

CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

Java
Script
A

CONTENTS

- Fundamental programming constructs:
 - Variable,
 - Arithmetic operators,
 - Assignment operator,
 - Input and output,
 - Arrays,
 - Loops,
 - Conditions,
 - Modules (Functions),
 - Structures, classes and objects,
 - Events and event-handling
 - Files and Databases

JAVA VS. JAVA-SCRIPT

- **Java**

- Java is an Object Oriented Programming (OOP) language created by James Gosling of Sun Microsystems.
- Java is a stand alone language and runs on a virtual machine.
- Java is compiled in a byte-code (an intermediate machine language) and produces a stand-alone executable.
- Object-oriented: (Encapsulation, inheritance, polymorphism)
- Strongly typed language.

- **Java-Script**

- JavaScript is a scripting language that was created by Netscape and was originally known as LiveScript.
- JavaScript must be part of a HTML document and runs within a browser.
- JavaScript is interpreted line-by-line by the browser.
- Object-based: Code uses built-in, extensible objects, but no classes or inheritance.

Dynamically typed language

Learn More

<http://en.wikipedia.org/wiki/JavaScript>

<http://sislands.com/coin70/week1/javajs.htm>

WHAT IS JAVA SCRIPT?

- Java script is a scripting language which is typically used to **enhance the functionality and appearance** of web pages.
- Java script is the de facto standard for client side programming for web based applications.
- Java scripts are executed by the browser, however some browsers (specifically Microsoft IE) disable this capability. Therefore, if you use IE, you need to enable the execution of java script in your browser.

JAVA SCRIPT

- The java script example creates a simple function for displaying the date, and connects it to the event handler of a button.

Example:

```
<!DOCTYPE html>
<html>
<head>
<script>
function displayDate()
{
    document.getElementById("placeholder").innerHTML=Date();
}
</script>
</head>
<body>

<h1>My First JavaScript</h1>
<p id="placeholder">the date will appear here...</p>

<button type="button" onclick="displayDate()">Display
Date</button>

</body>
</html>
```

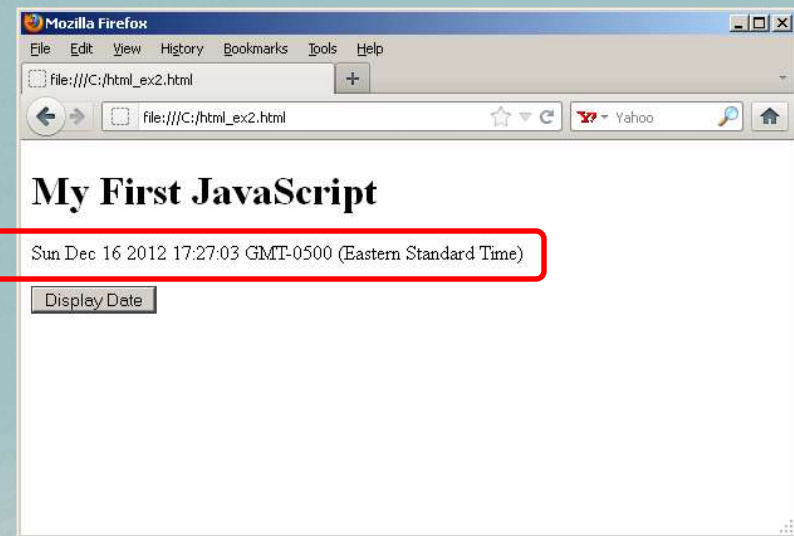
Java Script

*Called when the
button is clicked*

Event
Handler

OUTPUT

- Before Click
- After Click



JAVASCRIPT AND ITS RELATION WITH DOM VS. BOM

- **Document Object Model (DOM)**

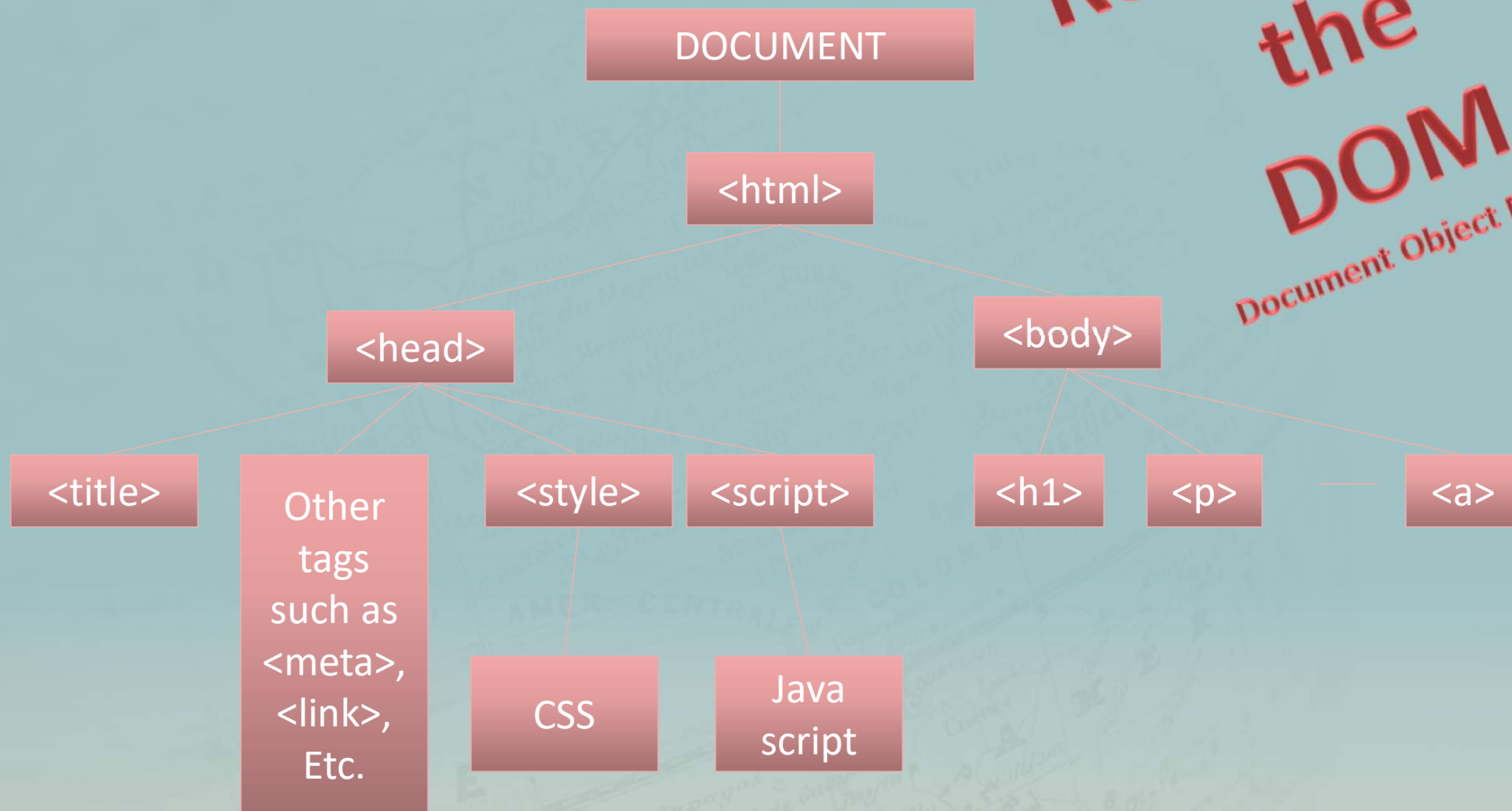
- The Document Object Model (DOM) is an application programming interface (API) for HTML as well as XML.
- The DOM organizes the entire web page as a document composed of a hierarchy of nodes like a tree structure and using the DOM API, nodes can be removed, added, and replaced.
- **DOM allows the developer to manipulate the document.**

- **Browser Object Model (BOM)**

- Browsers feature a Browser Object Model (BOM) that allows access and manipulation of the browser window. For example (browser history, location, navigator, and screen)
- Because no standards exist for the BOM, each browser has its own implementation.
- **BOM allows the developer to manipulate the browser window.**

THE STRUCTURE OF AN HTML DOCUMENT

Remember
the
DOM
Document Object Model

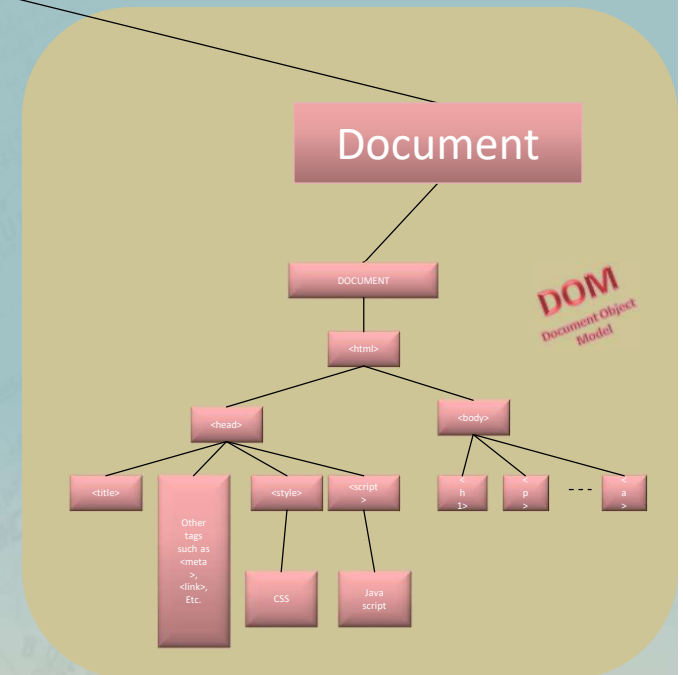
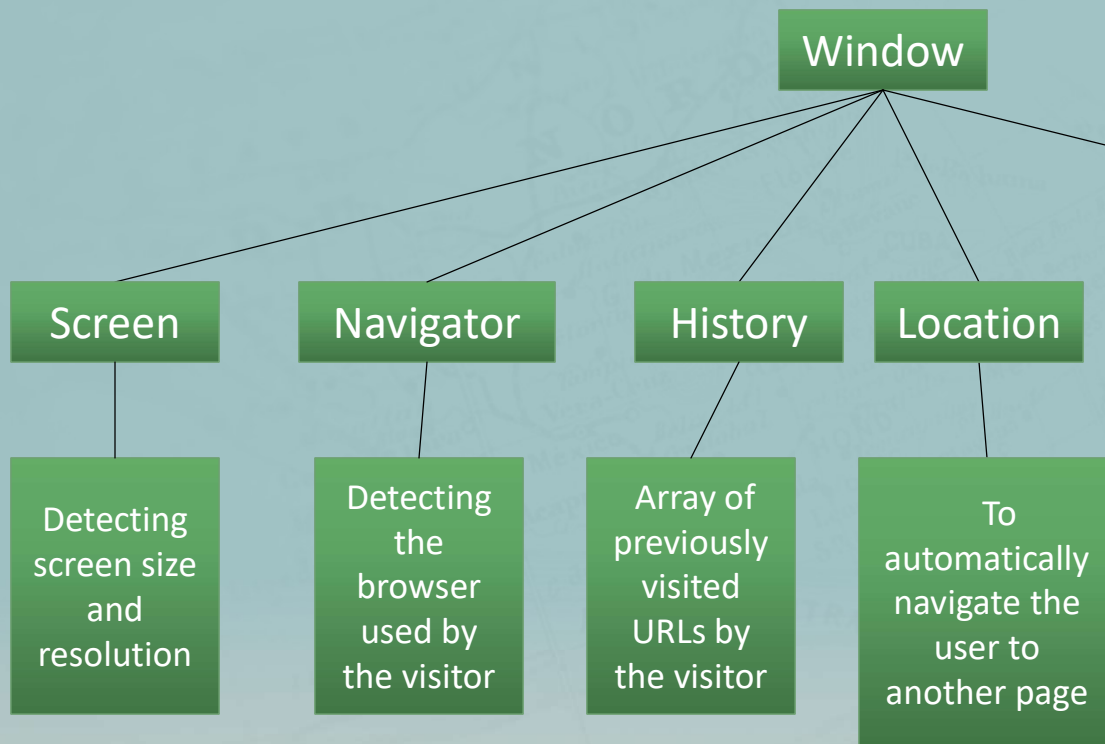


DOM IS ACTUALLY INCLUDED IN BOM

- The browser object has a **WINDOW** object, and one of Window object elements is its **DOCUMENT** object (e.g., the DOM)
- See next slide!

THE STRUCTURE OF BROWSER OBJECT

BOM
Browser Object Model



<http://www.javascriptkit.com/jsref/window.shtml>



CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

**Java
Script
B**

JAVA-SCRIPT SYNTAX

- Both Java and JavaScript syntax are based on the C and C++ language so it is easy to learn the basics.

JAVASCRIPT LANGUAGE

Basic Syntax:

- Variables
- Operators
- Conditionals
- Loops
- I/O
- Arrays
- String manipulation
- Math functions
- Error handling
- Regular expression

DOM manipulation:

- Document
- Element
- Attributes
- Style
- Events (mouse, keyboard, objects, form, drag-drop, copy-paste, etc)

BOM manipulation:

- Window
- Navigator
- Screen
- History
- Location

VARIABLES

- Variables in JavaScript are dynamically typed.
- Their type is determined only after they are assigned a value.
- Example:

```
var myName;           // No data type yet. (undefined)
var myNumber;

myName = "Cyrus";      // variable type will become string
myNumber = 5;          // variable type will become int

myNumber = 5.6;        // change the type int to float
myName = 4;            // change the type from string to int.
```

VARIABLES

```
C:\wamp\www\A340\02_HTML_HEAD_Script_5.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
02_HTML_HEAD_Script_5.html 02_HTML_HEAD_Script_5a.html
1 <!DOCTYPE html>
2 <!-- 02_HTML_HEAD_Script_5.html
3 variables
4 -->
5
6 <html>
7 <head>
8 <script type="text/javascript">
9
10     var myName;
11     var myNumber;
12     document.writeln("myName  = " + myName  + "<br />");
13     document.writeln("myNumber = " + myNumber + "<hr />");
14
15     myName  = "Cyrus";
16     myNumber = 5;
17     document.writeln("myName  = " + myName  + "<br />");
18     document.writeln("myNumber = " + myNumber + "<hr />");
19
20     myName  = 4;
21     myNumber = 5.6;
22     document.writeln("myName  = " + myName  + "<br />");
23     document.writeln("myNumber = " + myNumber + "<hr />");
24
25 </script>
26 </head>
27 <body>
28
29 </body>
30 </html>
31
length : 618 lines : 31 Ln : 21 Col : 20 Sel : 0 Dos\Windows ANSI INS
```

```
Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/wamp/www/...L_HEAD_Script_5.html +
file:///C:/warr ☆ ↺ ↻ Yahoo!
myName = undefined
myNumber = undefined
myName = Cyrus
myNumber = 5
myName = 4
myNumber = 5.6
```

ARITHMETIC OPERATORS

- Same as C/C++ operators

`+, -, *, /, %, ++, --`

OTHER OPERATORS

- Bitwise

http://www.javascriptkit.com/jsref/bitwise_operators.shtml

&, |, ^, ~, >>, <<

- Comparison

==, !=, <, >, <=, >=

http://www.javascriptkit.com/jsref/comparison_operators.shtml

- Logical

&&, ||, !

