for ( var count = 1; count &lt;= 10; ++count )
14             {
15                 if ( count == 5 )
16                     break; // break loop only if count == 5
17
18                 document.writeln( count + " " );
19             } //end for
20
21             document.writeln(
22                 "&lt;p&gt;Broke out of loop at count = " + count + "&lt;/p&gt;" );
23
24         &lt;/script&gt;
25     &lt;/head&gt;&lt;body&gt;&lt;/body&gt;
26 &lt;/html&gt;</pre>
```



1 2 3 4

Broke out of loop at count = 5

continue

```
I  <!DOCTYPE html>
2
3  <!-- Fig. 8.12: ContinueTest.html -->
4  <!-- Using the continue statement in a for statement. -->
5  <html>
6    <head>
7      <meta charset = "utf-8">
8      <title>
9        Using the continue Statement in a for Statement
10     </title>
11
12    <script>
13
14      for ( var count = 1; count <= 10; ++count )
15      {
16        if ( count == 5 )
17          continue; // skip remaining loop code only if count == 5
18
19        document.writeln( count + " " );
20      } //end for
21
22      document.writeln( "<p>Used continue to skip printing 5</p>" );
23
24    </script>
25
26    </head><body></body>
27  </html>
```



1 2 3 4 6 7 8 9 10

Used continue to skip printing 5

Функции (1/7)

- Формат на кориснички дефинирана функција во JavaScript:

```
function function-name( parameter-list )  
{  
declarations and statements;  
}
```

- Функциите кои ги нудат веќе дефинирани објекти во JavaScript се нарекуваат **методи**
 - `Math.max`, `Math.pow` ,...

Функции (2/7)

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 9.2: SquareInt.html -->
4  <!-- Programmer-defined function square. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>A Programmer-Defined square Function</title>
9          <style type = "text/css">
10             p { margin: 0; }
11         </style>
12         <script>
13
14             document.writeln( "<h1>Square the numbers from 1 to 10</h1>" );
15
16             // square the numbers from 1 to 10
17             for ( var x = 1; x <= 10; ++x )
18                 document.writeln( "<p>The square of " + x + " is " +
19                     square( x ) + "</p>" );
20
21             // The following square function definition's body is executed
22             // only when the function is called explicitly as in line 19
23             function square( y )
24             {
25                 return y * y;
26             } // end function square
27
28         </script>
29     </head><body></body> <!-- empty body element -->
30 </html>
```

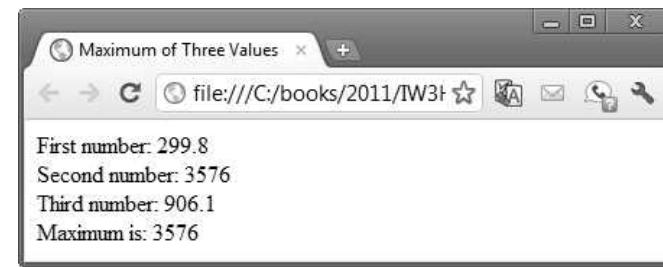
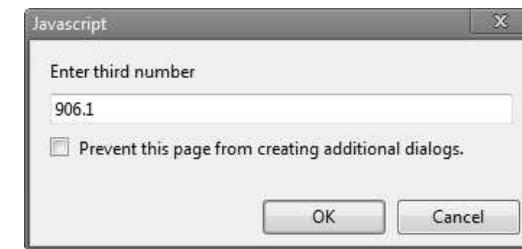
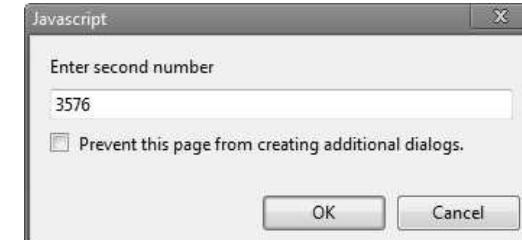
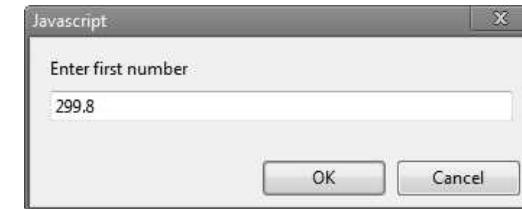


Square the numbers from 1 to 10

The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
The square of 6 is 36
The square of 7 is 49
The square of 8 is 64
The square of 9 is 81
The square of 10 is 100

Функции (3/7)

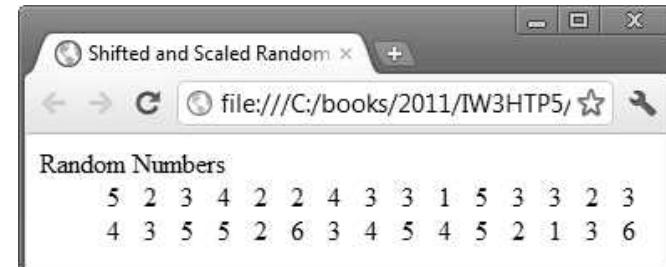
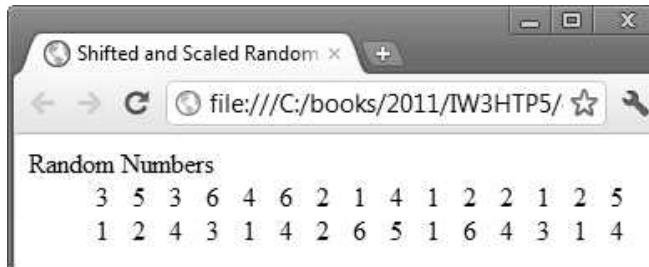
```
1 <!DOCTYPE html>
2
3 <!-- Fig. 9.3: maximum.html -->
4 <!-- Programmer-Defined maximum function. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Maximum of Three Values</title>
9     <style type = "text/css">
10       p { margin: 0; }
11     </style>
12     <script>
13
14       var input1 = window.prompt( "Enter first number", "0" );
15       var input2 = window.prompt( "Enter second number", "0" );
16       var input3 = window.prompt( "Enter third number", "0" );
17
18       var value1 = parseFloat( input1 );
19       var value2 = parseFloat( input2 );
20       var value3 = parseFloat( input3 );
21
22       var maxValue = maximum( value1, value2, value3 );
23
24       document.writeln( "<p>First number: " + value1 + "</p>" +
25                     "<p>Second number: " + value2 + "</p>" +
26                     "<p>Third number: " + value3 + "</p>" +
27                     "<p>Maximum is: " + maxValue + "</p>" );
28
29 // maximum function definition (called from line 22)
30 function maximum( x, y, z )
31 {
32   return Math.max( x, Math.max( y, z ) );
33 } // end function maximum
34
35   </script>
36 </head><body></body>
37 </html>
```



Функции (4/7)

- Random number generator
 - Math.random() – генерира случаен број во интервал од 0.0 до 1.0 (не вклучувајќи 1.0)
- Заокружување на целобројна вредност со Math.floor

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 9.4: RandomInt.html -->
4 <!-- Random integers, shifting and scaling. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Shifted and Scaled Random Integers</title>
9     <style type = "text/css">
10       p, ol { margin: 0; }
11       li { display: inline; margin-right: 10px; }
12     </style>
13     <script>
14
15       var value;
16
17       document.writeln( "<p>Random Numbers</p><ol>" );
18
19       for ( var i = 1; i <= 30; ++i )
20     {
21         value = Math.floor( 1 + Math.random() * 6 );
22         document.writeln( "<li>" + value + "</li>" );
23     } // end for
24
25       document.writeln( "</ol>" );
26
27     </script>
28   </head><body></body>
29 </html>
```

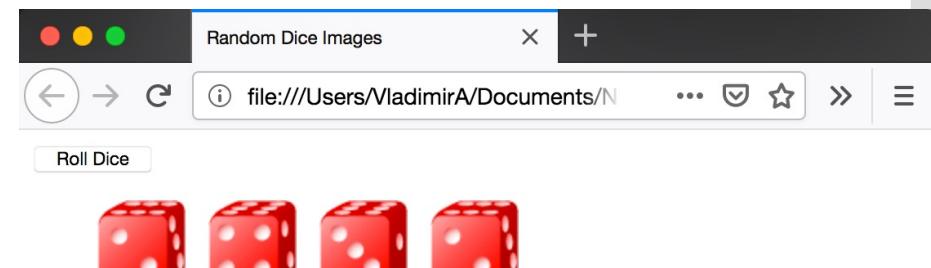
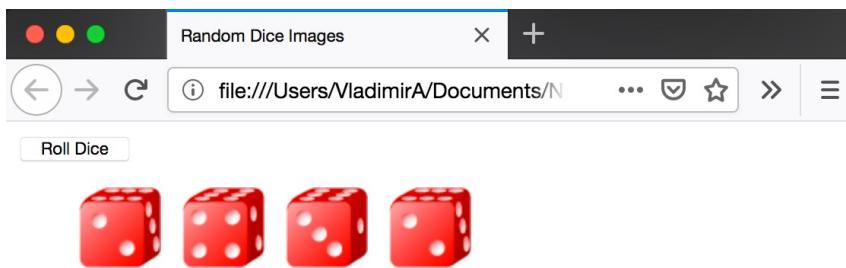


Функции (5/7)

- Random image generator
- Кориснички дефинирана скрипта која ќе прикажува комбинација од 4 коцки по случаен избор со притискање на копче “Roll Dice”
 - Код даден во RollDice.html
 - Неопходни се сликите за коцките

- Кориснички интеракции преку GUI event handling

```
52 <body>
53   <form action = "#">
54     <input id = "rollButton" type = "button" value = "Roll Dice">
55   </form>
56   <ol>
57     <li><img id = "die1" src = "blank.png" alt = "die 1 image"></li>
58     <li><img id = "die2" src = "blank.png" alt = "die 2 image"></li>
59     <li><img id = "die3" src = "blank.png" alt = "die 3 image"></li>
60     <li><img id = "die4" src = "blank.png" alt = "die 4 image"></li>
61   </ol>
62 </body>
```



Функции (6/7)

- Концепт на event-driven programming
 - Кликањето на “Roll Dice” копчето е настан (event)
 - Функцијата која се повикува за управување со настанот се нарекува event handler
- Повеќето HTML5 елементи имаат неколку различни event types
- Извршувањето на JavaScript функција кога документ завршува со вчитување во прозорецот на веб пребарувач се прави со помош на load event на window