- Other:

typeof, new, delete

http://www.javascriptkit.com/jsref/other_operators.shtml

typeof

```
C:\wamp\www\A340\02_HTML_HEAD_Script_5a.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Q2_HTML_HEAD_Script_5a.html Q2_HTML_HEAD_Script_5a.html
7 <head>
8 <script type="text/javascript">
9
10     var myName;
11     var myNumber;
12     document.writeln("myName = " + myName + "<br />");
13     document.writeln("myNumber = " + myNumber + "<br />");
14     document.writeln("typeof myName = " + typeof(myName) + "<br />");
15     document.writeln("typeof myNumber = " + typeof(myNumber) + "<br />");
16
17
18     myName = "Cyrus";
19     myNumber = 5;
20     document.writeln("myName = " + myName + "<br />");
21     document.writeln("myNumber = " + myNumber + "<br />");
22     document.writeln("typeof myName = " + typeof(myName) + "<br />");
23     document.writeln("typeof myNumber = " + typeof(myNumber) + "<br />");
24
25
26     myName = 4;
27     myNumber = 5.6;
28     document.writeln("myName = " + myName + "<br />");
29     document.writeln("myNumber = " + myNumber + "<br />");
30     document.writeln("typeof myName = " + typeof(myName) + "<br />");
31     document.writeln("typeof myNumber = " + typeof(myNumber) + "<br />");
32
33 </script>
34 </head>
35 <body>
36
37 </body>
38 </html>
length: 1041 lines: 39 Ln: 31 Col: 73 Sel: 0 Dos\Windows ANSI INS
```

```
Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/wamp/www/..._HEAD_Script_5a.html +
file:///C:/wamp/www/..._HEAD_Script_5a.html
Y Yahoo!

myName = undefined
myNumber = undefined
typeof myName =undefined
typeof myNumber =undefined

myName = Cyrus
myNumber = 5
typeof myName =string
typeof myNumber =number

myName = 4
myNumber = 5.6
typeof myName =number
typeof myNumber =number
```

OUTPUT

- Output methods in JavaScript apply to the DOM's DOCUMENT object.

- Example:

```
document.write("hello");
```

```
document.writeln("hello");
```

Remember however, document is part of a window (or BOM object)

```
myWin=window.open("");           //open blank window and write to it
myWin.document.open();           //open document stream
myWin.document.write("Hello");
myWin.document.close();
```

INPUT

- Input methods in JavaScript apply to the BOM's WINDOW object.

- Example:

```
window.prompt(msg, [input]);
```

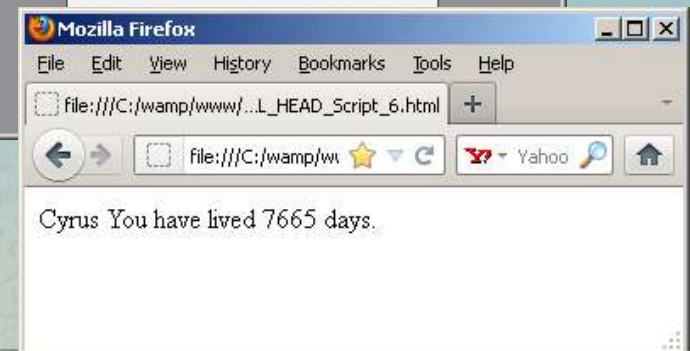
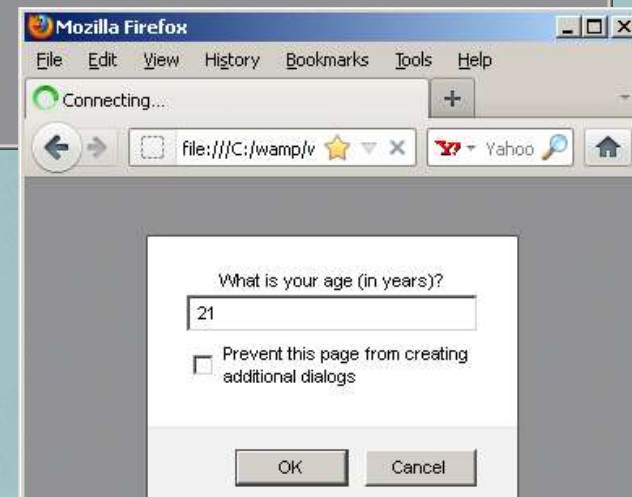
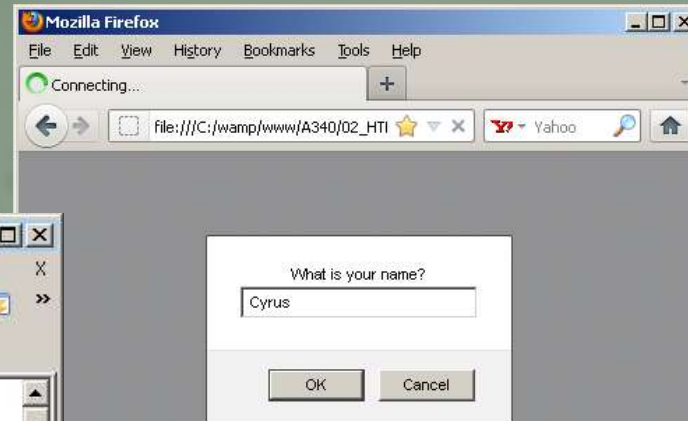
```
var myName;
```

```
myName = window.prompt("please enter your name")
```

- Displays a Prompt dialog box with a message. Optional "input" argument allows you to specify the default input (response) that gets entered into the dialog box.
- Prompt will return the string the user has entered.

INPUT

```
C:\wamp\www\A340\02_HTML_HEAD_Script_6.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
02_HTML_HEAD_Script_6.html
1 <!DOCTYPE html>
2 <!-- 02_HTML_HEAD_Script_5.html
3
4 Input and Oupptut
5
6 -->
7 <html>
8 <head>
9 <script type="text/javascript">
10
11     var myName ;
12     var myAge ;
13
14     myName = window.prompt("What is your name?");
15     myAge = window.prompt("What is your age (in years)?");
16
17     var daysLived = parseInt(myAge) * 365;
18     document.writeln(myName);
19     document.writeln("You have lived " + daysLived +
20         " days.");
21
22 </script>
23 </head>
24 <body>
25
26 </body>
27 </html>
length : 460 | Ln : 19 Col : 43 Sel : 0 Dos\Windows ANSI INS
```





CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

Java
Script
C

ARRAY

- Can be allocated statically or dynamically.

Dynamic arrays:

```
var mySchedule = new Array([size]);  
mySchedule[0] = "CSCI-A 340";  
mySchedule[1] = "MATH-M 215";
```

Literal arrays:

```
var mySchedule = ["CSCI-A 340", "MATH-M 215"];
```

Array properties:

```
mySchedule.length
```

ARRAY

```
C:\wamp\www\A340\02_HTML_HEAD_Script_7.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
02_HTML_HEAD_Script_6.html 02_HTML_HEAD_Script_7.html

10
11 //Dynamic arrays:
12 var mySchedule = new Array();
13 mySchedule[0] = "CSCI-A 340";
14 mySchedule[1] = "MATH-M 215";
15
16
17 //Array properties:
18 document.write(mySchedule.length);
19 document.write("<br>");
20
21 for(var i=0; i< mySchedule.length; i++)
22 {
23     document.write(mySchedule[i] + "<br />");
24 }
25
26 //Literal arrays:
27 var mySchedule2 = ["CSCI-A 340", "MATH-M 215"];
28 document.write(mySchedule2.length);
29 document.write("<br>");
30
31 for(var i=0; i< mySchedule2.length; i++)
32 {
33     document.writeln(mySchedule2[i] + "<br />");
34 }
35
36 </script>
37 </head>
38 <body>
39
40 </body>
```

Mozilla Firefox

file:///C:/wamp/www/...L_HEAD_Script_7.html

file:///C:/wamp

2
CSCI-A 340
MATH-M 215

2
CSCI-A 340
MATH-M 215

OPERATION ON ARRAYS

- **Array.sort([SortFunction])**

- By default the function sort(), sorts the arrays alphabetically and in ascending order.

- **//Sort Alphabetically and ascending:**

```
var myArray=["Bob","Bully","Amy"];  
myArray.sort(); //Array now becomes ["Amy", "Bob", "Bully"];
```

- **//Sort Alphabetically and descending:**

```
var myarray=["Bob","Bully","Amy"];  
myArray.sort();  
myArray.reverse(); //Array now becomes ["Bully", "Bob", "Amy"]
```

- To sort the array numerically, need to compare the relationship between "a" to "b", with a return value of <0 indicating to sort ascending, and >0 to sort descending

- **//Sort numerically and ascending:**

```
var myArray=[25, 8, 7, 41];  
myArray.sort(function(a,b){return a - b}); //Array now becomes [7, 8, 25, 41]
```

- **//Sort numerically and descending:**

```
var myArray=[25, 8, 7, 41];  
myArray.sort(function(a,b){return b - a}); //Array now becomes [41, 25, 8, 7]
```

<http://www.javascriptkit.com/jsref/arrays.shtml#e2>

OPERATIONS ON AN ARRAY:

- **Sorting an Array (Bubble Sort)**

```
Private Sub BubbleSort(ByRef TheArray() As Integer)
```

```
    Dim Pass, Index, Hold As Integer
```

```
    For Pass = 1 To TheArray.GetUpperBound(0)
```

```
        For Index = 0 To TheArray.GetUpperBound(0) - 1
```

```
            If (TheArray(Index) > TheArray(Index + 1) )Then
```

```
                Hold = TheArray(Index)  
                TheArray(Index) = TheArray(Index + 1)  
                TheArray(Index + 1) = Hold
```

Swap

```
            End If
```

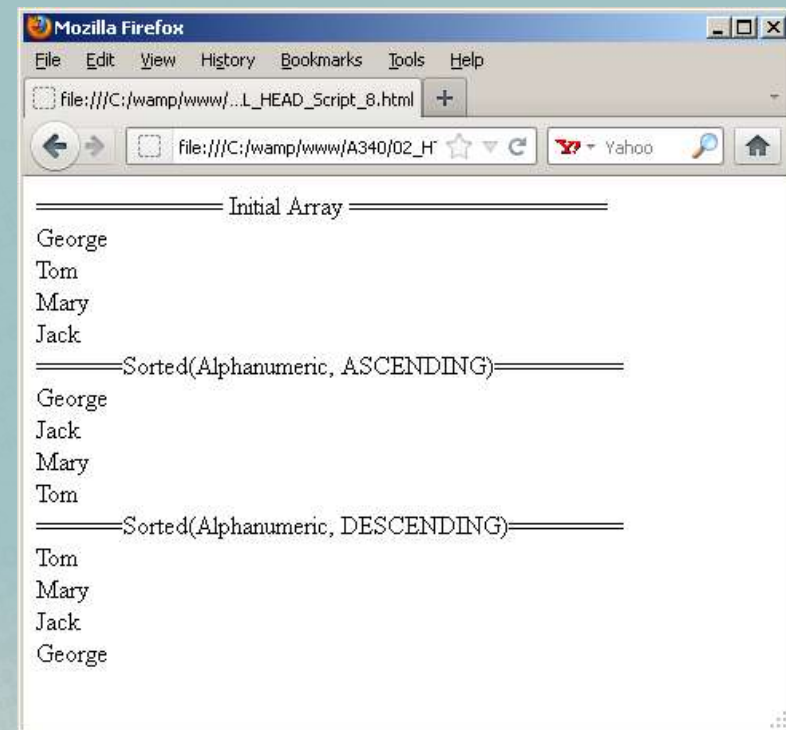
```
        Next Index
```

```
    Next Pass
```

```
End Sub
```

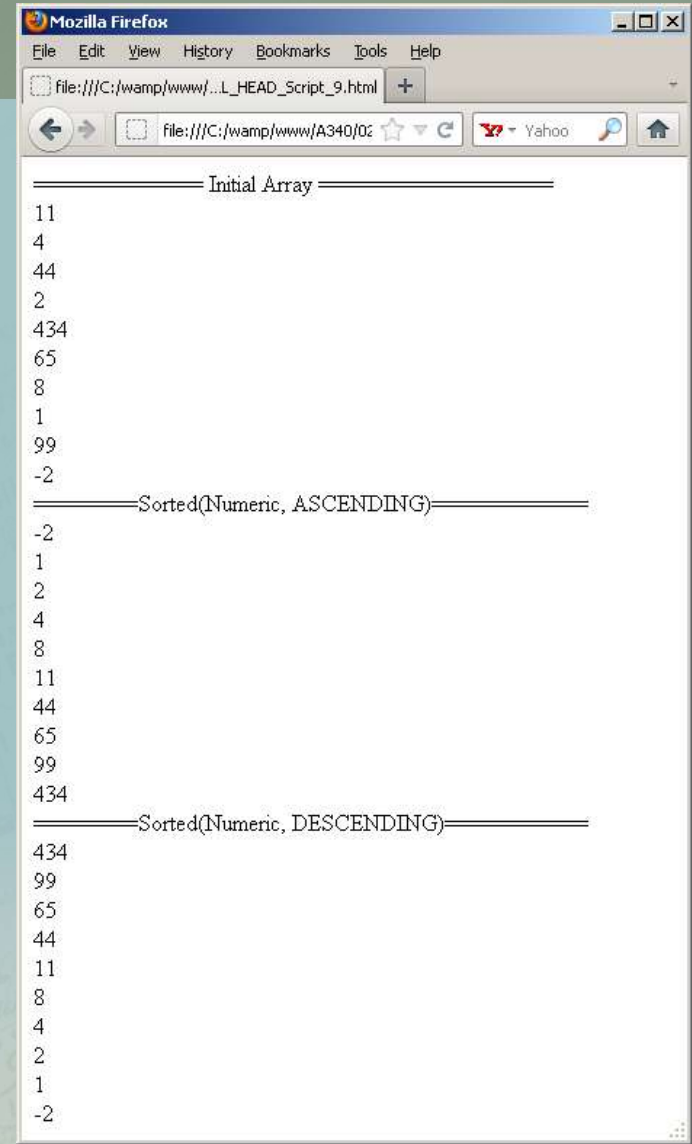
SORTING (ALPHANUMERIC)

```
1 <!DOCTYPE html>
2 <!-- 02_HTML_HEAD_Script_8.html
3 Array Operations
4 Alphanumeric Sort()
5 Reverse()
6 -->
7 <html>
8 <head>
9 <script type="text/javascript">
10
11     document.writeln("===== Initial Array =====<br>");
12     var myArray = ["George", "Tom", "Mary", "Jack"];
13
14     for(var i=0; i< myArray.length; i++)
15     {
16         document.writeln(myArray[i]);
17         document.write("<br>");
18     }
19
20     document.writeln("=====Sorted(Alphanumeric, ASCENDING)=====<br>");
21     myArray.sort();
22     for(var i=0; i< myArray.length; i++)
23     {
24         document.writeln(myArray[i]);
25         document.write("<br>");
26     }
27
28     document.writeln("=====Sorted(Alphanumeric, DESCENDING)=====<br>");
29     myArray.reverse();
30     for(var i=0; i< myArray.length; i++)
31     {
32         document.writeln(myArray[i]);
33         document.write("<br>");
34     }
35
36 </script>
37 </head>
38 <body>
39
40 </body>
41 </html>
42
```



SORTING (NUMERIC)

```
11 <script type="text/javascript">
12 // -----
13 // Note:
14 // The following two functions are used by the array.sort() function
15 // to allow for numeric sorting.
16 function ASCENDING (a,b) {return a-b}
17 function DESCENDING (a,b) {return b-a}
18 // -----
19 var myArray = [11,4,44,2,434,65,8,1,99,-2];
20
21 document.writeln("===== Initial Array =====<br>");
22 for(var i=0; i< myArray.length; i++)
23 {
24     document.writeln(myArray[i]);
25     document.write("<br>");
26 }
27
28 document.writeln("=====Sorted(Numeric, ASCENDING)=====<br>");
29 myArray.sort(ASCENDING);
30 for(var i=0; i< myArray.length; i++)
31 {
32     document.writeln(myArray[i]);
33     document.write("<br>");
34 }
35
36 document.writeln("=====Sorted(Numeric, DESCENDING)=====<br>");
37 myArray.sort(DESCENDING);
38 for(var i=0; i< myArray.length; i++)
39 {
40     document.writeln(myArray[i]);
41     document.write("<br>");
42 }
43
44 </script>
```



OPERATION ON ARRAYS

Array. Concat()

- Concatenates either a single elements or another array of elements with the existing array, and returns the new array.

```
var fruits=["Apple", "Oranges"];  
var meat=["Pork", "Chicken"];
```

```
var dinner=fruits.concat(meat);  
//creates ["Apple", "Oranges", "Pork", "Chicken"]. fruits and meat arrays not changed.
```

```
var snack=fruits.concat("Grapes", ["Cookies", "Milk"]);  
//creates ["Apple", "Oranges", "Grapes", "Cookies", "Milk"] fruits array not changed.
```

<http://www.javascriptkit.com/jsref/arrays.shtml#e2>

CONCATENATION

```
C:\wamp\www\A340\02_HTML_HEAD_Script_10.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
O2_HTML_HEAD_Script_10.html
1 <!DOCTYPE html>
2 <!-- O2_HTML_HEAD_Script_10.html
3
4 Array Operations
5 Concatenation
6 Concatenates either a single elements or another array of
7 elements with the existing array, and returns the new array.
8
9 -->
10 <html>
11 <head>
12 <script type="text/javascript">
13
14     var male=["Tom", "Jack", "John"];
15     var female=["Sue", "Mary", "Anne"];
16
17     female = female.concat("Niki");    //Concat Niki to females
18     var myClass =male.concat(female);  //Concatenate the two arrays
19
20     // var myClass =[];
21     // myClass = myClass.concat (male);
22     // myClass = myClass.concat (female, "Niki");
23
24     document.writeln("<b>Male: </b>" + male + "<br>");
25     document.writeln("<b>Female: </b>" +female + "<br>");
26     document.writeln("<b>Concat: </b>" + myClass + "<br>");
27
28     myClass.sort();
29     document.writeln("<b>Sorted: </b>" + myClass + "<br>");
30
31
32 </script>
33 </head>
34 <body>
35
36 </body>
37 </html>
38
length : 882 lines : 38 Ln : 9 Col : 1 Sel : 0 Dos\Windows ANSI INS
```

```
Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/wamp/www/..._HEAD_Script_10.html
file:///C:/wamp/www/
Yahoo
Male: Tom,Jack,John
Female: Sue,Mary,Anne,Niki
Concat: Tom,Jack,John,Sue,Mary,Anne,Niki
Sorted: Anne,Jack,John,Mary,Niki,Sue,Tom
```

OPERATION ON ARRAYS

Array. indexOf(targetElement)

- Returns the first index in which targetElement (value) is found within an array, or -1 if nothing is found.

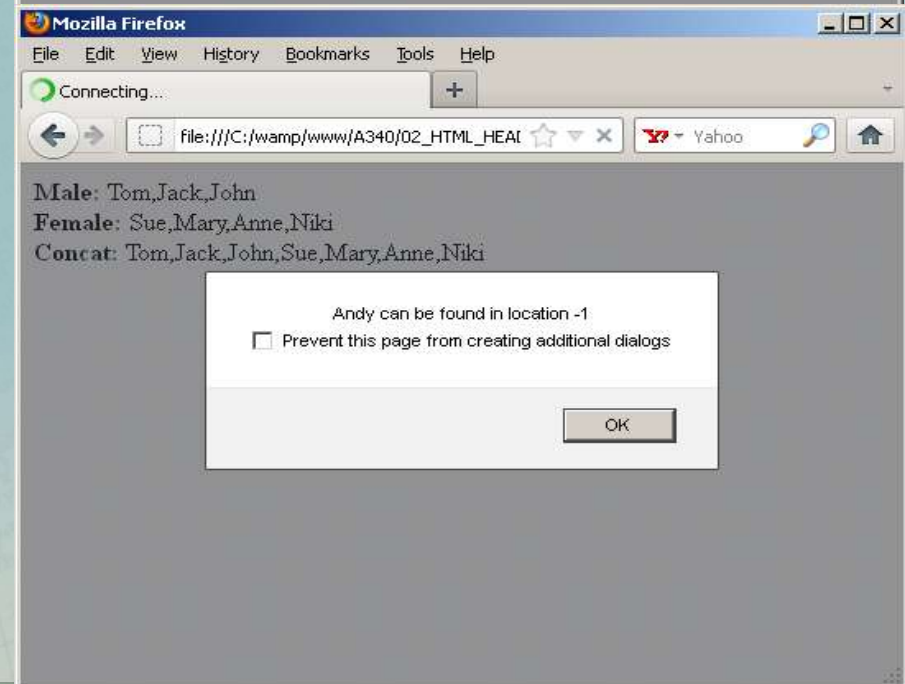
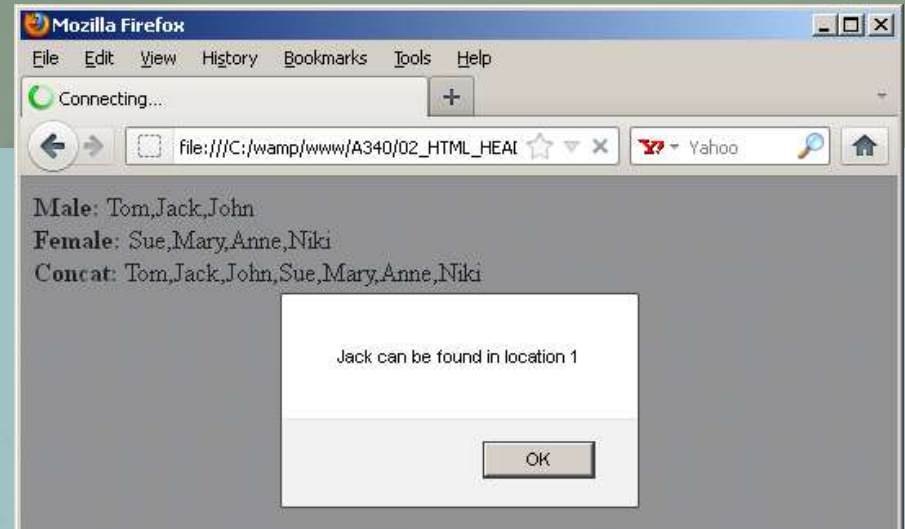
- `var fruits=["Apple", "Oranges", "Pork", "Chicken"];`

`alert(fruits.indexOf("Pork")); //alerts 2`

<http://www.javascriptkit.com/jsref/arrays.shtml#e2>

INDEX OF

```
1 <!DOCTYPE html>
2 <!-- O2_HTML_HEAD_Script_11.html
3
4 Array Operations
5 Array.indexOf(targetElement)
6
7 Returns the first index in which targetElement (value) is found within an
8 array, or -1 if nothing is found.
9
10 -->
11 <html>
12 <head>
13 <script type="text/javascript">
14
15     var male=["Tom", "Jack", "John"];
16     var female=["Sue", "Mary", "Anne"];
17
18     female = female.concat("Niki");    //Concat Niki to females
19     var myClass =male.concat(female);  //Concatenate the two arrays
20
21     document.writeln("<b>Male: </b>" + male + "<br>");
22     document.writeln("<b>Female: </b>" +female + "<br>");
23     document.writeln("<b>Concat: </b>" + myClass + "<br>");
24
25     alert( "Jack can be found in location " + myClass.indexOf("Jack"));
26     alert( "Andy can be found in location " + myClass.indexOf("Andy"));
27
28 </script>
```



OPERATION ON ARRAYS

Array. Join([separator])

- Converts each element within the array to a string, and joins them into one large string. Pass in an optional separator as argument to be used to separate each array element. If none is passed, the default comma (,) is used:

Array. Map(mappingfunction())

- Returns a new array based on the return value of testfunction() on each of the array elements. Original array is not changed. Use it to transform the values of all elements within an array using some logic and derive the results as a new array.

Array.push(value);

- Adds the argument values to the end of the array, and modifies the original array with the new additions. Returns the new length of the array.

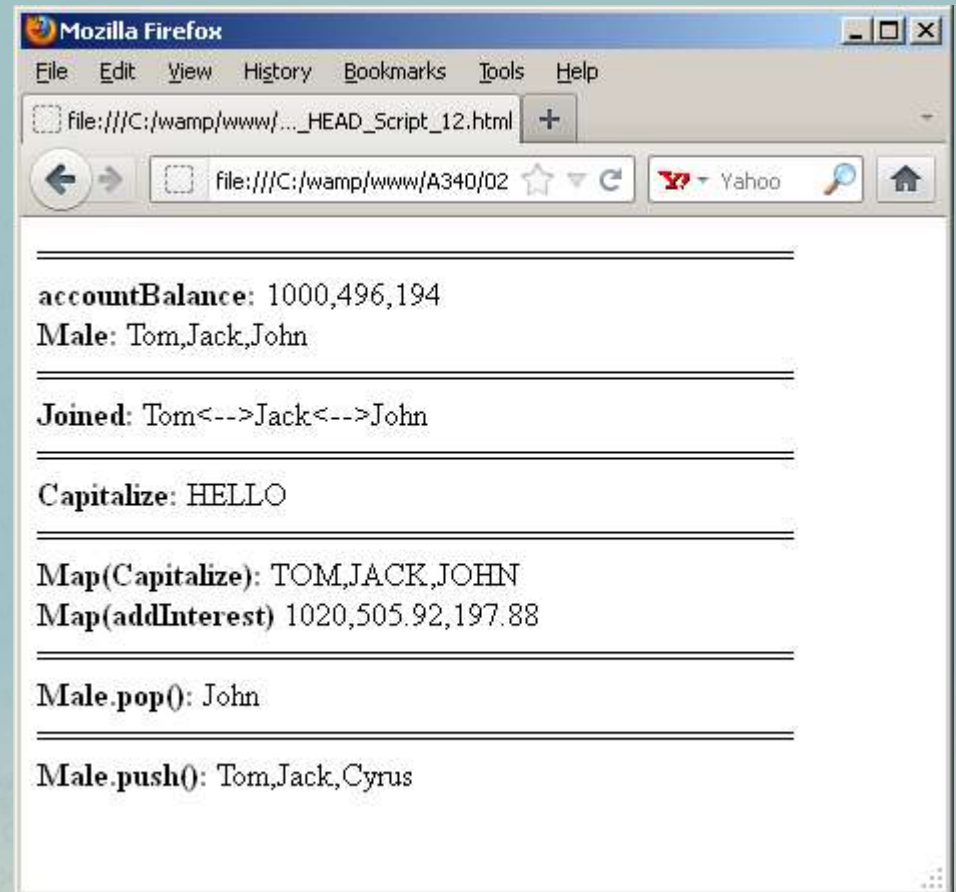
Array.pop();

- Deletes the last element within array and returns the deleted element. Original array is modified.

<http://www.javascriptkit.com/jsref/arrays.shtml#e2>

OTHER ARRAY OPERATIONS

```
14 <script type="text/javascript">
15     function capitalize(theString)
16     {
17         return (theString.toUpperCase());
18     }
19
20     function addInterest(element) {
21         return (element*1.02);
22     }
23
24     var accountBalance=[1000, 496, 194];
25     var male=["Tom", "Jack", "John"];
26
27     document.writeln("=====<br>");
28     document.writeln("<b>accountBalance: </b>" + accountBalance + "<br>");
29     document.writeln("<b>Male: </b>" + male + "<br>");
30
31     // Test join()
32     document.writeln("=====<br>");
33     var joined = male.join("<--->");
34     document.writeln("<b>Joined: </b>" + joined + "<br>");
35
36     // Test Capitalize
37     document.writeln("=====<br>");
38     document.writeln("<b>Capitalize: </b>" + capitalize("hello") + "<br>");
39
40     // Test map()
41     document.writeln("=====<br>");
42     var capitalized = male.map(capitalize);
43     document.writeln("<b>Map(Capitalize): </b>" + capitalized + "<br>");
44
45     var balancePlusInterest = accountBalance.map(addInterest);
46     document.writeln("<b> Map(addInterest) </b>" + balancePlusInterest +
47         "<br>");
48
49     // Test pop()
50     document.writeln("=====<br>");
51     var poppedElement = male.pop();
52     document.writeln("<b>Male.pop(): </b>" + poppedElement + "<br>");
53
54     // Test push()
55     document.writeln("=====<br>");
56     male.push("Cyrus");
57     document.writeln("<b>Male.push(): </b>" + male + "<br>");
58 </script>
```



ASSOCIATIVE ARRAYS

- Associative array is an array that uses a “string” (instead of a number) as the index to the elements of an array.