```
49      window.addEventListener( "load", start, false );  
  
23      var button = document.getElementById( "rollButton" );  
24      button.addEventListener( "click", rollDice, false );  
25      die1Image = document.getElementById( "die1" );  
26      die2Image = document.getElementById( "die2" );  
27      die3Image = document.getElementById( "die3" );  
28      die4Image = document.getElementById( "die4" );
```

интеракција

настан

Справувач
со настан

Функции (7/7)

Die Rolling Frequencies

file:///Users/VladimirA/Documents/Nastava/

Roll Dice

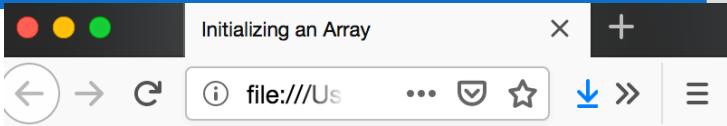
Face	Frequency	Percent
1	10	16.67
2	9	15.00
3	13	21.67
4	11	18.33
5	8	13.33
6	9	15.00

Низи (1/7)

- Низи (arrays) се податочни структури кои содржат поврзани податочни елементи
- JavaScript arrays се динамички ентитети бидејќи можат да ја менуваат својата големина
- Низите во JavaScript се објекти од типот **Array**
- Низа се креира со помош на операторот **new** (со или без аргументи)
 - **var c = new Array(12);**
- Креирањето на низа не значи и иницијализација на нејзините елементи
 - На почеток елементите во неиницијализирана низа имаат вредност **undefined**

Name of the array is c	
Position number of the element within the array c	Value of array element c[4]
c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	6453
c[11]	78

Низи (2/7)



```
1 <!DOCTYPE html>
2
3 <!-- Fig. 10.3: InitArray.html -->
4 <!-- Web page for showing the results of initializing arrays. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Initializing an Array</title>
9     <link rel = "stylesheet" type = "text/css" href = "tablestyle.css">
10    <script src = "InitArray.js"></script>
11  </head>
12  <body>
13    <div id = "output1"></div>
14    <div id = "output2"></div>
15  </body>
16 </html>
```

Array n1:

Index	Value
0	0
1	1
2	2
3	3
4	4

Array n2:

Index	Value
0	0
1	1
2	2
3	3
4	4

Низи (3/7)

```
1 // Fig. 10.4: InitArray.js
2 // Create two arrays, initialize their elements and display them
3 function start()
4 {
```

- Низа со 5 елементи
- Празна низа

- Динамично зголемување на низа

```
5 var n1 = new Array( 5 ); // allocate five-element array
6 var n2 = new Array(); // allocate empty array
7
8 // assign values to each element of array n1
9 var length = n1.length; // get array's length once before the loop
10
11 for ( var i = 0; i < length; ++i )
12 {
13     n1[ i ] = i;
14 } // end for
15
16 // create and initialize five elements in array n2
17 for ( i = 0; i < 5; ++i )
18 {
19     n2[ i ] = i;
20 } // end for
21
22 outputArray( "Array n1:", n1, document.getElementById( "output1" ) );
23 outputArray( "Array n2:", n2, document.getElementById( "output2" ) );
24 } // end function start
25
26 // output the heading followed by a two-column table
27 // containing indices and elements of "theArray"
28 function outputArray( heading, theArray, output )
29 {
30     var content = "<h2>" + heading + "</h2><table>" +
31             "<thead><th>Index</th><th>Value</th></thead><tbody>";
32
33     // output the index and value of each array element
34     var length = theArray.length; // get array's length once before loop
35
36     for ( var i = 0; i < length; ++i )
37     {
38         content += "<tr><td>" + i + "</td><td>" + theArray[ i ] +
39             "</td></tr>";
40     } // end for
41
42     content += "</tbody></table>";
43     output.innerHTML = content; // place the table in the output element
44 } // end function outputArray
45
46 window.addEventListener( "load", start, false );
```

Низи (4/7)

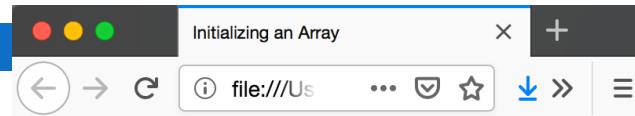
- Низите може да се иницијализираат и преку листа за иницијализација, односно елементи сместени во средни загради []
 - Големината на низата тогаш се одредува според бројот на елементи во заградите
- `var n = [10, 20, 30, 40, 50];`
- `var n = new Array(10, 20, 30, 40, 50);`
- `var n = [10, 20, , 40, 50];`
 - `(n[2]) -> undefined`

Низи (5/7)

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 10.5: InitArray2.html -->
4 <!-- Web page for showing the results of initializing arrays. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Initializing an Array</title>
9     <link rel = "stylesheet" type = "text/css" href = "tablestyle.css">
10    <script src = "InitArray2.js"></script>
11  </head>
12  <body>
13    <div id = "output1"></div>
14    <div id = "output2"></div>
15    <div id = "output3"></div>
16  </body>
17 </html>
```

```
1  // Fig. 10.6: InitArray2.js
2  // Initializing arrays with initializer lists.
3  function start()
4  {
5    // Initializer list specifies the number of elements and
6    // a value for each element.
7    var colors = new Array( "cyan", "magenta", "yellow", "black" );
8    var integers1 = [ 2, 4, 6, 8 ];
9    var integers2 = [ 2, , , 8 ];
10
11   outputArray( "Array colors contains", colors,
12               document.getElementById( "output1" ) );
13   outputArray( "Array integers1 contains", integers1,
14               document.getElementById( "output2" ) );
15   outputArray( "Array integers2 contains", integers2,
16               document.getElementById( "output3" ) );
17 } // end function start
18
19 // output the heading followed by a two-column table
20 // containing indices and elements of "theArray"
21 function outputArray( heading, theArray, output )
22 {
23   var content = "<h2>" + heading + "</h2><table>" +
24             "<thead><th>Index</th><th>Value</th></thead><tbody>";
25
26   // output the index and value of each array element
27   var length = theArray.length; // get array's length once before loop
28
29   for ( var i = 0; i < length; ++i )
30   {
31     content += "<tr><td>" + i + "</td><td>" + theArray[ i ] +
32               "</td></tr>";
33   } // end for
34
35   content += "</tbody></table>";
36   output.innerHTML = content; // place the table in the output element
37 } // end function outputArray
38
39 window.addEventListener( "load", start, false );
```

Низи (6/7)



Array colors contains

Index	Value
0	cyan
1	magenta
2	yellow
3	black

Array integers1 contains

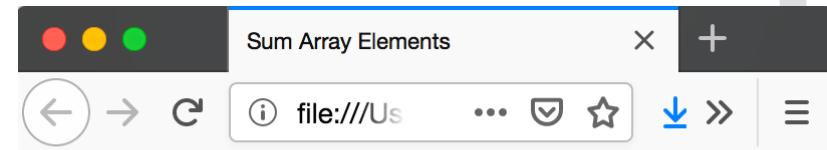
Index	Value
0	2
1	4
2	6
3	8

Array integers2 contains

Index	Value
0	2
1	undefined
2	undefined
3	8

Низи (7/7)

```
1 // Fig. 10.8: SumArray.js
2 // Summing the elements of an array with for and for...in
3 function start()
4 {
5     var theArray = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
6     var total1 = 0, total2 = 0;
7
8     // iterates through the elements of the array in order and adds
9     // each element's value to total1
10    var length = theArray.length; // get array's length once before loop
11
12    for ( var i = 0; i < length; ++i )
13    {
14        total1 += theArray[ i ];
15    } // end for
16
17    var results = "<p>Total using indices: " + total1 + "</p>";
18
19    // iterates through the elements of the array using a for... in
20    // statement to add each element's value to total2
21    for ( var element in theArray )
22    {
23        total2 += theArray[ element ];
24    } // end for
25
26    results += "<p>Total using for...in: " + total2 + "</p>";
27    document.getElementById( "output" ).innerHTML = results;
28 } // end function start
29
30 window.addEventListener( "load", start, false );
```



Total using indices: 55

Total using for...in: 55

Референцирање (1/2)

- Pass-by-value и pass-by-reference начини за предавање аргументи на функции (или методи)
- Броевите, boolean вредностите и стринговите се пренесуваат по вредност
- Низите (како и сите објекти) се пренесуваат по референца, индивидуалните елементи по вредност
 - `var hourlyTemperatures = new Array(24);`
 - `modifyArray(hourlyTemperatures);`
- JavaScript не го проверува бројот и типот на аргументи кои се пренесуваат во функцијата
 - Може да се пренесе било колкав број на аргументи

Референцирање (2/2)

```
1 // Fig. 10.14: PassArray.js
2 // Passing arrays and individual array elements to functions.
3 function start()
4 {
5     var a = [ 1, 2, 3, 4, 5 ];
6
7     // passing entire array
8     outputArray( "Original array: ", a,
9                 document.getElementById( "originalArray" ) );
10    modifyArray( a ); // array a passed by reference
11    outputArray( "Modified array: ", a,
12                 document.getElementById( "modifiedArray" ) );
13
14    // passing individual array element
15    document.getElementById( "originalElement" ).innerHTML =
16        "a[3] before modifyElement: " + a[ 3 ];
17    modifyElement( a[ 3 ] ); // array element a[3] passed by value
18    document.getElementById( "modifiedElement" ).innerHTML =
19        "a[3] after modifyElement: " + a[ 3 ];
20 } // end function start()
21
22 // outputs heading followed by the contents of "theArray"
23 function outputArray( heading, theArray, output )
24 {
25     output.innerHTML = heading + theArray.join( " " );
26 } // end function outputArray
27
28 // function that modifies the elements of an array
29 function modifyArray( theArray )
30 {
31     for ( var j in theArray )
32     {
33         theArray[ j ] *= 2;
34     } // end for
35 } // end function modifyArray
36
37 // function that modifies the value passed
38 function modifyElement( e )
39 {
40     e *= 2; // scales element e only for the duration of the function
41     document.getElementById( "inModifyElement" ).innerHTML =
42         "Value in modifyElement: " + e;
43 } // end function modifyElement
44
45 window.addEventListener( "load", start, false );
```