- **Example:**

```
var testScore = []; // Create an empty array
```

```
testScore["Bob"] = 89;  
testScore["Mary"] = 95;
```

```
document.write(testScore["Bob"]);
```

```
// The array.length will print a zero because apparently  
// the length only counts the array with numeric index!!  
document.write(testScore.length + "<br />");
```

```
// To examine the elements of the array  
for(var student in testScore)  
{  
    document.write(student + " : ");  
    document.write(testScore[student] + "<br />");  
}
```



CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

Java
Script
D

LOOPS

- **While Loop**

```
var number=0;
while (number<5){
    document.write(number+"<br>");
    number++;
}
```

- **Do-While Loop**

```
var number=0;
do{
    document.write(number+"<br>");
    number++;
}while (number<5) ;
```

- **For Loop**

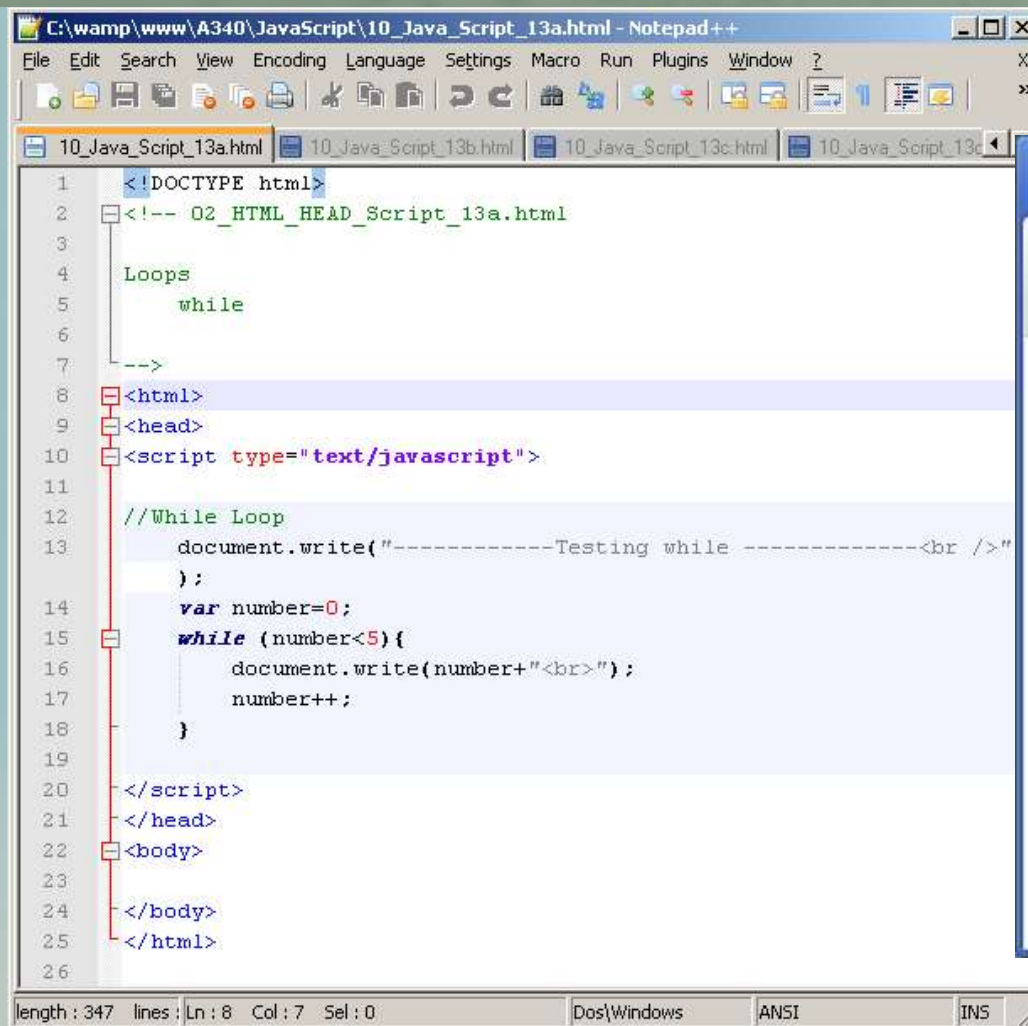
```
for (var i=0; i<3; i++){
    document.write("This text is repeated three times<br>");
}
```

- **For -in Loop**

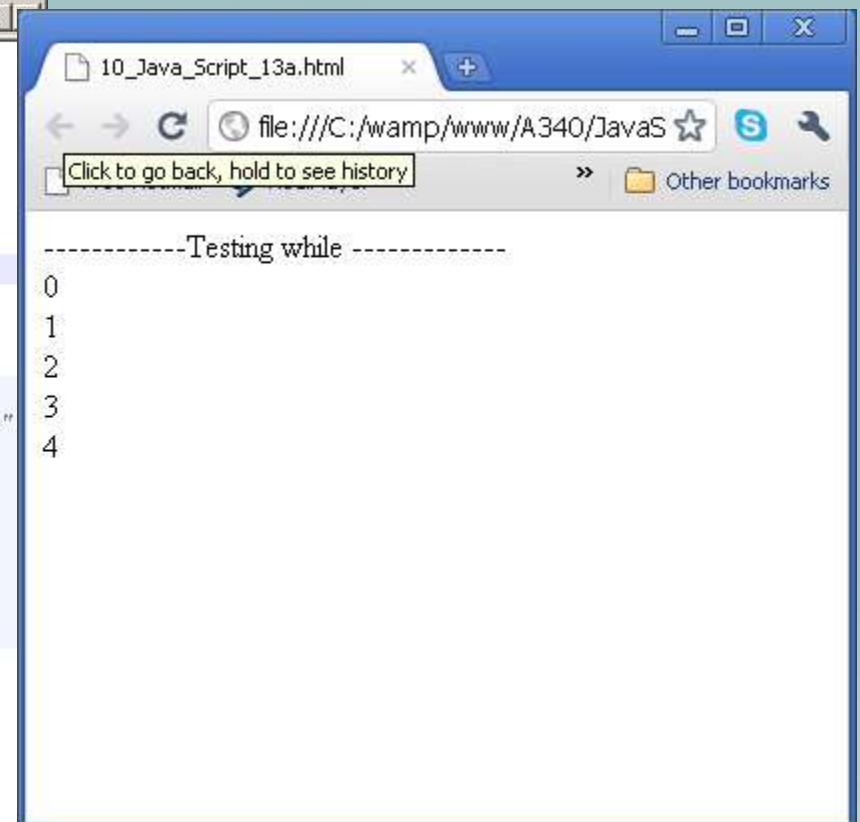
```
var userprofile={name:'George', age:30, sex:'male', getage:function(){return this.age}};

for (var attr in userprofile){
    document.write("<b>" +attr+":</b> "+userprofile[attr]+"<br />");
}
```

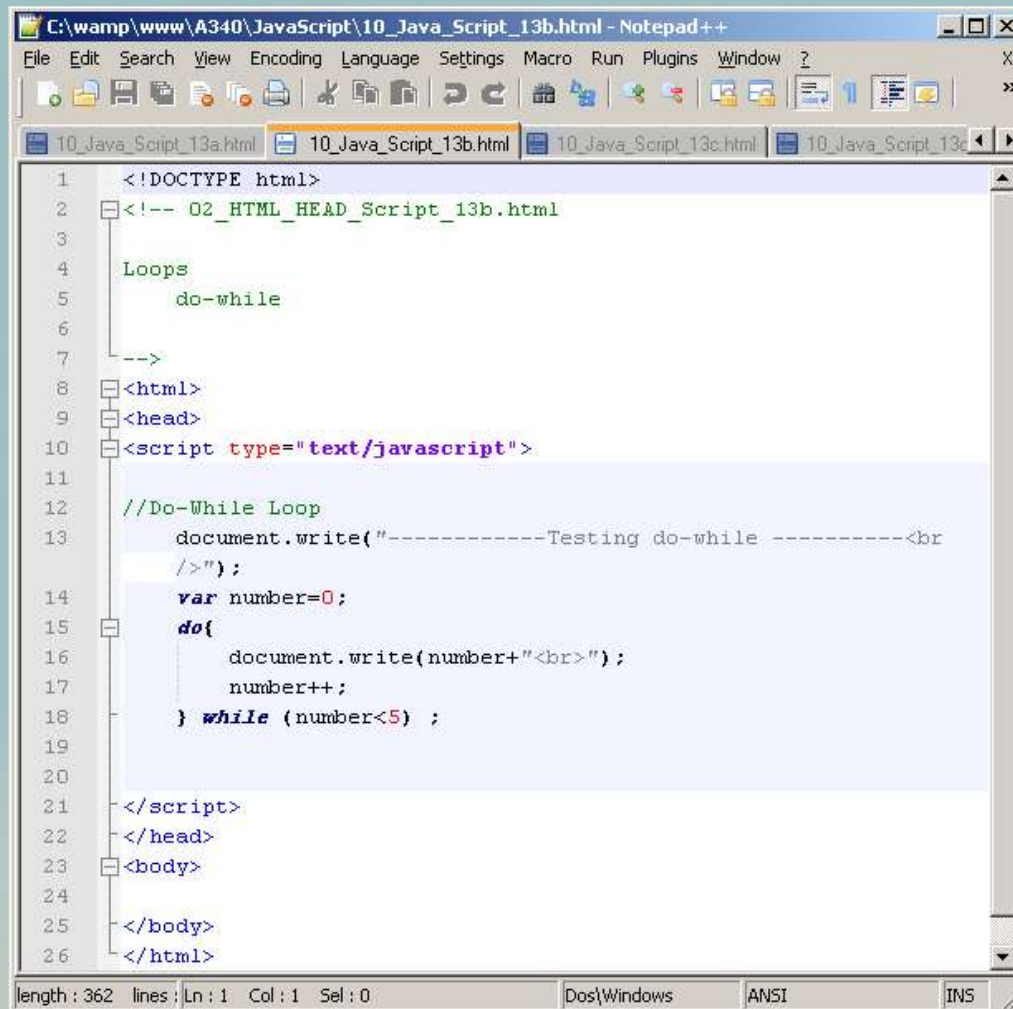
WHILE LOOP



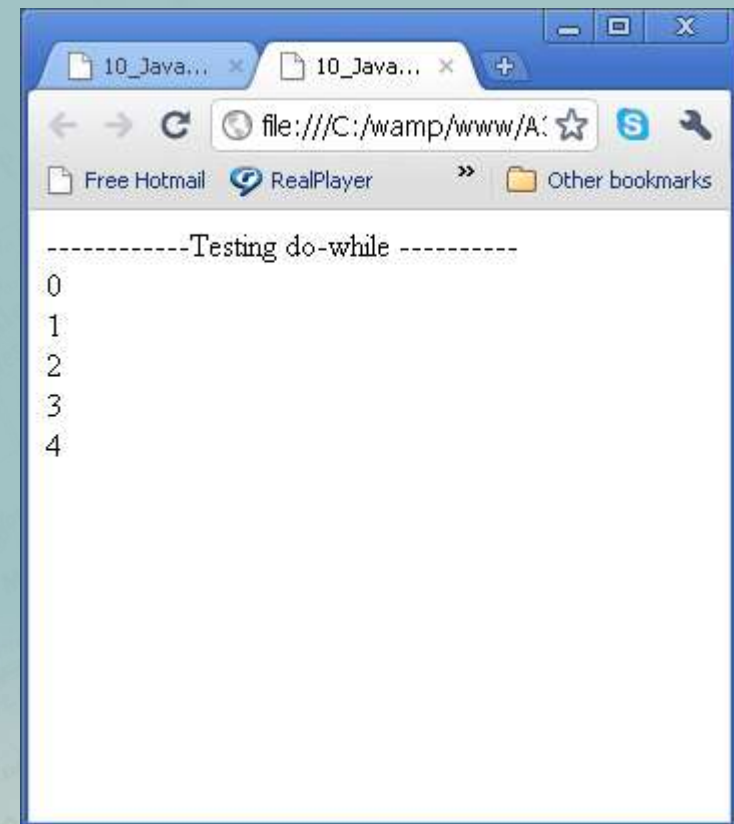
```
1 <!DOCTYPE html>
2 <!-- 02_HTML_HEAD_Script_13a.html
3
4   Loops
5     while
6
7   -->
8 <html>
9 <head>
10 <script type="text/javascript">
11
12   //While Loop
13   document.write("-----Testing while -----<br />"
14   );
15   var number=0;
16   while (number<5){
17     document.write(number+"<br>");
18     number++;
19   }
20 </script>
21 </head>
22 <body>
23
24 </body>
25 </html>
26
```



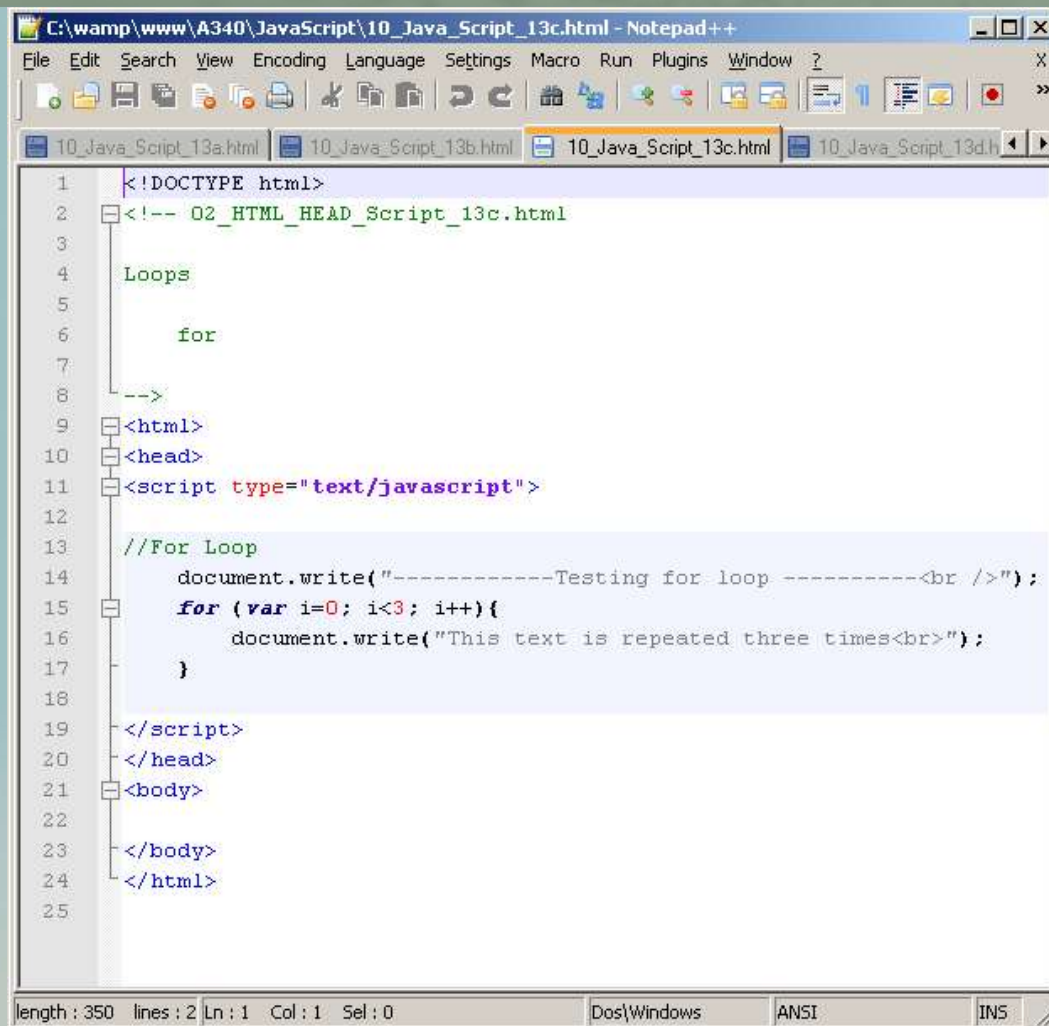
DO-WHILE LOOP



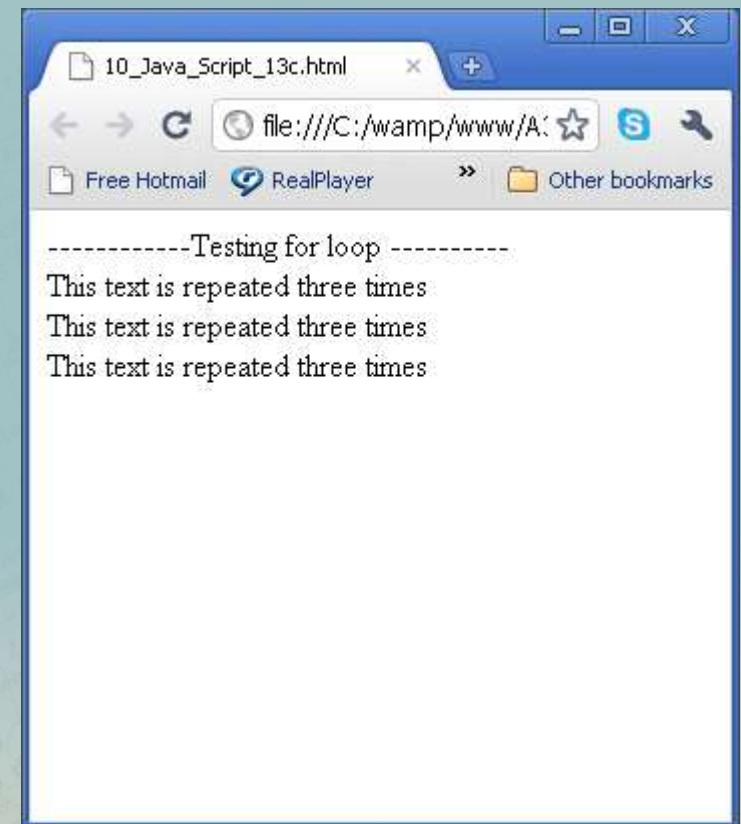
```
1  <!DOCTYPE html>
2  <!-- 02_HTML_HEAD_Script_13b.html
3
4  Loops
5      do-while
6
7  -->
8  <html>
9  <head>
10 <script type="text/javascript">
11
12 //Do-While Loop
13 document.write("-----Testing do-while -----<br
14 />");
15 var number=0;
16 do{
17     document.write(number+"<br>");
18     number++;
19 } while (number<5) ;
20
21 </script>
22 </head>
23 <body>
24
25 </body>
26 </html>
```



FOR LOOP



```
1 <!DOCTYPE html>
2 <!-- 02_HTML_HEAD_Script_13c.html
3
4 Loops
5
6 for
7
8 -->
9 <html>
10 <head>
11 <script type="text/javascript">
12
13 //For Loop
14 document.write("-----Testing for loop -----<br />");
15 for (var i=0; i<3; i++){
16     document.write("This text is repeated three times<br>");
17 }
18
19 </script>
20 </head>
21 <body>
22
23 </body>
24 </html>
25
```



FOR-IN LOOP

```
3
4  Loops
5    for-in
6
7  -->
8  <html>
9  <head>
10 <script type="text/javascript">
11
12
13 //For-in Loop
14 document.write("-----Testing for-in loop-----<br />");
15 var userprofile= {name: 'George',
16                  age: 30,
17                  sex: 'male',
18                  getage: function(){return this.age}
19                  };
20 for (var attr in userprofile){
21     document.write("<b>" + attr + "</b> " +
22                  userprofile[attr] +
23                  "<br />");
24 }
25
26
27 document.write("-----Examining the attributes-----<br />");
28 document.write(userprofile.getage + "<br />"); // display the element
29 document.write(userprofile.getage() + "<br />"); // execute the element
30
31 </script>
32 </head>
33 <body>
34
35 </body>
```

10_Java_Script_13d.html

file:///C:/wamp/www/A/

Free Hotmail RealPlayer

-----Testing for-in loop-----

name: George

age: 30

sex: male

getage: function () {return this.age}

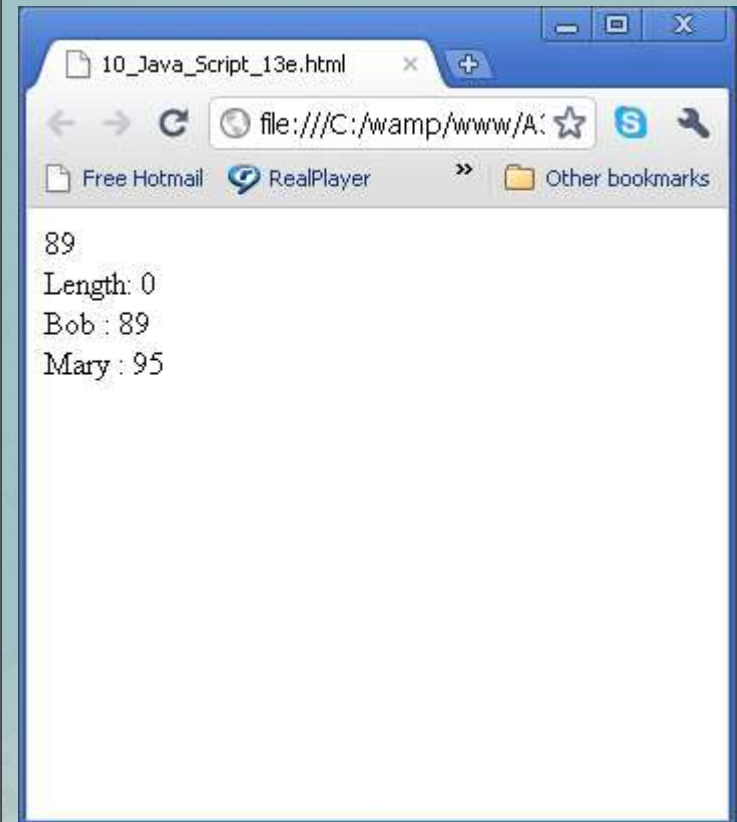
-----Examining the attributes-----

function () {return this.age}

30

ASSOCIATIVE ARRAYS AND FOR-IN LOOP

```
10_Java_Script_13c.html 10_Java_Script_13d.html 10_Java_Script_13e.html
4   Use of for-in loops and Associative Arrays
5
6   -->
7   <html>
8   <head>
9   <script type="text/javascript">
10
11
12   var testScore = [];           // Create an empty array
13
14   testScore["Bob"] = 89;
15   testScore["Mary"] = 95;
16
17   document.write(testScore["Bob"]);
18
19   // The array.length will print a zero because apparently
20   // the length only counts the array with numeric index!!
21   document.write(testScore.length + "<br />");
22
23
24   // To examine the elements of the array
25   for(var student in testScore)
26   {
27       document.write(student + " : ");
28       document.write(testScore[student] + "<br />");
29   }
30
31
32   </script>
33   </head>
34   <body>
35
36   </body>
```



CONDITIONAL

- Same as C and C++

- If Statement

```
if (expression)
    statement1;
else if (expression2)
    statement2;
else
    statement3;
```

- Switch Statement

```
switch (expression){
    case label1:
        statement1
        break
    case label2:
        statement2
        break
    default: statement3;
}
```

MODULES

- JavaScript has several built-in objects. Each of these objects have several **built-in Functions**:

Built-in Objects:

- Array
- Boolean
- Date
- Math
- Number
- String
- RegExp
- Global

Some Built-in Functions:

- alert();
- confirm();
- prompt();
- parseInt(x);
- parseFloat(x);
- Math.max(x,y);
- Math.min(x,y);
- Math.pow();
- Math.random();
- Date();
- toString()
- charAt()
- toLowerCase()
- toUpperCase()
- escape(), unescape()

MODULES

- **User-defined Functions:**

```
function getArea(w, h)
{
    var area = w*h;
    return area;
}
```

```
document.writeln(getArea(3,5)); //call the function
```

MODULES

- Recursive Functions:

```
function factorial(number)
{
    if (number <=1)        //base case
        return 1;
    else
        return (number * factorial(number-1));
}
```

alert(factorial(5)); //call the function

RECURSIVE FUNCTIONS

```
O2_HTML_HEAD_Script_13.html
1  <!DOCTYPE html>
2  <!-- O2_HTML_HEAD_Script_13.html
3
4  Recursive functions
5
6  -->
7  <html>
8  <head>
9  <script type="text/javascript">
10
11
12
13  //Recursive Functions:
14  function factorial(number)
15  {
16      if (number <=1)    //base case
17          return 1;
18      else
19          return (number * factorial(number-1));
20  }
21
22
23  for (var i=1; i<20; i++)
24      document.write("Factorial of " + i + " = " + factorial(i) + "<br>");
25
26  </script>
27  </head>
28  <body>
29
30  </body>
31  </html>
32
```

Mozilla Firefox

File Edit View History Bookmarks Tools He

file:///C:/wamp/www/..._HEAD_Script_13.html +

file:///C:/wamp/www/A340/O2_HTML_

Factorial of 1 =1
Factorial of 2 =2
Factorial of 3 =6
Factorial of 4 =24
Factorial of 5 =120
Factorial of 6 =720
Factorial of 7 =5040
Factorial of 8 =40320
Factorial of 9 =362880
Factorial of 10 =3628800
Factorial of 11 =39916800
Factorial of 12 =479001600
Factorial of 13 =6227020800
Factorial of 14 =87178291200
Factorial of 15 =1307674368000
Factorial of 16 =20922789888000
Factorial of 17 =355687428096000
Factorial of 18 =6402373705728000
Factorial of 19 =121645100408832000

JAVASCRIPT STRINGS

- A JavaScript string is zero or more characters written inside quotes. (single or double quotes)

Example:

```
var myCar = "Honda"; // Double quotes  
var yourCar = 'Ford'; // Single quotes
```

- An **Escape character** can be used if we need to use the “quote” inside a string

Example:

```
var myCar = "Honda \\"Accord\\" ";  
var yourCar = 'Ford \'Mustang\'' ;
```

- We can also mix single and double quotes.

Example:

```
var myCar = "Honda 'Accord' ";
```

STRING LITERALS VS. STRING OBJECTS

- `var x = "John";`
- `var y = new String("John");`

These are not the same. One is a string one is an Object.

`// typeof x will return string`
`// typeof y will return object`

- The condition `(x == y)`

Returns **true** because it compares their values.

- The condition `(x === y)`

Returns **false** because it compare their type and their value

COMPARING TWO STRING OBJECTS

NOW:

- `var x = new String("John");`
- `var y = new String("John");`

These are not the same. One is a string one is an Object.

`// typeof x will return object`
`// typeof y will return object`

- The condition `(x == y)`

Returns **false** because the two objects are different!

- The condition `(x === y)`

Returns **false** because the two objects are different

STRING OPERATIONS AND PROPERTIES

- Length property.

```
var alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var count = alpha.length;
```

- IndexOf() method

- returns the location/index of the first occurrence of a specified text in a string.

```
var str = "Please locate where 'locate' occurs!";  
var pos = str.indexOf("locate"); // returns 7
```

STRING OPERATIONS AND PROPERTIES

- Length property.

```
var alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var count = alpha.length;
```

- IndexOf() and lastIndexOf() methods
 - returns the location/index of the first occurrence of a specified text in a string. Returns -1 if not there.

```
var str = "Please locate where 'locate' occurs!";  
var firstOccurrence = str.indexOf("locate");    // returns 7  
var lastOccurrence = str.lastIndexOf("locate"); // returns 21
```

STRING OPERATIONS AND PROPERTIES

- **Look these up on your own:**

- Slice(start, end);
- Substring(start, end);
- Substr(start, length);
- toUpperCase();
- toLowerCase();
- Concat();
- Trim();

- Search(regExpr); or
- Search(string);
 - returns the location/index of the first occurrence of a specified text in a string. Returns -1 if not there.

```
var str = "Please locate where 'locate' occurs!";  
var firstOccurence = str.search("locate");           // returns 7
```

- Replace(substring_value_in_string, new_sub_string); or
- Replace(regExpr, "new_sub_string");

REGULAR EXPRESSIONS

- Regular expressions describe regular languages in formal language theory. (outside the scope of our discussion, but covered in concept of program language, formal languages, or compilers class in CS)
- Regular expressions consist of **constants**, which denote sets of strings, and **operator symbols**, which denote operations over these sets.

Examples:

- `a|b*` denotes $\{\epsilon, "a", "b", "bb", "bbb", \dots\}$
- `(a|b)*` denotes the set of all strings with no symbols other than "a" and "b", including the empty string: $\{\epsilon, "a", "b", "aa", "ab", "ba", "bb", "aaa", \dots\}$
- `ab*(c|ε)` denotes the set of strings starting with "a", then zero or more "b"s and finally optionally a "c": $\{"a", "ac", "ab", "abc", "abb", "abbc", \dots\}$
- `(0|(1(01*0)*1))*` denotes the set of binary numbers that are multiples of 3: $\{\epsilon, "0", "00", "11", "000", "011", "110", "0000", "0011", "0110", "1001", "1100", "1111", "00000", \dots\}$

- You can read more about formal language theory here:
- https://en.wikipedia.org/wiki/Regular_expression#Formal_definition

REGULAR EXPRESSIONS (REGEXES)

- A **regular expression** is a sequence of characters that defines a “**search pattern**”.
 - For example: “ABC”
- The pattern is then used by a “search algorithm” to search a **target string** of characters.
 - For example: “ABC Computer Company”
- A regular expression can be a single character, or a more complicated pattern.
- An example of a more **complicated** pattern may include patterns with **WILD CARD** characters.
 - For example “a.” means any two letter patterns starting with an “a” and followed by any single character.
- So characters in regular express can be categorized as follows:
 - Regular character (with Literal meaning)
 - Metacharacter (chars that have special meaning)
 - Boolean Char “or”: |
 - Grouping chars: ()
 - Quantifiers: ?, *, +, {n}, {min, }, {min, max}
 - Wild Cards: . ,
 - For example “a.” has two characters in it. “a” is a regular character, and “.” is a metacharacter.