Effects of passing entire array by reference

Original array: 1 2 3 4 5
Modified array: 2 4 6 8 10

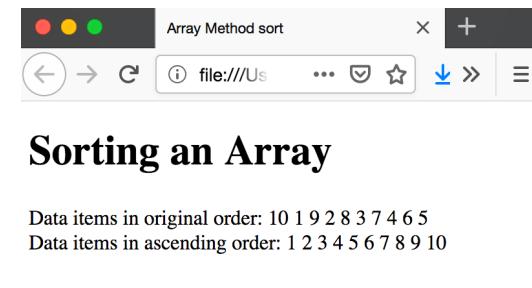
Effects of passing array element by value

a[3] before modifyElement: 8
Value in modifyElement: 16
a[3] after modifyElement: 8

Сортирање на низи

- Објектите од типот Array располагаат со вграден метод за сортирање, `sort`
- Ако `sort` е без аргументи, тогаш користи стрингови за споредба
- Како аргумент може да се пренесе име на функција која ги споредува своите два аргументи и враќа:
 - Негативна вредност, ако првиот аргумент е помал од вториот
 - Нула, ако аргументите се еднакви
 - Позитивна вредност, ако вториот аргумент е помал од првиот

```
1 // Fig. 10.16: Sort.js
2 // Sorting an array with sort.
3 function start()
4 {
5     var a = [ 10, 1, 9, 2, 8, 3, 7, 4, 6, 5 ];
6
7     outputArray( "Data items in original order: ", a,
8                 document.getElementById( "originalArray" ) );
9     a.sort( compareIntegers ); // sort the array
10    outputArray( "Data items in ascending order: ", a,
11                 document.getElementById( "sortedArray" ) );
12 } // end function start
13
14 // output the heading followed by the contents of theArray
15 function outputArray( heading, theArray, output )
16 {
17     output.innerHTML = heading + theArray.join( " " );
18 } // end function outputArray
19
20 // comparison function for use with sort
21 function compareIntegers( value1, value2 )
22 {
23     return parseInt( value1 ) - parseInt( value2 );
24 } // end function compareIntegers
25
26 window.addEventListener( "load", start, false );
```



Пребарување на низи

□ Методи:

- **indexOf** – го бара првото појавување на некоја вредност во низата
- **lastIndexOf** - го бара последното појавување на некоја вредност во низата
- Ако не е најдена вредноста, се враќа -1
- Опционо, можен е втор аргумент кај методите **indexOf** и **lastIndexOf** кој го прикажува индексот од кој треба да почне пребарувањето

```
1 // Fig. 10.18: search.js
2 // Search an array with indexOf.
3 var a = new Array( 100 ); // create an array
4
5 // fill array with even integer values from 0 to 198
6 for ( var i = 0; i < a.length; ++i )
7 {
8     a[ i ] = 2 * i;
9 } // end for
10
11 // function called when "Search" button is pressed
12 function buttonPressed()
13 {
14     // get the input text field
15     var inputVal = document.getElementById( "inputVal" );
16
17     // get the result paragraph
18     var result = document.getElementById( "result" );
19
20     // get the search key from the input text field then perform the search
21     var searchKey = parseInt( inputVal.value );
22     var element = a.indexOf( searchKey );
23
24     if ( element != -1 )
25     {
26         result.innerHTML = "Found value in element " + element;
27     } // end if
28     else
29     {
30         result.innerHTML = "Value not found";
31     } // end else
32 } // end function buttonPressed
33
34 // register searchButton's click event handler
35 function start()
36 {
37     var searchButton = document.getElementById( "searchButton" );
38     searchButton.addEventListener( "click", buttonPressed, false );
39 } // end function start
40
41 window.addEventListener( "load", start, false );
```

Повеќедимензионални низи

- Дводимензионални низи = низи со два индекси
- JavaScript не поддржува повеќедимензионални низи директно туку дозволува спецификација на низи чии елементи се исто така низи

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Column subscript
Row subscript
Array name

JavaScript објекти

- Не е објектно-ориентиран туку **објектно-базиран**
- **Едноставно се декларираат и креираат** JavaScript објекти
- Објектите **треба да имаат имена**
- Објекти се: прозори, документи, слики, табели, форми, копчиња, врски (линкови) и сл.
- Сите JavaScript вредности освен примитивните се објекти

Креирање објекти

- Креирање објектен тип побарува користење на функција за конструкција на објект (`object constructor function`)
- „**Конструктор**“ во JavaScript е **функција која доделува на `this`**
- JavaScript **нема еквивалент** на дефиниција на класа

```
function Person(name, age) { <= object constructor func
    this.name = name;
    this.age = age ;
}
var p = new Person('Annie', 23); <= креирање објект
document.write('Name: ' + p.name);
```

Функција во конструктор

- Функција може да се декларира и во конструкторот

```
function Person(name, age) {  
    this.name = name;  
    this.age = age ;  
    this.show = function () {  
        document.write ('Name: ' + this .name) ; }  
}  
  
var p = new Person('Annie', 23);  
p.show( );
```

JavaScript својства (properties)

- Својствата се вредности кои се асоцираат со објект (објектот може да се гледа како колекција од неподредени својства)
- Пристап до својствата се прави најчесто со ИменаОбјект.ИменаСвојство
- Пр. Боја на заднината се означува како **document.backgroundColor**, каде што
 - **document** е објектот
 - **backgroundColor** е својството
- **obj.name** и **obj ["name"]** **се еквивалентни**
- Својствата може да се менуваат, додаваат, бришат (некои се read-only)
 - **Ново свойство** едноставно се креира при негово дodelување на вредност
 - **Ако свойството не постои**, JavaScript ќе го креира автоматски

```
var test = new Object();
test.field1 = "Value 1"; // Create field1 property
test.field2 = 7; // Create field2 property
```

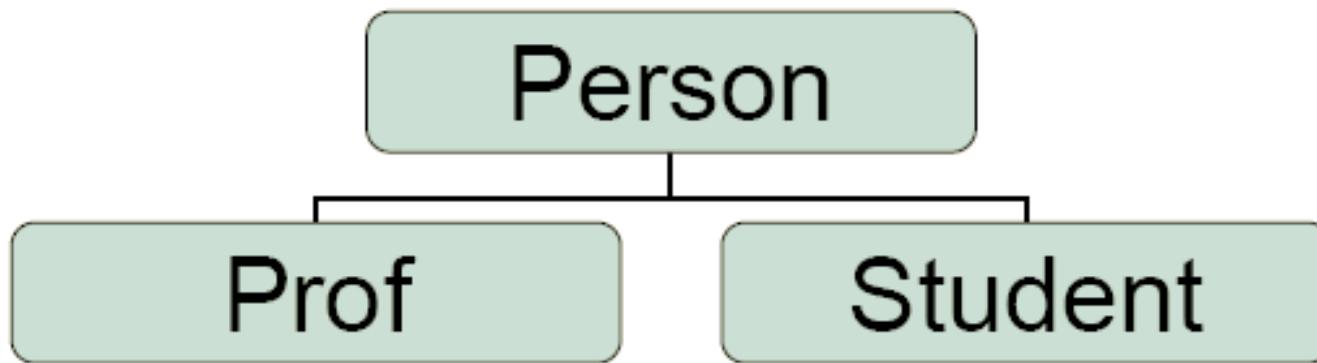
JavaScript методи (methods)

- JavaScript метод е свойство што содржи дефиниција на функција
- Специфицираат **што може да прават објектите**
 - Пр. `document.write("Hello World")`
 - `document` е објектот
 - `write` е методот
- **Пример:**

```
function show() {  
    document.write('Name: ' + this.name);  
}  
  
function Person(name, age) {  
    this.name = name;  
    this.age = age;  
    this.show = show();  
}  
  
var p = new Person('Annie' ,23);  
p.show();
```

Наследување

- **Нема** вградено наследување
- **Наследување при извршување:** се клонираат објектите и се додаваат дополнителни свойства



Вградени објекти

- **Number, Boolean, String**
- Примитивните типови **автоматски се претвораат во објекти** кога ќе се доделат на променлива
 - `var str = "abc";`
 - `str` е објект **String**
- **String** има корисни својства и функции:
length, toUpperCase, substring, link
- **Date** нема својства, но има методи за земање, поставување и работа со датум и време
- **Math** пресметува PI, SIN, COS од агол и сл.

Math објект - својства

Constant	Description	Value
Math.E	Base of a natural logarithm (e).	Approximately 2.718
Math.LN2	Natural logarithm of 2.	Approximately 0.693
Math.LN10	Natural logarithm of 10.	Approximately 2.302
Math.LOG2E	Base 2 logarithm of e .	Approximately 1.442
Math.LOG10E	Base 10 logarithm of e .	Approximately 0.434
Math.PI	π —the ratio of a circle's circumference to its diameter.	Approximately 3.141592653589793
Math.SQRT1_2	Square root of 0.5.	Approximately 0.707
Math.SQRT2	Square root of 2.0.	Approximately 1.414

Math објект - методи

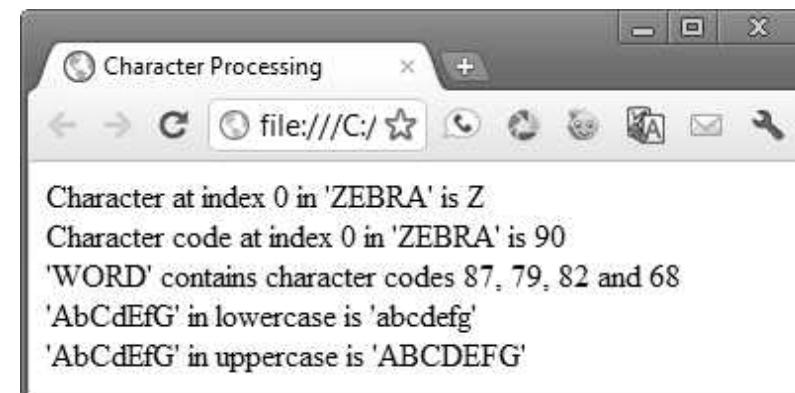
Method	Description	Examples
<code>abs(x)</code>	Absolute value of x.	<code>abs(7.2)</code> is 7.2 <code>abs(0)</code> is 0 <code>abs(-5.6)</code> is 5.6
<code>ceil(x)</code>	Rounds x to the smallest integer not less than x.	<code>ceil(9.2)</code> is 10 <code>ceil(-9.8)</code> is -9.0
<code>cos(x)</code>	Trigonometric cosine of x (x in radians).	<code>cos(0)</code> is 1
<code>exp(x)</code>	Exponential method e^x .	<code>exp(1)</code> is 2.71828 <code>exp(2)</code> is 7.38906
<code>floor(x)</code>	Rounds x to the largest integer not greater than x.	<code>floor(9.2)</code> is 9 <code>floor(-9.8)</code> is -10.0
<code>log(x)</code>	Natural logarithm of x (base e).	<code>log(2.718282)</code> is 1 <code>log(7.389056)</code> is 2
<code>max(x, y)</code>	Larger value of x and y.	<code>max(2.3, 12.7)</code> is 12.7 <code>max(-2.3, -12.7)</code> is -2.3
<code>min(x, y)</code>	Smaller value of x and y.	<code>min(2.3, 12.7)</code> is 2.3 <code>min(-2.3, -12.7)</code> is -12.7
<code>pow(x, y)</code>	x raised to power y (x^y).	<code>pow(2, 7)</code> is 128 <code>pow(9, .5)</code> is 3.0
<code>round(x)</code>	Rounds x to the closest integer.	<code>round(9.75)</code> is 10 <code>round(9.25)</code> is 9
<code>sin(x)</code>	Trigonometric sine of x (x in radians).	<code>sin(0)</code> is 0
<code>sqrt(x)</code>	Square root of x.	<code>sqrt(900)</code> is 30 <code>sqrt(9)</code> is 3
<code>tan(x)</code>	Trigonometric tangent of x (x in radians).	<code>tan(0)</code> is 0

String објект - методи

Method	Description
<code>charAt(index)</code>	Returns a string containing the character at the specified <i>index</i> . If there's no character at the <i>index</i> , <code>charAt</code> returns an empty string. The first character is located at <i>index</i> 0.
<code>charCodeAt(index)</code>	Returns the Unicode value of the character at the specified <i>index</i> , or <code>NaN</code> (not a number) if there's no character at that <i>index</i> .
<code>concat(string)</code>	Concatenates its argument to the end of the string on which the method is invoked. The original string is not modified; instead a new <code>String</code> is returned. This method is the same as adding two strings with the string-concatenation operator <code>+</code> (e.g., <code>s1.concat(s2)</code> is the same as <code>s1 + s2</code>).
<code>fromCharCode(value1, value2, ...)</code>	Converts a list of Unicode values into a string containing the corresponding characters.
<code>indexOf(substring, index)</code>	Searches for the <i>first</i> occurrence of <i>substring</i> starting from position <i>index</i> in the string that invokes the method. The method returns the starting index of <i>substring</i> in the source string or <code>-1</code> if <i>substring</i> is not found. If the <i>index</i> argument is not provided, the method begins searching from index 0 in the source string.
<code>lastIndexOf(substring, index)</code>	Searches for the <i>last</i> occurrence of <i>substring</i> starting from position <i>index</i> and searching toward the beginning of the string that invokes the method. The method returns the starting index of <i>substring</i> in the source string or <code>-1</code> if <i>substring</i> is not found. If the <i>index</i> argument is not provided, the method begins searching from the <i>end</i> of the source string.
	<code>replace(searchString, replaceString)</code>
	<code>slice(start, end)</code>
	<code>split(string)</code>
	<code>substr(start, length)</code>
	<code>substring(start, end)</code>
	<code>toLowerCase()</code>
	<code>toUpperCase()</code>

Пример (1)

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 11.4: CharacterProcessing.html -->
4 <!-- HTML5 document to demonstrate String methods charAt, charCodeAt,
5     fromCharCode, toLowerCase and toUpperCase. -->
6 <html>
7   <head>
8     <meta charset = "utf-8">
9     <title>Character Processing</title>
10    <link rel = "stylesheet" type = "text/css" href = "style.css">
11    <script src = "CharacterProcessing.js"></script>
12  </head>
13  <body>
14    <div id = "results"></div>
15  </body>
16 </html>
```



Пример (2)

```
1 // Fig. 11.5: CharacterProcessing.js
2 // String methods charAt, charCodeAt, fromCharCode,
3 // toLowerCase and toUpperCase.
4 function start()
5 {
6     var s = "ZEBRA";
7     var s2 = "AbCdEfG";
8     var result = "";
9
10    result = "<p>Character at index 0 in '" + s + "' is " +
11        s.charAt( 0 ) + "</p>";
12    result += "<p>Character code at index 0 in '" + s + "' is " +
13        s.charCodeAt( 0 ) + "</p>";
14
15    result += "<p>'" + String.fromCharCode( 87, 79, 82, 68 ) +
16        "' contains character codes 87, 79, 82 and 68</p>";
17
18    result += "<p>'" + s2 + "' in lowercase is '" +
19        s2.toLowerCase() + "'</p>";
20    result += "<p>'" + s2 + "' in uppercase is '" +
21        s2.toUpperCase() + "'</p>";
22
23    document.getElementById( "results" ).innerHTML = result;
24 } // end function start
25
26 window.addEventListener( "load", start, false );
```

Date објект - методи (1/4)

Method	Description
getDate()	Returns a number from 1 to 31 representing the day of the month in local time or UTC.
getUTCDate()	
getDay()	Returns a number from 0 (Sunday) to 6 (Saturday) representing the day of the week in local time or UTC.
getUTCDay()	
getFullYear()	Returns the year as a four-digit number in local time or UTC.
getUTCFullYear()	
getHours()	Returns a number from 0 to 23 representing hours since midnight in local time or UTC.
getUTCHours()	
getMilliseconds()	Returns a number from 0 to 999 representing the number of milliseconds in local time or UTC, respectively. The time is stored in hours, minutes, seconds and milliseconds.
getUTCMilliseconds()	
getMinutes()	Returns a number from 0 to 59 representing the minutes for the time in local time or UTC.
getUTCMinutes()	
getMonth()	Returns a number from 0 (January) to 11 (December) representing the month in local time or UTC.
getUTCMonth()	
getSeconds()	Returns a number from 0 to 59 representing the seconds for the time in local time or UTC.
getUTCSeconds()	

Date објект – методи (2/4)

<code>getTime()</code>	Returns the number of milliseconds between January 1, 1970, and the time in the Date object.
<code>getTimezoneOffset()</code>	Returns the difference in minutes between the current time on the local computer and UTC (Coordinated Universal Time).
<code> setDate(val)</code> <code>setUTCDate(val)</code>	Sets the day of the month (1 to 31) in local time or UTC.
<code>setFullYear(y, m, d)</code> <code>setUTCFullYear(y, m, d)</code>	Sets the year in local time or UTC. The second and third arguments representing the month and the date are optional. If an optional argument is not specified, the current value in the Date object is used.
<code>setHours(h, m, s, ms)</code> <code>setUTCHours(h, m, s, ms)</code>	Sets the hour in local time or UTC. The second, third and fourth arguments, representing the minutes, seconds and milliseconds, are optional. If an optional argument is not specified, the current value in the Date object is used.
<code>setMilliseconds(ms)</code> <code>setUTCMilliseconds(ms)</code>	Sets the number of milliseconds in local time or UTC.

Date објект – методи (3/4)

<code>setMinutes(m, s, ms)</code>	Sets the minute in local time or UTC. The second and third arguments, representing the seconds and milliseconds, are optional. If an optional argument is not specified, the current value in the Date object is used.
<code>setMonth(m, d)</code>	Sets the month in local time or UTC. The second argument, representing the date, is optional. If the optional argument is not specified, the current date value in the Date object is used.
<code>setSeconds(s, ms)</code>	Sets the seconds in local time or UTC. The second argument, representing the milliseconds, is optional. If this argument is not specified, the current milliseconds value in the Date object is used.
<code>setTime(ms)</code>	Sets the time based on its argument—the number of elapsed milliseconds since January 1, 1970.
<code>toLocaleString()</code>	Returns a string representation of the date and time in a form specific to the computer's locale. For example, September 13, 2007, at 3:42:22 PM is represented as <i>09/13/07 15:47:22</i> in the United States and <i>13/09/07 15:47:22</i> in Europe.

Date објект – методи (4/4)

`toUTCString()`

Returns a string representation of the date and time in the form: *15 Sep 2007 15:47:22 UTC*.

`toString()`

Returns a string representation of the date and time in a form specific to the locale of the computer (*Mon Sep 17 15:47:22 EDT 2007* in the United States).

`valueOf()`

The time in number of milliseconds since midnight, January 1, 1970. (Same as `getTime()`.)

Boolean објект

- Boolean (и Number) објектот се т.н. **object wrappers за boolean true/false вредности (и броеви)**
- Кога boolean вредност е потребна, JavaScript автоматски креира boolean објект за зачувување на вредноста
- JavaScript програмерите можат да креираат **boolean објекти експлицитно**

```
var b = new Boolean(booleanValue);
```

booleanValue специфицира дали boolean објектот треба да содржи true или false

 - Ако *booleanValue* е false, 0, null, Number.NaN, празен стринг ("") или не е специфициран аргумент, тогаш новиот boolean објект ќе содржи false (инаку, ќе содржи true)

Method	Description
toString()	Returns the string "true" if the value of the Boolean object is true; otherwise, returns the string "false".
valueOf()	Returns the value true if the Boolean object is true; otherwise, returns false.

Number објект

- Слично како за Boolean објект, можно е автоматско или експлицитно креирање на Number објект

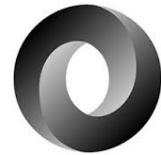
Method or property	Description
<code>toString(radix)</code>	Returns the string representation of the number. The optional <i>radix</i> argument (a number from 2 to 36) specifies the number's base. Radix 2 results in the <i>binary</i> representation, 8 in the <i>octal</i> representation, 10 in the <i>decimal</i> representation and 16 in the <i>hexadecimal</i> representation. See Appendix E, Number Systems, for an explanation of the binary, octal, decimal and hexadecimal number systems.
<code>valueOf()</code>	Returns the numeric value.
<code>Number.MAX_VALUE</code>	The largest value that can be stored in a JavaScript program.
<code>Number.MIN_VALUE</code>	The smallest value that can be stored in a JavaScript program.
<code>Number.NaN</code>	<i>Not a number</i> —a value returned from an arithmetic expression that doesn't result in a number (e.g., <code>parseInt("hello")</code>) cannot convert the string "hello" to a number, so <code>parseInt</code> would return <code>Number.NaN</code> .) To determine whether a value is <code>NaN</code> , test the result with function <code>isNaN</code> , which returns <code>true</code> if the value is <code>NaN</code> ; otherwise, it returns <code>false</code> .
<code>Number.NEGATIVE_INFINITY</code>	A value less than <code>-Number.MAX_VALUE</code> .
<code>Number.POSITIVE_INFINITY</code>	A value greater than <code>Number.MAX_VALUE</code> .

document објект

- Понуден од пребарувачот и овозможува JavaScript код да манипулира со моменталниот документ во пребарувачот
- Методи:

Method	Description
<code>getElementById(<i>id</i>)</code>	Returns the HTML5 element whose <code>id</code> attribute matches <i>id</i> .
<code>getElementsByTagName(<i>tagName</i>)</code>	Returns an array of the HTML5 elements with the specified <i>tagName</i> .

JavaScript Object Notation (JSON)



- JSON => lightweight data-interchange format
 - Едноставен начин за претставување на JavaScript објекти како стрингови
- JSON е текст
 - Секој JavaScript објект може да се конвертира во JSON и да се прати до сервер
 - Секој примен JSON од сервер може да се конвертира во JavaScript објект
- JSON е изграден на две структури:
 - Колекција од име/вредност парови (најчесто објект)
 - Секој JSON објект е претставен од листа на своства и вредности во големи загради:
 - `{ propertyName1 : value1, propertyName2 : value2 }`
 - Подредена листа на вредности (најчесто поле/низа)
 - Низите се запишуваат во JSON со средни загради:
 - `[value0, value1, value2]`
- Секоја вредност може да биде стринг, број, JSON објект, true, false или null

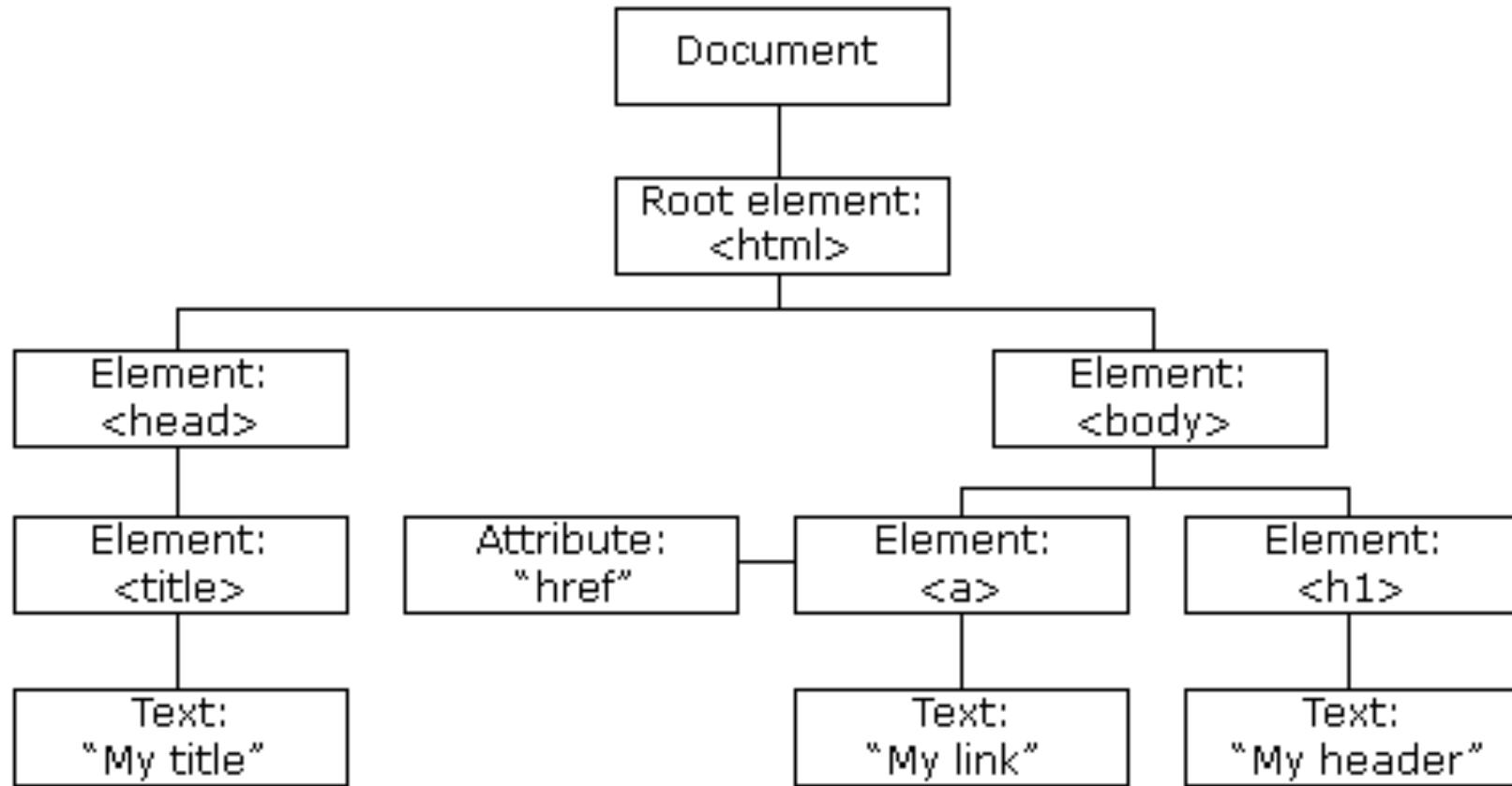
JSON пример

- Пример за JSON string: '{"name":"John", "age":30, "car":null}'
 - Дефинира објект со 3 свойства и секое име вредност
- Парсирање на JSON string со JS и пристап до податоците како до објект (let за разлика од var не дозволува промена на тип на променлива):
 - let personName = obj.name;
 - let personAge = obj.age;
- Конверзија на JSON string во JS object / JS низа: JSON.parse()
 - const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
- Конверзија на објект / низа во JSON string: JSON.stringify();
 - const obj = {name: "John", age: 30, city: "New York"};
 - const myJSON = JSON.stringify(obj);

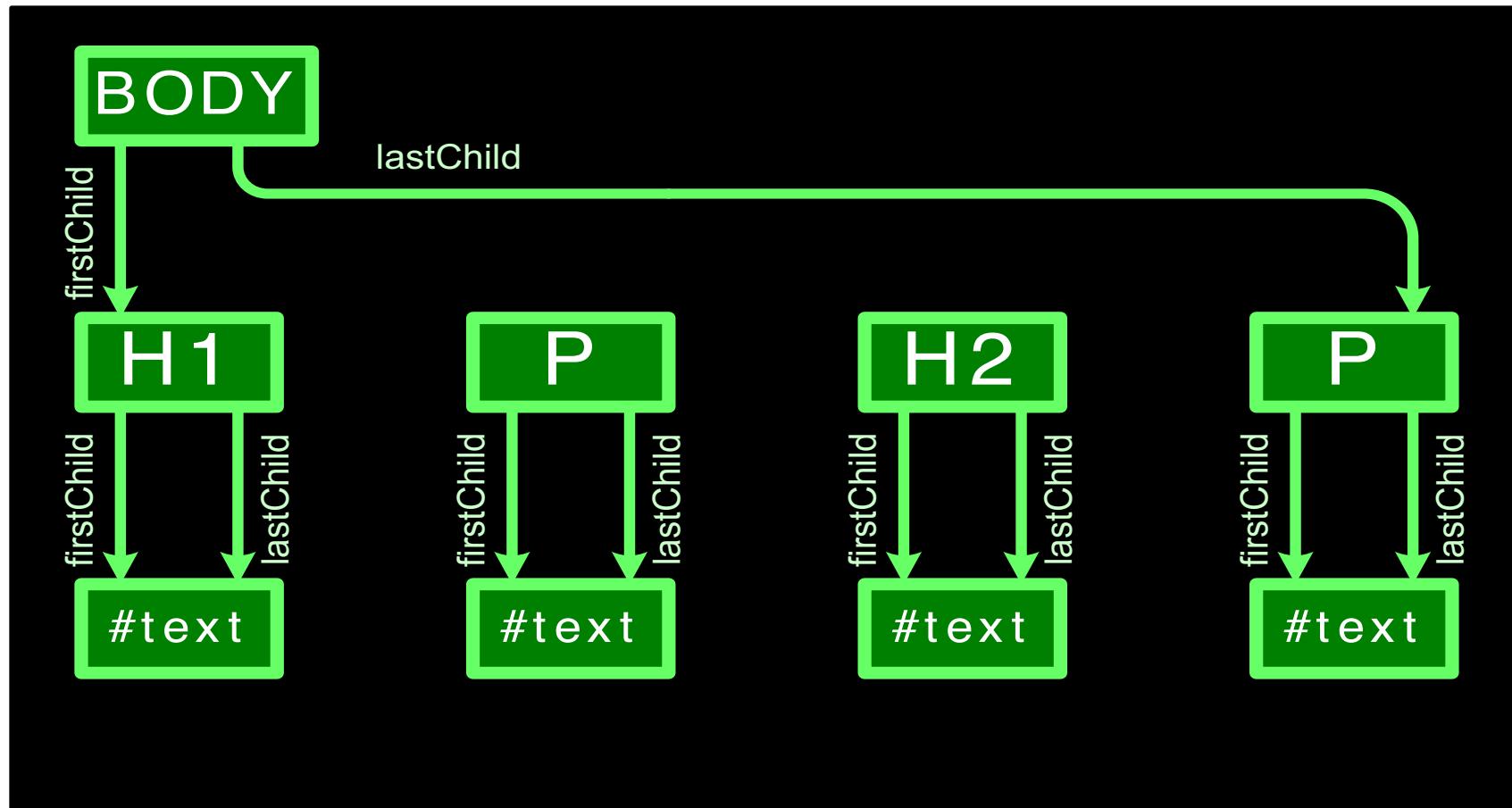
Document Object Model (DOM)

- Со помош на HTML DOM, JavaScript може да пристапи и да ги менува сите елементи во еден HTML документ
 - **Лоцирање на кој бил објект во една HTML страна** (уникатна адреса)
- **Дрво структура** на сите HTML елементи, вклучувајќи ги и атрибутите и текстот кој тие го содржат
- DOM е W3C стандард, “The DOM”, **DOM Level (1/2/3)**
- HTML DOM е стандарден објектен модел и програмирачки интерфејс за HTML и ги дефинира:
 - HTML елементите како објекти
 - Својствата на сите HTML елементи
 - Методите за пристап до сите HTML елементи
 - Настаните за сите HTML елементи
- **HTML DOM** е стандард за пристап, промена, додавање или бришење HTML елементи

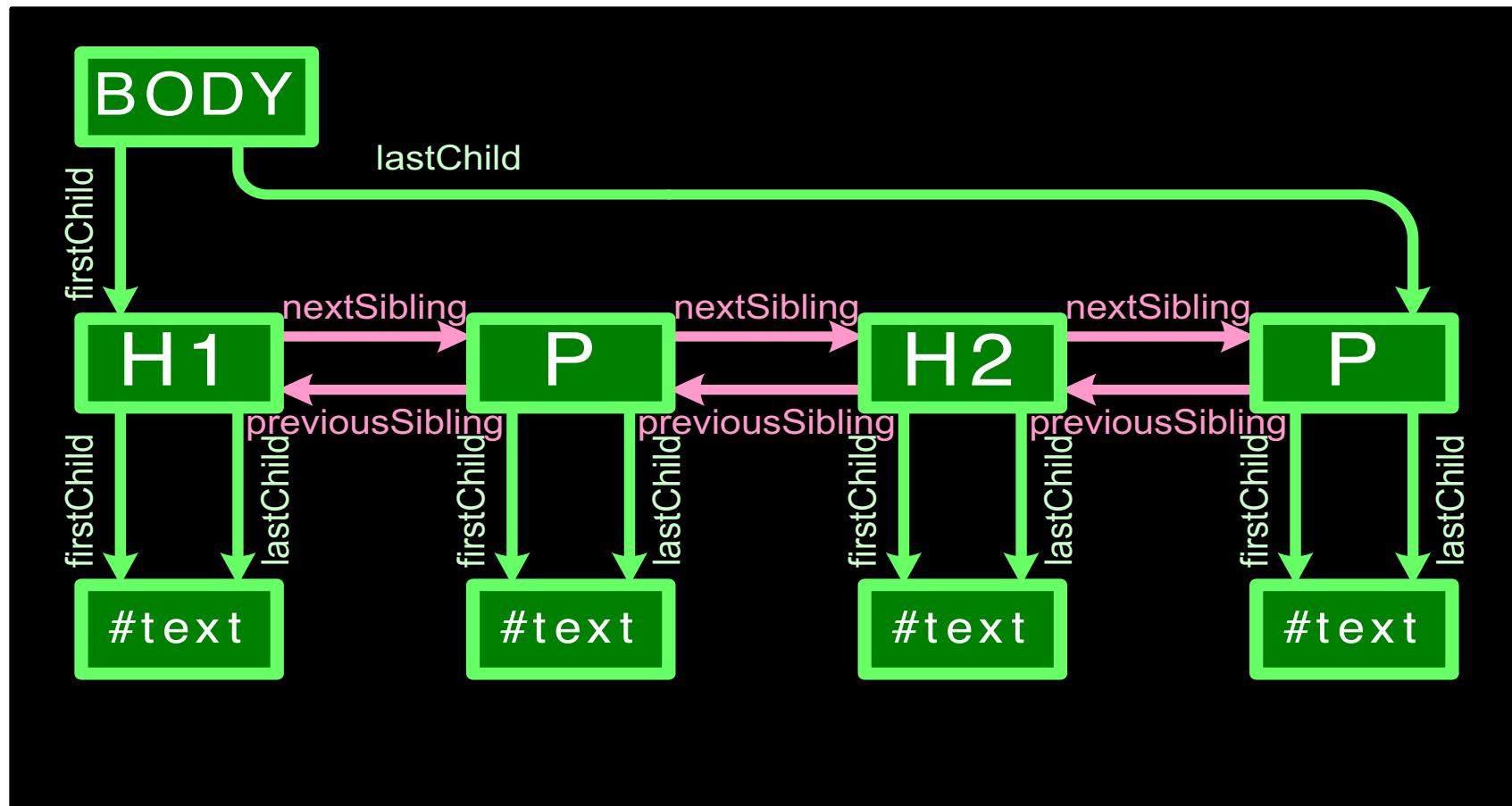
HTML DOM Tree of Objects



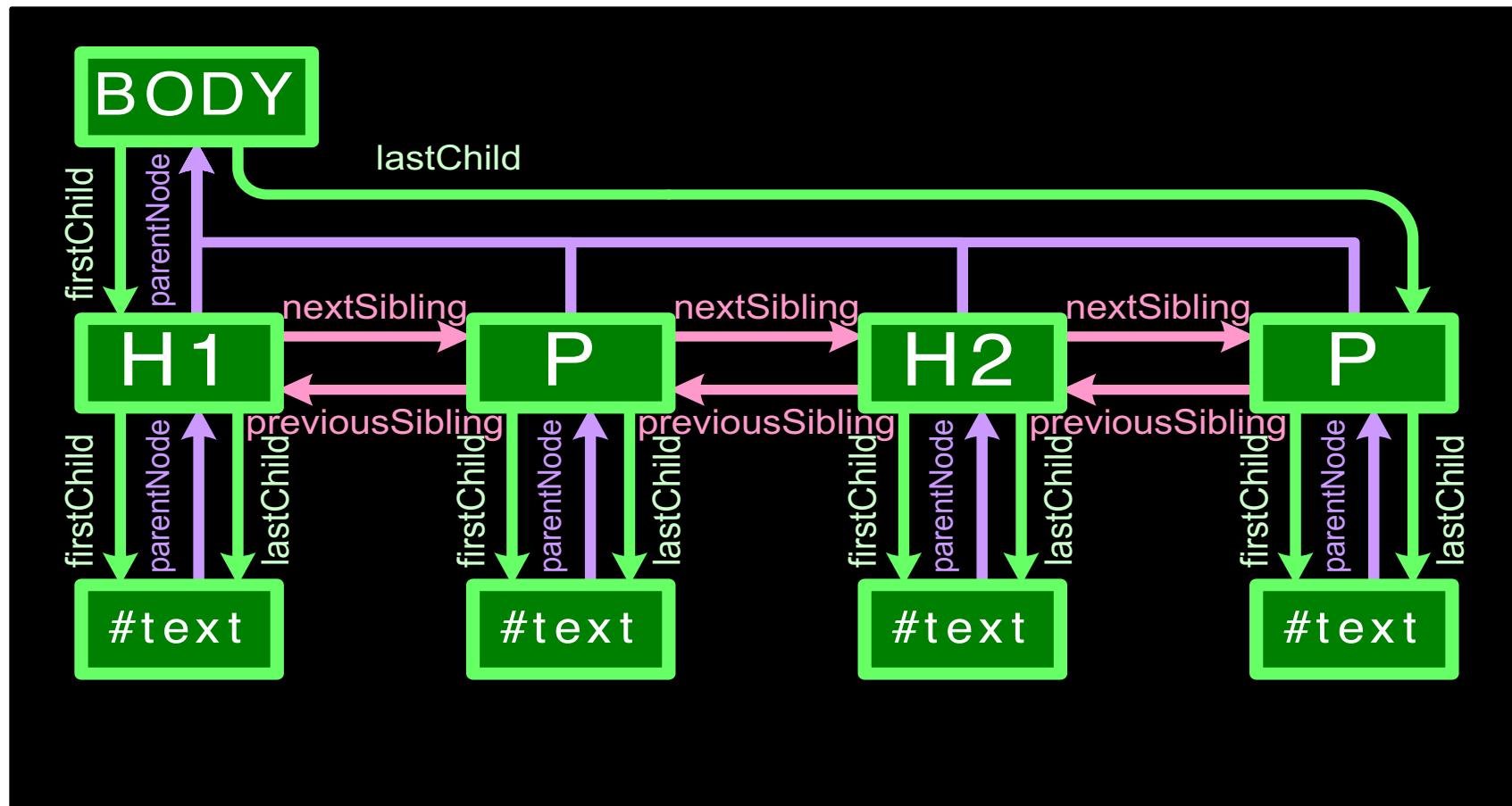
DOM navigation: child, sibling, parent



DOM navigation: child, sibling, parent

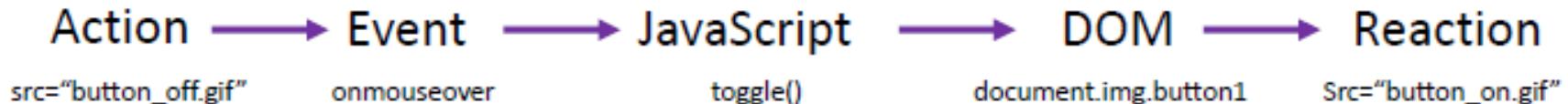


DOM navigation: child, sibling, parent



Како работи DOM?

```
<head>
  <script>
    function toggle()
    { document.getElementsByName("button1")[0].src="button_on.gif"; }
  </script>
</head>
<body>
  <a href="test.html" onmouseover="toggle()"> </a>
</body>
```



- 1) Корисникот го поставува глушецот над објектот
- 2) Event чувствува дека нешто се случило со објектот
- 3) JavaScript му кажува на објектот што да стори (Event handler)
- 4) Го одредува (лоцира) објектот на web страната
- 5) Се менува изворот на сликата на објектот

Window објект

- **Објект на највисоко ниво** во хиерархијата на објекти на еден прелистувач
- Тоа е предефинираниот објект и **се креира автоматски кога се вчитува страната**
- Тој е **JavaScript претстава на прозорец** на еден прелистувач
- Бидејќи е предефиниран објект, **може да се испушти при пишување** на патеки до објекти
 - `document.write("a test message");`
 - `window.document.write("a test message");`
- Детали за својства и методи на
https://www.w3schools.com/jsref/obj_window.asp

Свойства и методи на објект Window History

Property	Description
<code>length</code>	An integer value representing the number of links in the history object
<code>current</code>	Contains the URL of the current page
<code>next</code>	Contains the URL of the next entry in the history list
<code>previous</code>	Contains the URL of the previous entry in the history list

Method	Description
<code>back ()</code>	Sends the user to the previous page in the history list
<code>forward ()</code>	Sends the user to the next page in the history list
<code>go (x)</code>	Sends back or forward by “x” number of pages in the history list

HTML DOM document object

- Објектот `document` претставува **веб документ или страна во еден прозорец** од прелистувач
- Ако се пристапува до **повеќе сајтови** истовремено има отворено **повеќе прозорци**
 - Секој прозорец има соодветен **објект window**
 - Секој објект `window` има свој **објект document**
- **Користење на `document object` за пристап и манипулација на HTML:**
 - **По име (By name)**
`document.getElementsByTagName(name)`
 - **По ID (By ID)**
`document.getElementById(id)`
 - **По класа**
`document.getElementsByClassName(name)`

Манипулација со HTML елементи

□ Менување HTML елементи

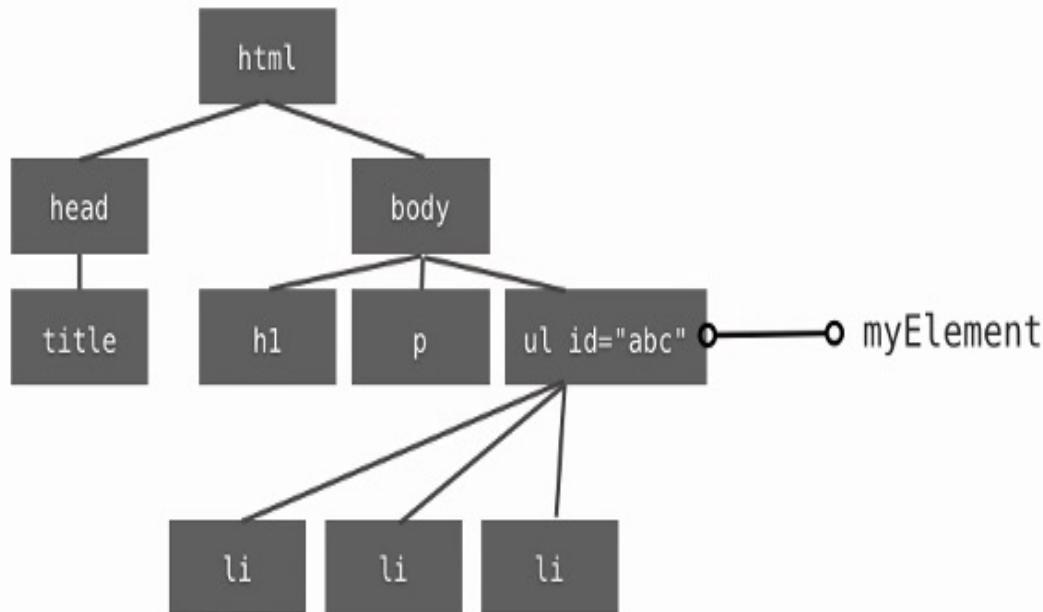
Property	Description
<code>element.innerHTML = new html content</code>	Change the inner HTML of an element
<code>element.attribute = new value</code>	Change the attribute value of an HTML element
<code>element.style.property = new style</code>	Change the style of an HTML element
Method	Description
<code>element.setAttribute(attribute, value)</code>	Change the attribute value of an HTML element

□ Додавање и бришење елементи

Method	Description
<code>document.createElement(element)</code>	Create an HTML element
<code>document.removeChild(element)</code>	Remove an HTML element
<code>document.appendChild(element)</code>	Add an HTML element
<code>document.replaceChild(new, old)</code>	Replace an HTML element
<code>document.write(text)</code>	Write into the HTML output stream

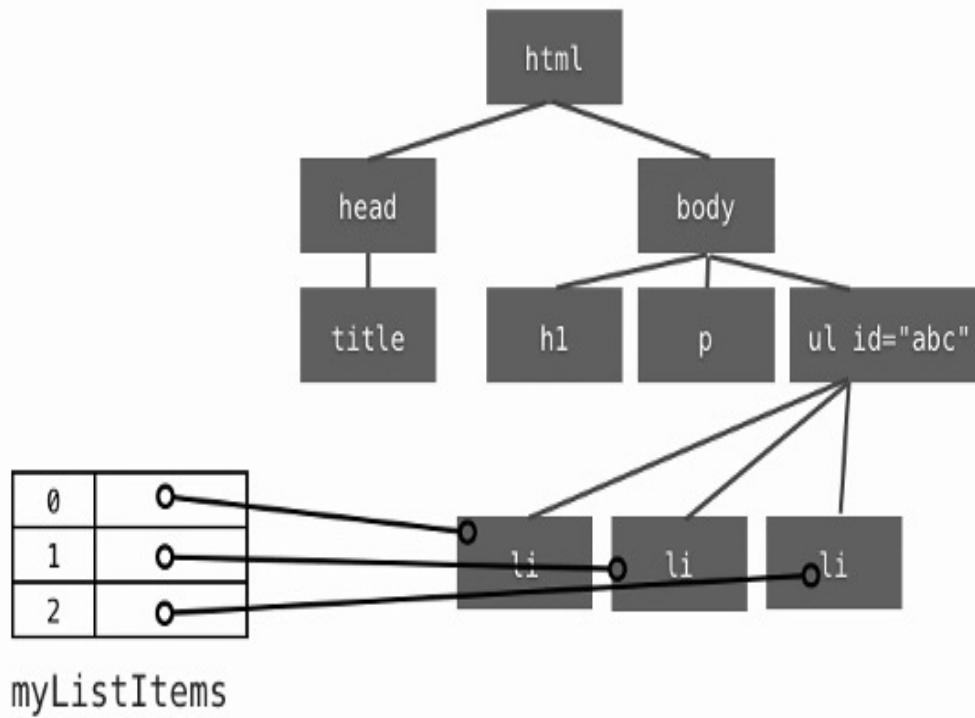
Приступување до објект преку ID

```
var myElement = document.getElementById("abc");
```



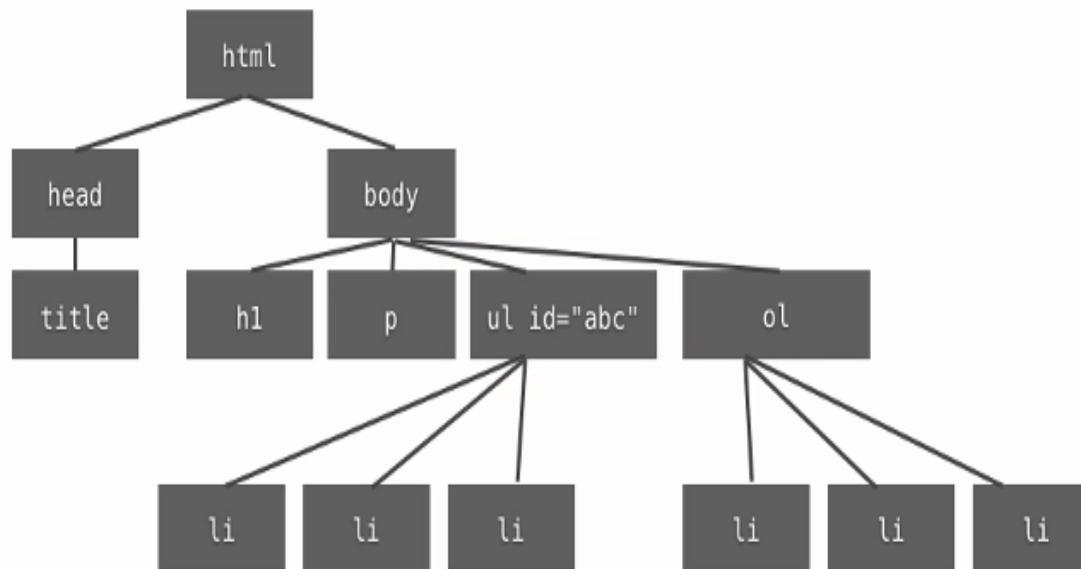
Приступување до објект преку таг

```
var myListItems = document.getElementsByTagName("li");
```



Приступување до објект преку таг

```
var myListItems = document.getElementsByTagName("li");
var myFirstList = document.getElementById("abc");
var limitedList = myFirstList.getElementsByTagName("li");
```



Промена на атрибути на ДОМ објект

name - in quotes

```
myElement.getAttribute("align");
```

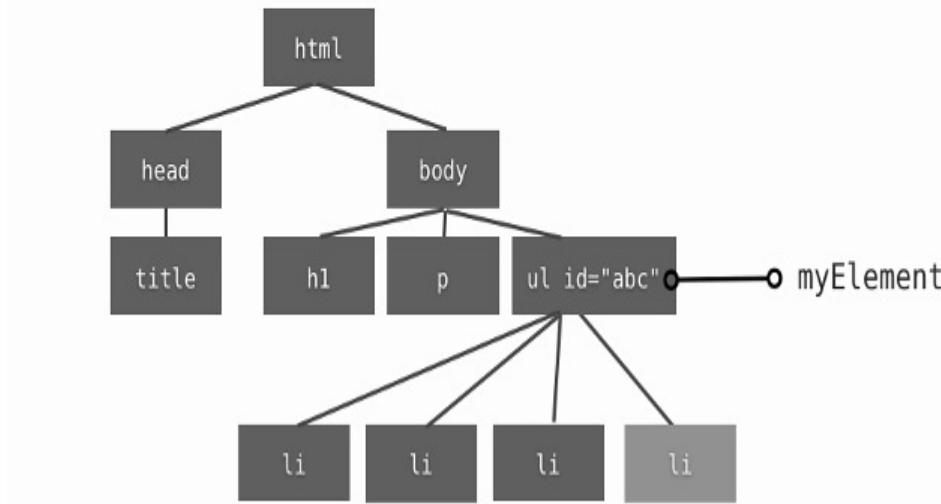
name value

```
myElement.setAttribute("align","left");
```

```
var mainContent = document.getElementById("mainContent");
mainContent.setAttribute("align","right");
```

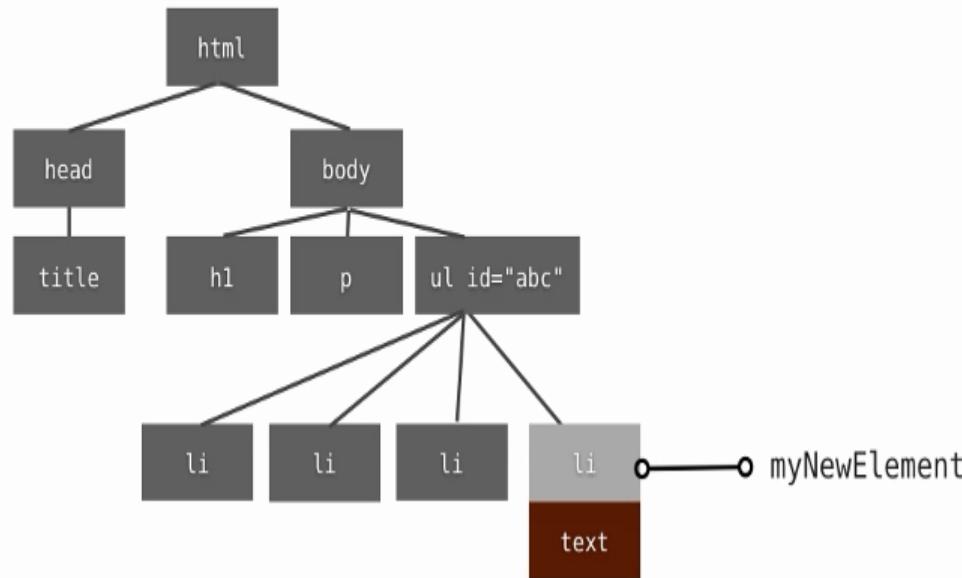
Креирање на нови ДОМ објекти

```
var myNewElement = document.createElement("li");
myElement.appendChild(myNewElement);
myNewElement.innerHTML = "New item text";
```



Креирање на текст на DOM објекти

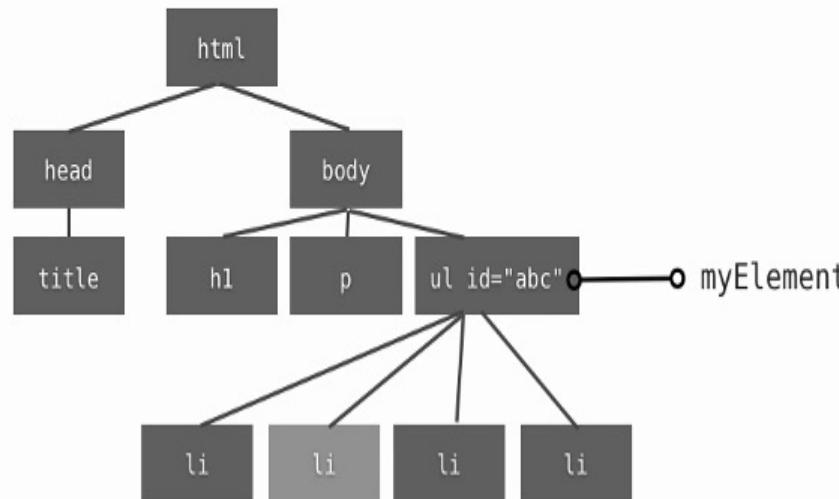
```
var myText = document.createTextNode("New list item text")
myNewElement.appendChild(myText);
```



Креирање на нови ДОМ објекти

```
parent.insertBefore(newElement, existingElement);
```

```
var myNewElement = document.createElement("li");
var secondItem = myElement.getElementsByTagName("li")[1];
myElement.insertBefore(myNewElement, secondItem);
```



Form објект (1/2)

- До објектот form се пристапува како до **својство на објектот document**
- Сите елементи од формата – form (поле за внес на текст, радио копчиња – radio buttons), понатаму се дефинираат со други објекти
- Преелистувачот креира **уникатен објект form за секоја форма во документот**
- Објектот form со име form1 **може да се пристапи како document.form1**

Form објект (2/2)

- Пристап до објекти form:
 - window.document.myForm или
 - window.document.forms[0]
- Својства
 - action, target, length, method, etc...
- Функции
 - window.document.myForm.submit();
 - window.document.myForm.reset();
- Пристап до вредности на полиња од формата
 - window.document.myForm.firstname.value

Настани (Events)

- Настаните (Events) се **акции кои се случуваат** вообично **кога корисникот ќе направи нешто**
- Пример: клик на копче, промена на текст во поле за текст, поминување со глушецот над хиперврска
- **Акциите на корисникот предизвикуваат настан**
- Настани: click, change, focus, load, mouseover, mouseout, reset, submit, select

Категории на настани

- **Настани од тастатура и глушец**
 - Настан направен од глушецот е едноставен за детектирање
- **Настани при вчитување**
 - Кога страната прв пат ќе се појави на екранот: таа се „вчитува“ (“loads”), кога ќе се изгаси: таа „заминува“ (“unloads”)
- **Настани поврзани со форми**
 - `onfocus()` – поставување на курсорот во делот за внесување текст од формата
- **Други**
 - Грешки (Errors), менување големина на прозорец (window resizing)...

Настани дефинирани од JavaScript

HTML elements	HTML tags	JavaScript defined events	Description
Link	<a>	<code>click</code> <code>dblClick</code> <code>mouseDown</code> <code>mouseUp</code> <code>mouseOver</code>	Mouse is clicked on a link Mouse is double-clicked on a link Mouse button is pressed Mouse button is released Mouse is moved over a link
Image		<code>load</code> <code>abort</code> <code>error</code>	Image is loaded into a browser Image loading is abandoned An error occurs during the image loading
Area	<area>	<code>mouseOver</code> <code>mouseOut</code> <code>dblClick</code>	The mouse is moved over an image map area The mouse is moved from image map to outside The mouse is double-clicked on an image map
Form	<form>	<code>submit</code> <code>reset</code>	The user submits a form The user refreshes a form
...

Справувачи со настани (1/2)

- Справувач со настан = код што се извршува како одговор на специфичен настан
 - Имињата на справувачите со настани се многу слични на имињата на настаните со кои управуваат
 - На пр. справувач со настан “click” е “onclick”
- Синтакса <HTMLtag eventhandler=“JavaScript Code”>
- Може да се користат справувачи со настани како onchange и onclick за да се направат скриптите динамички (да реагираат на настани)

```
<input type="button" onclick="javascript:doButton()">  
<select onchange="javascript:doChange()">  
<a onclick="javascript:doSomething()"> </a>  
<form onsubmit="javascript:validate()">  
<body onload="javascript:init()">
```

Справувачи со настани (2/2)

Event Handlers	Description
onchange	The value of the text field, textarea, or a drop down list is modified
onclick	A link, an image or a form element is clicked once
ondblclick	The element is double-clicked
onmousedown	The user presses the mouse button
onload	A document or an image is loaded
onsubmit	A user submits a form
onreset	The form is reset
onunload	The user closes a document or a frame
onresize	A form is resized by the user

Настан load

- Постојат два модели за регистрација на справувачи со настани
 - Inline модел кој ги третира настаните како атрибути на HTML елементите
 - Inline моделот вметнува повици до JavaScript функции директно во HTML кодот
 - Следниов код означува дека JavaScript функцијата start треба да се повика кога body елементот се вчитува
 - `<body onload = "start()">`
- Традиционален модел кој доделува име на функција на својството настан на DOM јазел
 - Традиционалниот модел користи својство на објект за да специфицира справувач со настан
 - Следниов JavaScript код индицира дека функцијата start треба да се повика кога документот се вчитува

`document.onload = start;`

Реакција при клик со глувче

```
<script language="javascript" type="text/javascript">
function myfunc() {
    myWin=window.open("news.html", "myWindow",
    "width=200,height=200,toolbar=no,scrollbars=yes")
}
</script>
<form>
<input type="button" value="news" onclick="myfunc()" />
<input type="button" value="close" onclick="myWin.close()" />
</form>
```

Движење на глувчето

```
<script language="javascript" type="text/javascript">
function load1() {
    document["flower"].src = "flower1.jpg"
}
function load2() {
    document["flower"].src = "flower2.jpg"
}
</script>
</head>
<body>
<a href="test.html" onmouseover="load2()"
       onmouseout="load1()">

</a>
```

Events

Event	Description
abort	Fires when image transfer has been interrupted by user.
change	Fires when a new choice is made in a <code>select</code> element, or when a text input is changed and the element loses focus.
click	Fires when the user clicks the mouse.
dblclick	Fires when the user double clicks the mouse.
focus	Fires when a form element gets the focus.
keydown	Fires when the user pushes down a key.
keypress	Fires when the user presses then releases a key.
keyup	Fires when the user releases a key.
load	Fires when an element and all its children have loaded.
mousedown	Fires when a mouse button is pressed.
mousemove	Fires when the mouse moves.
mouseout	Fires when the mouse leaves an element.
mouseover	Fires when the mouse enters an element.
mouseup	Fires when a mouse button is released.
reset	Fires when a form resets (i.e., the user clicks a reset button).
resize	Fires when the size of an object changes (i.e., the user resizes a window or frame).
select	Fires when a text selection begins (applies to <code>input</code> or <code>textarea</code>).
submit	Fires when a form is submitted.
unload	Fires when a page is about to unload.

Домашна задача #3

- Да се направи игра "Погоди го зборот"
 - На располагање стојат 10 збора од кои по случаен избор ќе се избере еден за играње
 - Зборовите да се долги до макс 5 букви, а 2 букви да се појават на старт на играта (случајно избрани)
- Играчот внесува букви на празните места и бара проверка дали е одговорот точен или не
- Зборот мора да се погоди од макс 5 обиди
- Ако зборот се погоди во рамките на бројот на дозволени обиди тогаш да излезе ропор дека е успешна играта
- Ако зборот не се погоди во рамките на бројот на дозволени обиди тогаш да излезе ропор дека не е успешна играта
- По завршување на една игра, да се појави опција за почеток на нова игра
- Да се постават html, css и js документ на github до 10.11.2024, 23:59