ERROR HANDLING

- Similar to Java /C++ error handling:

```
try {  
    Block of code to try  
}  
catch(err) {  
    Block of code to handle errors  
}  
finally {  
    Block of code to be executed  
    regardless of the try / catch result  
}
```

- https://www.w3schools.com/js/js_errors.asp

- **try** lets you test a block of code for errors.
- **catch** lets you handle the error.
- **throw** lets you create custom errors.
- **finally** lets you execute code, after try and catch, regardless of the result.

ERROR HANDLING

- Sample from https://www.w3schools.com/js/tryit.asp?filename=tryjs_finally_error

```
<!DOCTYPE html>
<html>
<body>

<p>Please input a number between 5 and 10:</p>

<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>

<p id="p01"></p>

<script>
function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if(x == "") throw "is empty";
    if(isNaN(x)) throw "is not a number";
    x = Number(x);
    if(x > 10) throw "is too high";
    if(x < 5) throw "is too low";
  }
  catch(err) {
    message.innerHTML = "Input " + err;
  }
  finally {
    document.getElementById("demo").value = "";
  }
}
</script>

</body>
</html>
```

Please input a number between 5 and 10:

3 Test Input

Input is too low



CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

Java
Script
E



JAVASCRIPT AND ITS RELATION WITH DOM VS. BOM

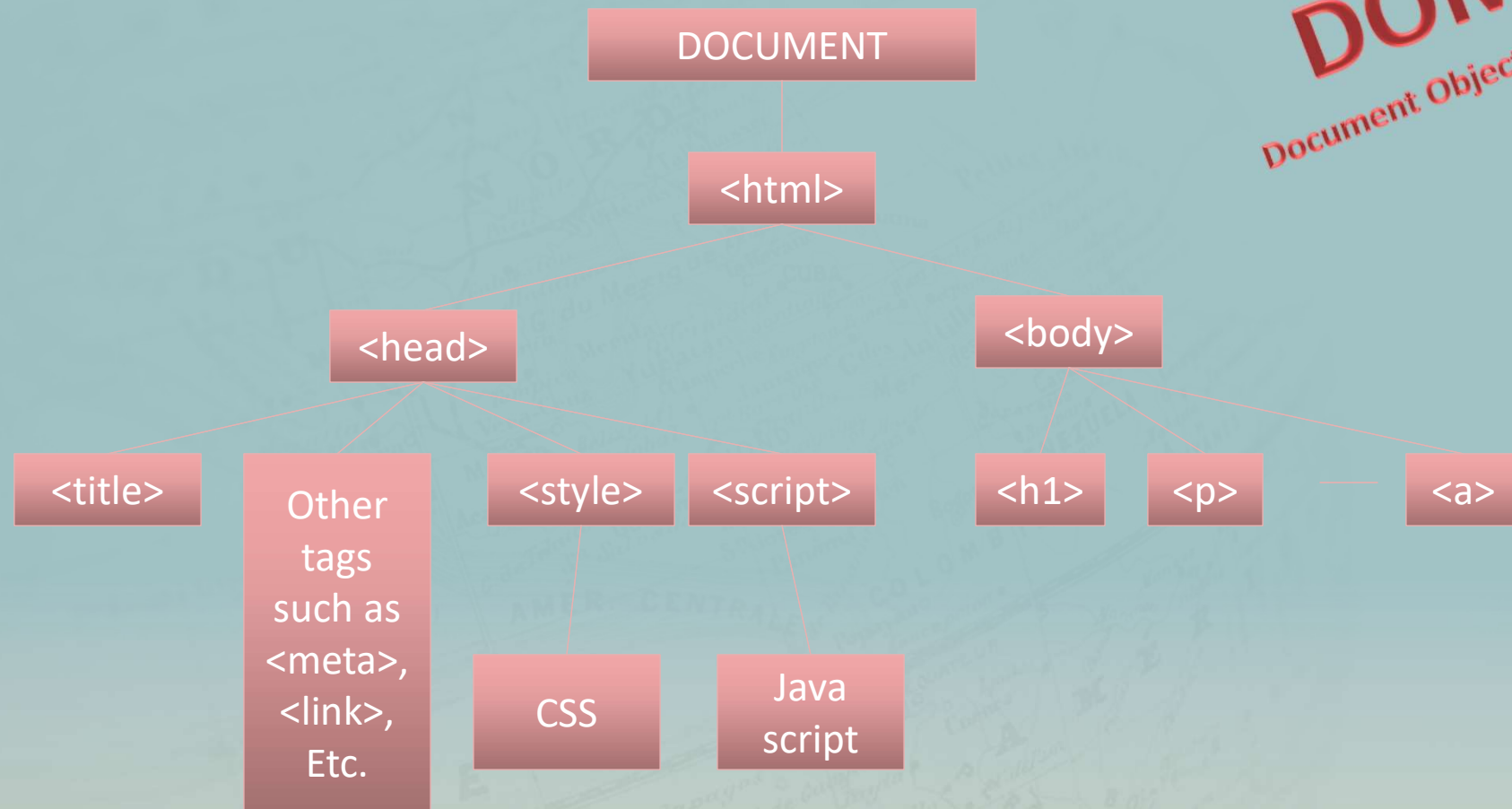
- **Document Object Model (DOM)**

- The Document Object Model (DOM) is an application programming interface (API) for HTML as well as XML.
- The DOM organizes the entire web page as a document composed of a hierarchy of nodes like a tree structure and using the DOM API, nodes can be removed, added, and replaced.
- DOM allows the developer to manipulate the document.

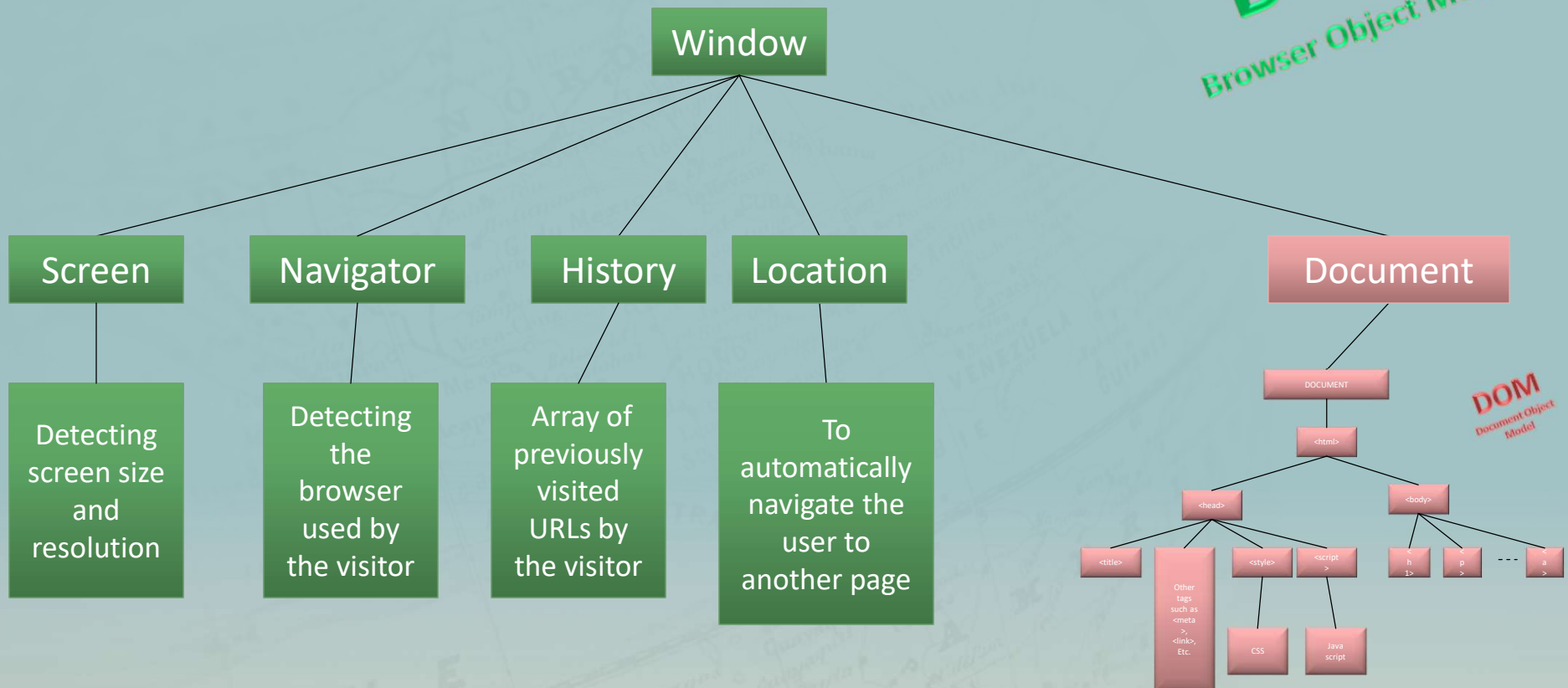
- **Browser Object Model (BOM)**

- Browsers feature a Browser Object Model (BOM) that allows access and manipulation of the browser window. For example (browser history, location, navigator, and screen)
- Because no standards exist for the BOM, each browser has its own implementation.
- BOM allows the developer to manipulate the browser window.

THE STRUCTURE OF AN HTML DOCUMENT



THE STRUCTURE OF BROWSER OBJECT



<http://www.javascriptkit.com/jsref/window.shtml>

INTERACTING WITH BOM

- **Screen:** contains information about the visitor's screen.
 - (height, width, colorDepth, etc)
- **Window:** represents an open window in a browser.
 - (open, close, name, length, width, parent, screenleft, screentop, alert(), prompt(), print(), moveTo(),moveBy(), scrollBy(), scrollTo(), blur(), setTimeout(), etc)
- **Navigator:** contains information about the browser.
 - (appName, CookiesEnabled, JavaEnabled, etc)
- **History:** contains the URLs visited by the user (within a browser window).
 - (length, back(), forward(), go(), etc.)
- **Location:** contains information about the current URL.
 - (host, hostname, href, port, protocol, assign(), reload(), replace())

http://www.w3schools.com/jsref/obj_window.asp

INTERACTING WITH DOM

- The Document object is the root of a document tree. It gives us access to the document's data elements.
- Since element nodes, text nodes, attributes, comments, etc. cannot exist outside the document, the Document object contains methods to create and access these objects.
- **Properties:** doctype, documentURI, inputEncoding, xmlEncoding, anchors, forms, images, links, referrer, title, URL, domain, lastmodified, etc.
- **Methods:** open(), close(), write(), writeln(), getElementByName(), getElementById(), getElementByTagName(), renameNode(), setAttribute(), getAttribute(), removeAttribute(), etc.
- **Events:** load, unload, resize, scroll, click, dblclick, mousedown, mouseup, mouseover, mouseover, mouseout, keydown, keyup, keypress, etc.

http://en.wikipedia.org/wiki/DOM_events

http://www.w3schools.com/jsref/dom_obj_node.asp

SCREEN / MONITOR OBJECT

BOM
Browser Object Model

```
document.write("<b>Total Width of the screen (including the Windows Taskbar):</b> " + screen.width + "<br />");  
document.write("<b>Total Height of the screen (including the Windows Taskbar):</b> " + screen.height + "<br />");
```

```
document.write("<b>Width of the screen (excluding the Windows Taskbar):</b> " + screen.availWidth + "<br />");  
document.write("<b>Height of the screen (excluding the Windows Taskbar):</b> " + screen.availHeight + "<br />");
```

```
document.write("<b>Bit depth of the color palette for displaying images:</b> " + screen.colorDepth + "<br />");
```



WINDOW /BROWSER OBJECT

BOM
Browser Object Model

```
document.write("<b>X coordinate of the window relative to the screen:</b> " + window.screenX + "<br />");  
document.write("<b>Y coordinate of the window relative to the screen:</b> " + window.screenY + "<br />");  
document.write("<b>Inner Width of a window's content area:</b> " + window.innerWidth + "<br />");  
document.write("<b>Inner Height of a window's content area:</b> " + window.innerHeight + "<br />");  
document.write("<b>Outer width of a window, including toolbars/scrollbars:</b> " + window.outerWidth + "<br />");  
document.write("<b>Outer height of a window, including toolbars/scrollbars:</b> " + window.outerHeight + "<br />");  
document.write("<b>Current URL:</b> " + window.location + "<br />");
```



WINDOW OBJECT (CONTINUED...)

- **To create a new Browser Window:**

- `var myWin = window.open('', '_blank', 'width=100, height=400');` //make sure the pop-up blocker is off

- **To write to that window:**

- `myWin.document.write("Hello");`

- **To position and size the window the window:**

- `myWin.moveTo(0,0);` // not consistent (works in IE)
 - `myWin.resizeTo(500,500);` // not consistent (works in IE)

- **To focus on the window:**

- `myWin.focus();` // not consistent (works in IE)

- **Creating a Timer Object: (to do something at a later time)**

- `myWin.document.write("....close in 15 seconds...");`
 - `myWin.setTimeout(function() {myWin.close()}, 15000);` // close window after 15 seconds

- **Creating a Timer Object: (to repeat something on regular intervals)**

- `myWin.setInterval(function() {displayClock(myWin)}, 1000);` // run the display clock function every second.

WINDOW OBJECT (CONTINUED...)

- **To Scroll the Browser Window:**

- `window.scrollTo(x,y);`
- This may be useful if we want to move the browser to a specific location on screen after an operation has reloaded the page.
- Alternatively, if a number of steps have to be skipped in a page, and we know how many pixels each step is we could try to skip by that many pixels in the “y” direction.



CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

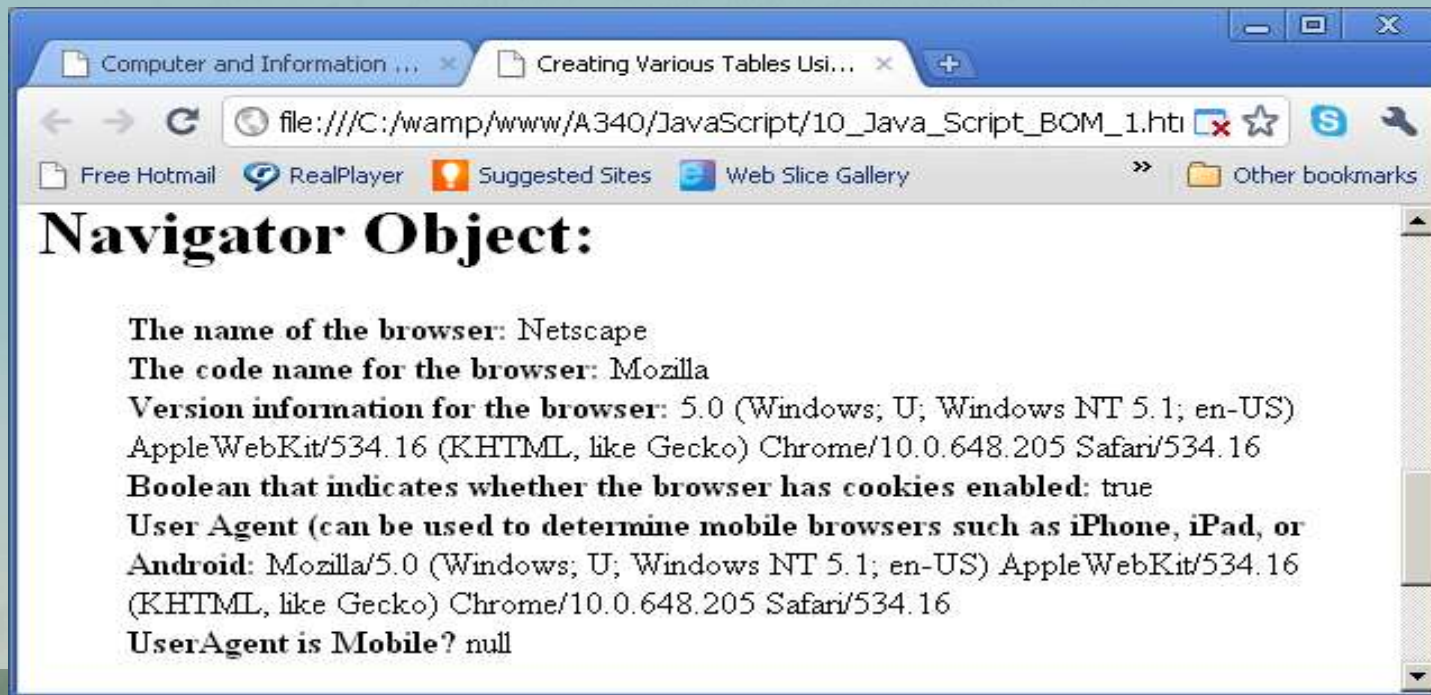
Java
Script
E1

NAVIGATOR OBJECT

BOM
Browser Object Model

```
document.write("<b>The name of the browser:</b> " + navigator.appName + "<br />");
document.write("<b>The code name for the browser:</b> " + navigator.appCodeName + "<br />");
document.write("<b>Version information for the browser:</b> " + navigator.appVersion + "<br />");
document.write("<b>Boolean that indicates whether the browser has cookies enabled:</b> " + navigator.cookieEnabled + "<br />");
document.write("<b>User Agent (can be used to determine mobile browsers such as iPhone, iPad, or Android:</b> " +
navigator.userAgent + "<br />");          //returns true if user is using one of the following mobile browsers

var isMobile=navigator.userAgent.match(/(iPad)|(iPhone)|(iPod)|(android)|(webOS)/i);
document.write("<b>UserAgent is Mobile?</b> " + isMobile + "<br />");
```



HISTORY OBJECT

BOM
Browser Object Model

- To get the length:

- document.write("Number of URLs in History object: " + history.length + "
");

- To move Backward or Forward:

- history.back();
 - history.forward();

```
function goBack()
{
    window.history.go(-1);
}
function goForward()
{
    window.history.go(+1);
}
```

<body>

<input type="button" value="Go Back" onclick="goBack()">

<input type="button" value="Go Forward" onclick="goForward()">

NOTE: (Forward and Backward will only work if there is something in the history!)

</body>

History Object:

Number of URLs in History object: 1

Go Back

Go Forward

(Forward may not do anything here since there may be no links in the forward history!)

LOCATION OBJECT

```
document.write("<b>URL of the current page:</b> " + location.href + "<br />");  
document.write("<b>Show the Host:</b> " + location.host + "<br />");  
document.write("<b>Show the Hostname:</b> " + location.hostname + "<br />");  
document.write("<b>Show the Protocol (FILE, HTTP, HTTPS, etc):</b> " +  
                location.protocol + "<br />");
```

```
document.write("<b>Show the Port number:</b> " + location.port + "<br />");
```

```
document.write("<b>Reload the current page:</b> (Notice this should be controlled by an IF  
statement or an Event handler.) " + "<br />");  
//location.reload();
```

```
document.write("<b>Replace the URL of the current page with another URL:</b> " + "<br />");  
//location.replace('http://www.iusb.edu');
```

Location: Object:

```
URL of the current page: file:///C:/wamp/www/A340/JavaScript/10_Java_Script_BOM_1.html  
Show the Host:  
Show the Hostname:  
Show the Protocol (FILE, HTTP, HTTPS, etc): file:  
Show the Port number:  
Reload the current page: (Notice this should be controlled by an IF or an Event handler.)  
Replace the URL of the current page with another URL:
```

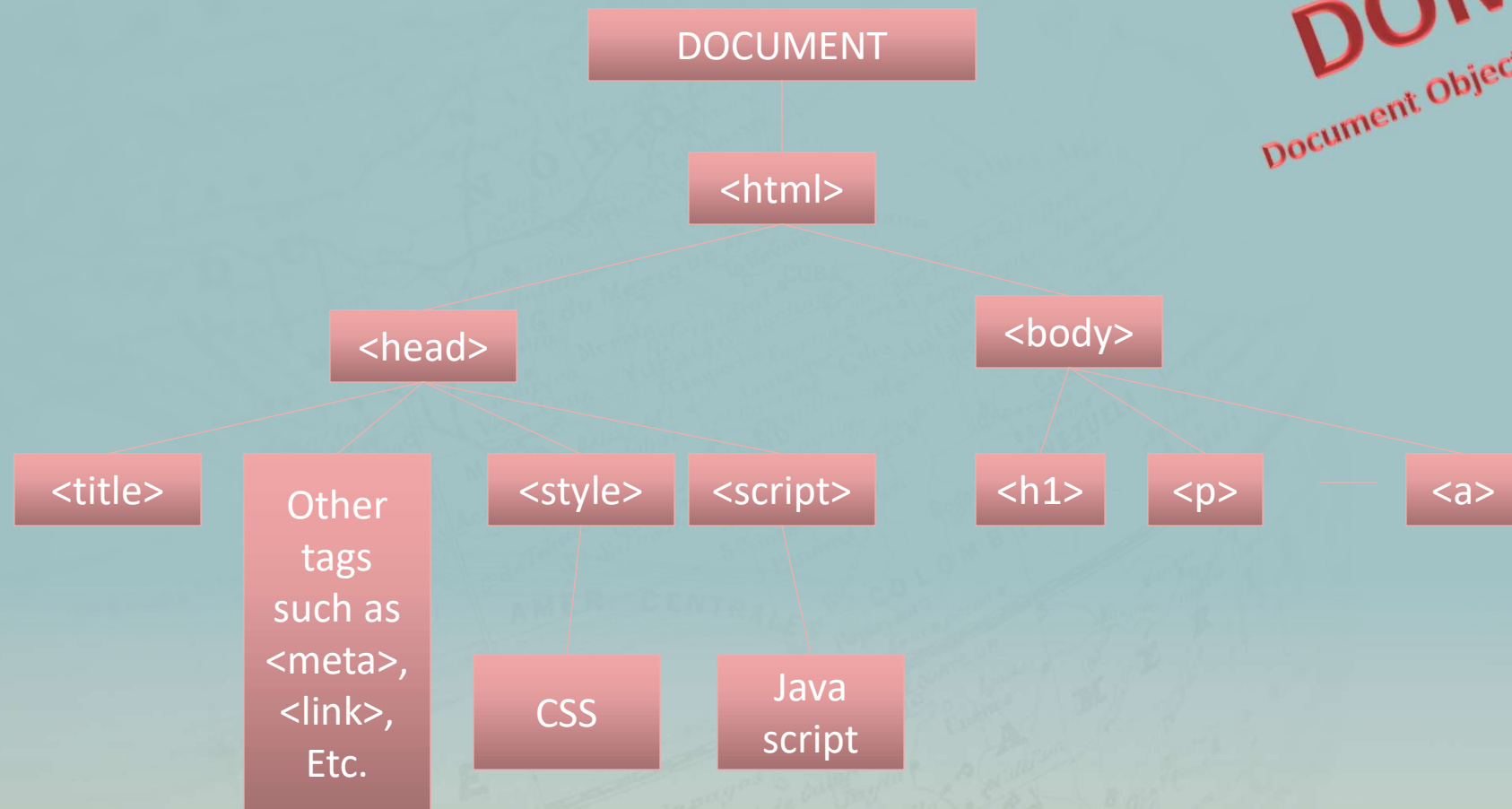


CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

Java
Script
F

THE STRUCTURE OF AN HTML DOCUMENT



DOM
Document Object Model

INTERACTING WITH DOM

- The Document object is the root of a document tree. It gives us access to the document's data elements.
- Since element nodes, text nodes, attributes, comments, etc. cannot exist outside the document, the Document object contains methods to create and access these objects.
- **Properties:** doctype, documentURI, inputEncoding, xmlEncoding, anchors, forms, images, links, referrer, title, URL, domain, lastmodified, etc.
- **Methods:** open(), close(), write(), writeln(), getElementByName(), getElementById(), getElementByTagName(), renameNode(), setAttribute(), getAttribute(), removeAttribute(), etc.
- **Events:** load, unload, resize, scroll, click, dblclick, mousedown, mouseup, mouseover, mouseover, mouseout, keydown, keyup, keypress, etc.

http://en.wikipedia.org/wiki/DOM_events

http://www.w3schools.com/jsref/dom_obj_node.asp

DOM OBJECT

- `document.getElementById ()`



DOM OBJECT

```
var checkAllBtn = document.getElementById("ckallbtn");  
checkAllBtn.value = "Uncheck All";
```



AUTOMATICALLY CHECK THE CHECKBOXES

```
<!DOCTYPE html>
<html>
<head>

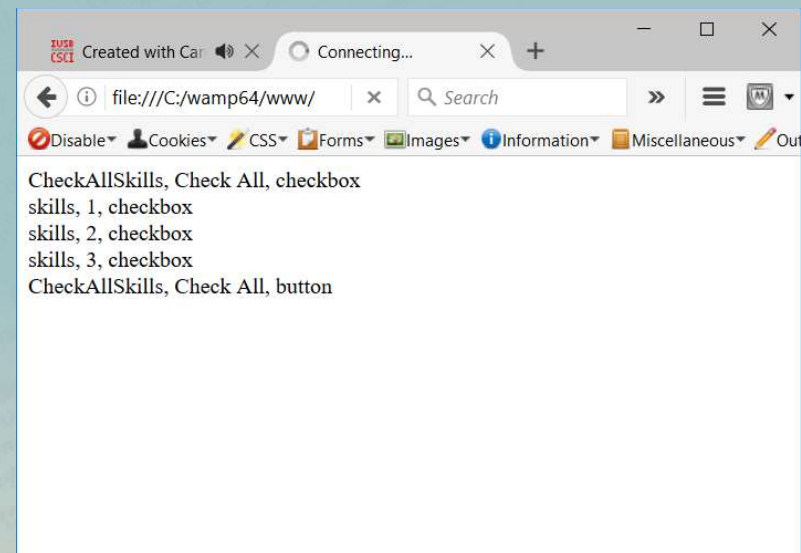
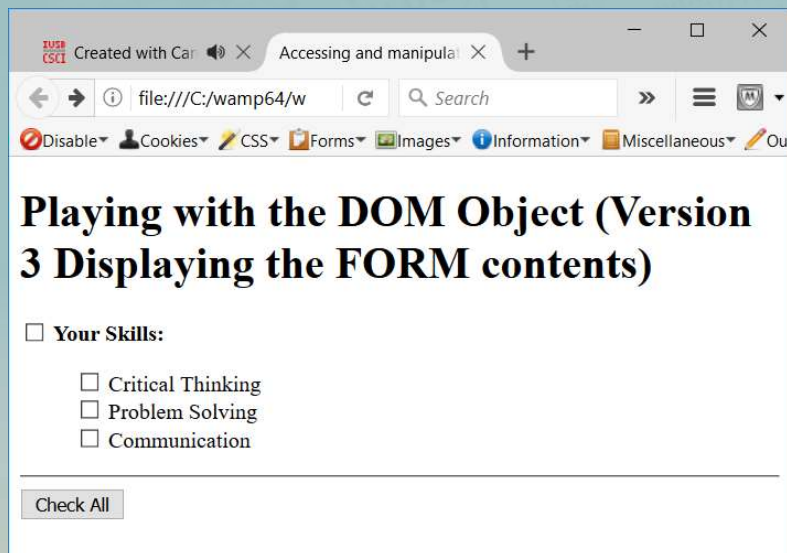
  <title>Accessing and manipulating DOM objects in JavaScript</title>
  <meta charset="UTF-8">
  <meta name="Keywords" content="DOM Object, Java Script, CSCI-A340, Web Programming" >

  <script type="text/javascript">
    function checkAllCheckBoxes()
    {
      for (var i=1; i <=3; i++)
      {
        // Get the checkbox object.
        var obj = document.getElementById("ck" + i );
        obj.checked = true;
      }
    }
  </script>
</head>
<body>
  <h1>Playing with the DOM Object (Version 1: No capability to uncheck!)</h1>
  <form action= "#">
    <legend><b> Your Skills: </b> </legend>
    <ul>
      <input id="ck1" type="checkbox" name="skills" value="1"> Critical Thinking<br>
      <input id="ck2" type="checkbox" name="skills" value="2"> Problem Solving <br>
      <input id="ck3" type="checkbox" name="skills" value="3"> Communication <br>
    </ul>
    <hr>
    <input id="ckallbtn" type="button" value="Check All" onclick="checkAllCheckBoxes()">
  </form>
</body>
</html>
```

NAVIGATING THE DOM OBJECT

```
// Get the form object  
var theForm = document.getElementById("form1");
```

```
// Get the elements within the form  
var theElements = theForm.elements;
```



NAVIGATING THE DOM OBJECT

```
<script type="text/javascript">
```

```
function checkAllCheckBoxes()  
{
```

```
    // Get the form object
```

```
    var theForm = document.getElementById("form1");
```

```
    // Get the elements within the form
```

```
    var theElements = theForm.elements;
```

```
    for (var i=0; i<theElements.length; i++)
```

```
    {
```

```
        document.write(theElements[i].name);
```

```
        document.write(", ");
```

```
        document.write(theElements[i].value);
```

```
        document.write(", ");
```

```
        document.write(theElements[i].type);
```

```
        document.write("<br>");
```

```
    }
```

```
}
```

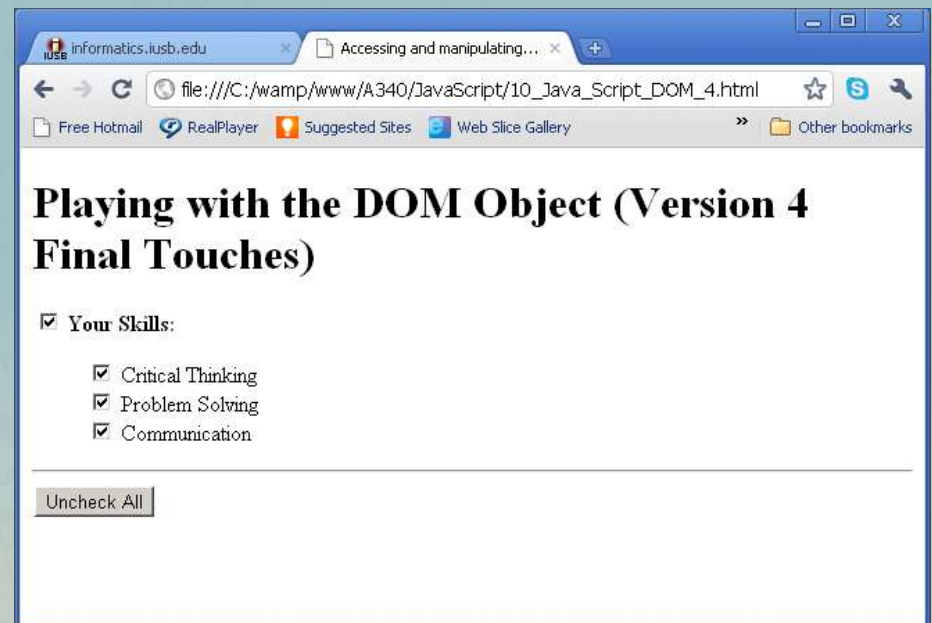
```
</script>
```

Java_Script_DOM_3_Checkbox.html

DOM OBJECT

```
// Get the form object  
var theForm = document.getElementById("form1");
```

```
// Get the elements within the form  
var theElements = theForm.elements;
```



Java_Script_DOM_4_Checkbox.html

```
<script type="text/javascript">

    function checkAllCheckBoxes()
    {
        // Get the form object
        var theForm = document.getElementById("form1");

        // Get the elements within the form
        var theElements = theForm.elements;

        // Get the CheckAll button object
        var checkAllBtn = document.getElementById("ckallbtn");

        var checkAllChkBox = document.getElementById("ckallcheckboxes");

        for (var i=1; i <=3; i++)
        {
            // Get the checkbox object.
            obj = theElements[i];
            if (checkAllBtn.value == "Check All" )
                obj.checked=true;
            else
                obj.checked=false;
        }

        // Change the button value
        if (checkAllBtn.value == "Check All" )
        {
            checkAllBtn.value = "Uncheck All";
            checkAllChkBox.checked = true;
        }
        else
        {
            checkAllBtn.value = "Check All";
            checkAllChkBox.checked = false;
        }
    }

</script>
```

DOM OBJECT

- Two Dimensional Arrays
- Changing the internal attributes of an HTML table.
- Changing the background color, changing the format.

The image displays three sequential browser screenshots of a web page titled "Accessing and manipulating..." at the URL "file:///C:/wamp/www/A340/JavaScript/10_Java_Script_DOM_5.html". Each screenshot shows a 10x10 table of numbers. The first screenshot shows the initial state with a blue background. The second screenshot shows the table with a pink background. The third screenshot shows the table with a pink background and some numbers highlighted in red. Below the table, there are two buttons: "Change Background Color" and "Show Primes".

Step1: Load a 2 Dimensional Array
Step2: Display Array Contents

423	895	652	718	41	532	830	176	951	23
176	214	581	926	535	637	567	11	524	941
989	223	680	57	921	495	660	903	529	368
699	189	876	853	139	862	594	830	83	927
715	915	944	107	504	637	652	522	149	136
465	834	702	196	699	451	895	103	579	402
162	248	880	194	363	638	727	700	674	249
732	532	374	521	966	756	295	803	504	594
64	102	85	671	338	600	329	920	803	293
753	633	49	638	960	950	799	413	337	327

Step 3: Playing with the DOM Object
(Dynamically changing the content of the array)

Change Background Color | Dynamically change the background color of the cells in the table.
Show Primes | Go through the array and identify the primes numbers in the table.

Step1: Load a 2 Dimensional Array
Step2: Display Array Contents

423	895	652	718	41	532	830	176	951	23
176	214	581	926	535	637	567	11	524	941
989	223	680	57	921	495	660	903	529	368
699	189	876	853	139	862	594	830	83	927
715	915	944	107	504	637	652	522	149	136
465	834	702	196	699	451	895	103	579	402
162	248	880	194	363	638	727	700	674	249
732	532	374	521	966	756	295	803	504	594
64	102	85	671	338	600	329	920	803	293
753	633	49	638	960	950	799	413	337	327

Step 3: Playing with the DOM Object
(Dynamically changing the content of the array)

Change Background Color | Dynamically change the background color of the cells in the table.
Show Primes | Go through the array and identify the primes numbers in the table.

Step1: Load a 2 Dimensional Array
Step2: Display Array Contents

423	895	652	718	41	532	830	176	951	23
176	214	581	926	535	637	567	11	524	941
989	223	680	57	921	495	660	903	529	368
699	189	876	853	139	862	594	830	83	927
715	915	944	107	504	637	652	522	149	136
465	834	702	196	699	451	895	103	579	402
162	248	880	194	363	638	727	700	674	249
732	532	374	521	966	756	295	803	504	594
64	102	85	671	338	600	329	920	803	293
753	633	49	638	960	950	799	413	337	327

Step 3: Playing with the DOM Object
(Dynamically changing the content of the array)

Change Background Color | Dynamically change the background color of the cells in the table.
Show Primes | Go through the array and identify the primes numbers in the table.



CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

Java
Script
G

USER DEFINED OBJECTS

- Javascript objects are slightly different than what you are used to in C++
- There is no notion of a Class! Inheritance!, etc. (unless you use **Typescript**) more later!
- There are several ways that one can create a javascript object, and that is part of the problem!

CREATING AND USING AN OBJECT

EXAMPLE 1

// Creating a person object using the keyword (Object) (For single use only)

```
var person = new Object();
    person.firstName = "John";
    person.lastName = "Doe";
    person.age = 50;
    person.eyeColor = "blue";
    person.daysLived = function() {return(person.age * 365)};
    person.changeLastName = function(newLastName) {
        person.lastName = newLastName;
        alert("changing last name to: " + person.lastName);
    }
```

```
document.writeln("person.firstName: " + person.firstName + "<br>");
document.writeln("person.lastName: " + person.lastName + "<br>");
document.writeln("person.daysLived: " + person.daysLived() + "<br>");
person.changeLastName("Rally");
document.writeln("person.lastName: " + person.lastName + "<br>");
```

CREATING AND USING AN OBJECT

EXAMPLE 2

```
// Creating a Function Objects  
// This example allows us to create multiple objects of type person  
// using the function constructor.  
// Using this method of object creation, we can not create a static object!
```

```
function person(first, last, age, eye) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eye;  
    this.daysLived = function() {return(age * 365);};  
    this.changeLastName = function(newLastName) {  
        this.lastName = newLastName;  
        alert("changing last name to: " + this.lastName);  
    }  
}
```

```
var myFather = new person("John", "Doe", 50, "blue");  
var myMother = new person("Sally", "Rally", 48, "green");  
// var myBrother = person("Tom", "Rally", 48, "green"); // Can not create a static person object  
  
document.writeln("myFather.firstname: " + myFather.firstName + "<br>")  
document.writeln("myMother.firstname: " + myMother.firstName + "<br>")  
document.writeln("myMother.age: " + myMother.age + "<br>")  
document.writeln("myMother.daysLived: " + myMother.daysLived() + "<br>")  
myFather.changeLastName("Rally");
```

CREATING AND USING AN OBJECT

EXAMPLE 3

```
//-----  
// Create a Class called Car (with some properties and functions)  
// My approach to creating a javascript class is to create the class  
// as a variable with properties and methods. Then create a constructor  
// function for the class which initializes all of the properties as well  
// as its methods. See below. Note that the methods are initially empty,  
// they are initialized in the constructor. I tried it the other way  
// (initialized them outside of the constructor function), and  
// it did not work. Not sure why!  
//  
// This approach allows for both statically and dynamically instantiating  
// a Car object, and both approaches work.  
var Car = {  
    make: "",  
    model: "",  
    year: 0,  
    getMake: "", // this is a method  
    getModel: "", // this is a method  
    constructor: function(the_make, the_model, the_year) {  
        this.make = the_make;  
        this.model = the_model;  
        this.year = the_year;  
        this.getMake=function(){return(this.make)};  
        this.getModel=function(){return(this.model)};  
    }  
};
```

// Statically allocating a javascript object

```
var myCar = Car;  
myCar.constructor("honda", "accord", 2016);  
document.writeln("myCar.make: " + myCar.make + "<br>");  
document.writeln("myCar.model: " + myCar.model + "<br>");  
document.writeln("myCar.year: " + myCar.year + "<br>");  
  
document.writeln("myCar.getMake(): " + myCar.getMake() + "<br>");  
document.writeln("myCar.getModel(): " + myCar.getModel() + "<br>");
```

// Dynamically allocating an object

```
var myNewCar = new Car.constructor("Ford", "Model-T", 1920);  
document.writeln(myNewCar.make + "<br>");  
document.writeln(myNewCar.model + "<br>");  
document.writeln(myNewCar.year + "<br>");  
  
// Note the difference between the next two lines:  
//document.writeln(myNewCar.getMake + "<br>");  
document.writeln(myNewCar.getMake() + "<br>");  
document.writeln(myNewCar.getModel() + "<br>");
```



CLIENT-SERVER WEB DEVELOPMENT

Dr. Hossein Hakimzadeh
Department of Computer and Information Sciences
Indiana University
South Bend, IN

Java
Script
G

CREATING DOM ELEMENTS DYNAMICALLY

1. Start with a blank web page
2. Add a style sheet
3. Modify the style sheet (Dynamically)
4. Add other element (divs, img, etc.)
5. Add event handlers for those elements

STEP 1

- Start with a page like this:

```
<!DOCTYPE html>
<html>
<head>
  <script>

  </script>
</head>

<body>

</body>

</html>
```

STEP 2

- Assume we have a CSS file as shown below and we would like to replace the CSS with another file Dynamically:

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="old_style.css"/>

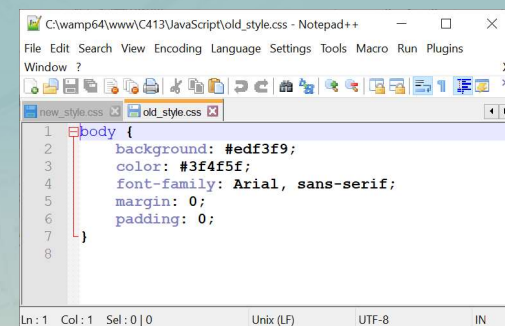
    <script type="text/javascript">

    </script>

</head>

<body>

</body>
</html>
```



STEP 2 - CODE

- Place the following code under the `<script>` tag.

```
// Assuming a <link> element exists, find it and modify its content.  
// a typical <link> element: <link rel="stylesheet" type="text/css" href="style.css"/>  
  
function changeCSS(cssFile) {  
    // Get the first link element which holds the information about the first included CSS file.  
    var thelink = document.getElementsByTagName("link").item(0);  
  
    thelink.setAttribute("rel", "stylesheet");  
    thelink.setAttribute("type", "text/css");  
    thelink.setAttribute("href", cssFile);  
}
```

STEP 2 - CODE

- After placing the code, we create a button to call it, and we pass the name of the new CSS file to it:

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="old_style.css"/>

    <script type="text/javascript">

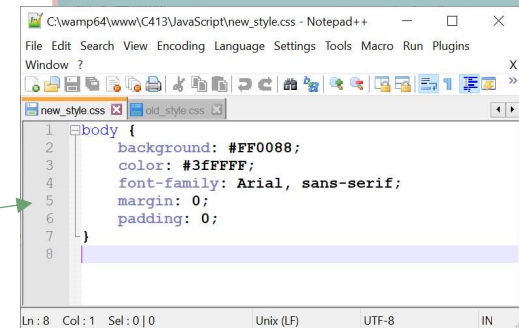
        // Assuming a <link> element exists, find it and modify its content.
        // a typical <link> element: <link rel="stylesheet" type="text/css" href="style.css"/>
        function changeCSS(cssFile) {

            // Get the first link element which holds the information about the first included CSS file.
            var thelink = document.getElementsByTagName("link").item(0);

            thelink.setAttribute("rel", "stylesheet");
            thelink.setAttribute("type", "text/css");
            thelink.setAttribute("href", cssFile);

        }

    </script>
</head>
<body>
    <button onclick="changeCSS('new_style.css');"> Click Here to dynamically change the CSS </button>
</body>
</html>
```



STEP 3

- Now let's add some more **content** to the file.
(couple of divs and an image tag)

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="style.css"/>

    <script type="text/javascript">

    </script>
</head>

<body>

<button onclick="changeCSS('new_style.css');"> Click Here to dynamically change the CSS </button>

<div id="div1" style="width:80px; height:80px; background-color:black; color:yellow;" onclick="displayEventInfo(event);">
    Click me to see the event information.
</div>

<img id="pic1"
    src='../images/JS_slides/Slide1.JPG'
    width = '90'
    height= '80'
    onclick="displayImage(event);"/>

<div id="div2">

    <p> Placeholder for the large image. (click on the image above..) </p>
</div>

</body>
</html>
```

STEP 3 - CODE

- Note that the first div and the img tags have **event handlers** attached to them:

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="style.css"/>

    <script type="text/javascript">
    ....
    </script>
</head>

<body>

<button onclick="changeCSS('new_style.css');"> Click Here to dynamically change the CSS </button>

<div id="div1" style="width:80px; height:80px; background-color:black; color:yellow;" onclick="displayEventInfo(event);">
    Click me to see the event information..
</div>

<img id="pic1"
    src='../images/JS_slides/Slide1.JPG'
    width = '90'
    height= '80'
    onclick="displayImage(event);"/>

<div id="div2">

    <p> Placeholder for the large image. (click on the image above..) </p>
</div>

</body>
</html>
```

STEP 3 – CODE / EVENT HANDLERS

- Add the following event handlers in the <script> section. (https://www.w3schools.com/jsref/obj_event.asp)

```
function displayEventInfo(event) {  
    // the event parameter passed to this function corresponds to the event that just happened.  
    // in this case the "click" event.  
    // we can use the properties of the event object to extract information about which object triggered the event.  
    // see below:  
    alert("Event target : " + event.target); //Returns the element that triggered the event  
    alert("Event timestamp: " + event.timeStamp); //Returns the time (in milliseconds relative to the epoch) at which the event was created  
    alert("Event Type/Name: " + event.type); //Returns the name/type of the event  
  
    // Furthermore, we can extract information about the target of the event. see below:  
  
    // if target is an image:  
    alert(event.target.src); // display the image file name  
    // if target is a div:  
    alert(event.target.id); // display its id.  
}  
  
function displayImage(event) {  
    // the event parameter passed to this function corresponds to the event that just happened.  
    // in this case the "click" event.  
    // we can use the properties of the event object to extract information about which object triggered the event.  
    // see below:  
  
    displayEventInfo(event);  
    var newContent = '';  
  
    document.getElementById('div2').innerHTML = newContent;  
}
```

RESULTS:

