Web Technologies I

Lecture # 7 : JAVASCRIPT

Where are we now?

- HTML and CSS
- Client-side Scripting
  - JavaScript
- Server-side Scripting
  - PHP
- Advanced Scripting
  - Ajax

What You Can Do with JavaScript

- *Validate form fields*
  - Validate form input before submitting the contents to the server. This saves time and server resources, and provides immediate feedback.
- *Set and retrieve web cookies*
  - Persist information such as usernames, account numbers, or preferences in a controlled, safe environment saving users time the next time they access a site.
- *Dynamically alter the appearance of a page element*
  - Provide feedback by highlighting incorrect form entries; increase the size of a section's font based on the reader's request.
- *Hide and show elements*
  - Based on personal preference or user actions, show or hide page content, such as form elements, expanding writing, and changing the displayed size of an image.

What You Can Do with JavaScript

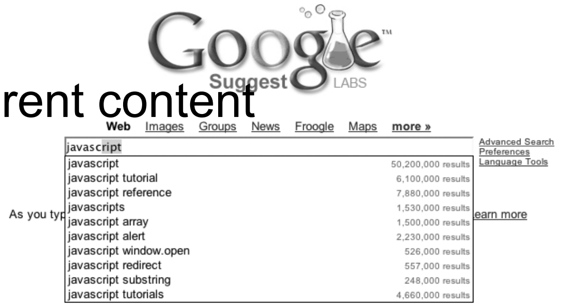
- *Move elements about the page*
  - Create a drop-down menu, or provide an animated cursor to accent page elements.
- *Capture user events and adjust the page accordingly*
  - Based on keyboard or mouse actions, show some popup message.
- *Scroll content*
  - For larger images or content areas, provide a way to grab the element with a mouse or keyboard, and scroll it right or left, up or down.
- *Interface with a server-side application without leaving the page*
  - This is the basis of Ajax and is used to populate selection lists, update data, and refresh a display all without having to reload the page. This helps eliminate round trips to the server,

JavaScript

- Program code is left in a text format, and interpreted “on the fly,” (as opposed to a compiled language, in which the program is compiled into binary code).
- Client side JavaScript is interpreted by a JavaScript aware browser.
- Developed by Netscape (Microsoft has a version called Jscript)
- Syntax similar to C++ and Java
- JavaScript is not Java!
- Object oriented
- Dynamically typed (you don’t have to declare a variable type such as integer, floating point, or text. Type is determined by content and context). This is called coercion.

Defined

- Scripting language
- Gives developer control over pages
  - Change content
    - Image rollovers
    - Validate content of a form
    - Make calculations
    - Detect the browser in use and display different content
    - Dynamic form generation
  - Alert and Redirect user
    - Pop-up Windows
    - Detect a plug-in and notify user if plug-in is installed
  - Loops
    - Perform a repetitive task – display a series of pictures
- JavaScript is Case sensitive!



javascript

javascript	50,200,000 results
javascript tutorial	6,100,000 results
javascript reference	7,880,000 results
javascripts	1,530,000 results
javascript array	1,500,000 results
javascript alert	2,230,000 results
javascript window.open	526,000 results
javascript redirect	557,000 results
javascript substring	248,000 results
javascript tutorials	4,660,000 results

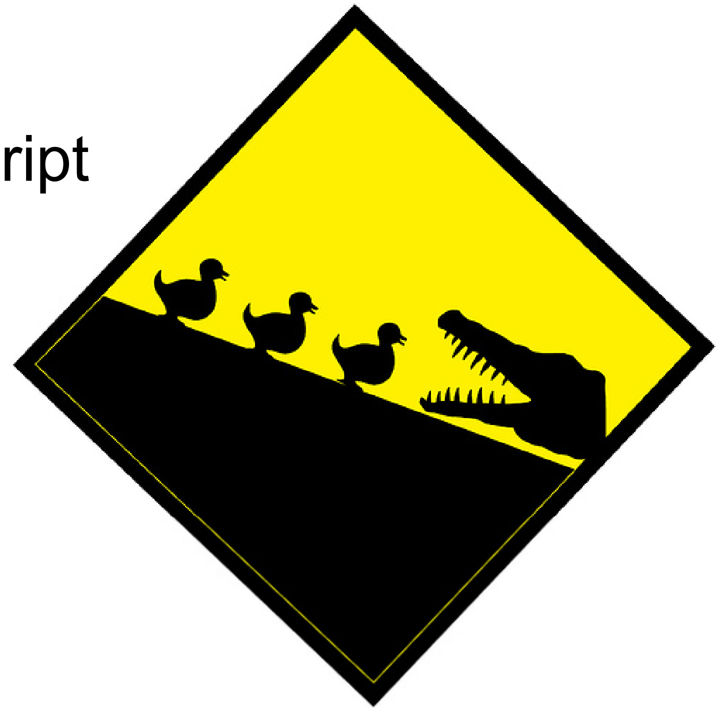
What is DOM?

- Document Object Model (DOM)
  - "The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents."

- You can access any HTML element and perform any action on it through JavaScript using that elements name or id.

Document Object Model

- An object could be anything (e.g. a Web page)
- Objects perform methods (show an image)
- Methods have properties which can be modified
  - Image
    - File
    - Height/Width
- Manipulate properties with JavaScript



How to use JavaScript

- Use the <SCRIPT> tag

```
<html>
<body>
<script type="text/javascript">
    document.write("Hello World!");
</script>
</body>
</html>
```

- Insert into your Web page
  - Within <HEAD> tags
  - Within <BODY> tags

Where to use JavaScript

- Insert into your Web page
  - Within <HEAD> tags
  - Within <BODY> tags
- JavaScripts in the **body** section
  - will be executed WHILE the page loads.
- JavaScripts in the **head** section
  - will be executed when CALLED.
- Using external JavaScripts
  - Sometimes you might want to run same JavaScript on several pages, without having to write the same script on every page.

```
<html>
<head><script type="text/javascript" src="file.js"></script>
</head>
<body> </body>
</html>
```

Adding JavaScript to HTML

- **Embedded Script**

Script in Head Section

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

Script in Body Section

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
</html>
```

- **External File**

```
<script language="JavaScript"
type="text/javascript" src="file.js" ></script>
```

  – The external script cannot contain the <script> tag!

Embedding a Javascript

- Script statements should be in contiguous lines, ending with a carriage return or semicolon
- More than one statements on a single line must be separated with semi-colons
- Best practice is to use one statement per line, and end line with semi-colon
- See Example Below
  - Document is an object of current webpage
  - Write is a method (you can tell that by the parentheses)

```
<SCRIPT LANGUAGE=JavaScript>
document.write("Hello
world!<BR>");
</SCRIPT>
```

External Scripts

- External JavaScript need not be in the html page on which they will be displayed
- By convention, external scripts have extensions .js
- They contain no raw HTML (although you can used a document.write to pass HTML to the browser)
- Use a src statement to pull it into the html page:

<SCRIPT LANGUAGE="JavaScript“ src="datefile.js">

Javascript Comments

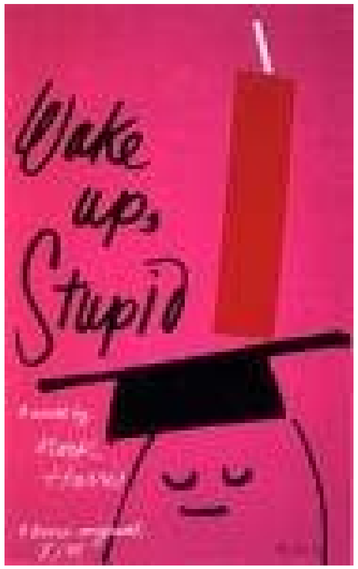
- Single line comments start with //
  //this is a comment
- Multiple line comments start with /* and end with */
  /*This is a multiple line comment so you can drone on and on and on as much as you care to*/

Hiding Javascripts

- Some browsers don’t understand Javascript, and display the code.
- You can use HTML comments to hide Javascripts:

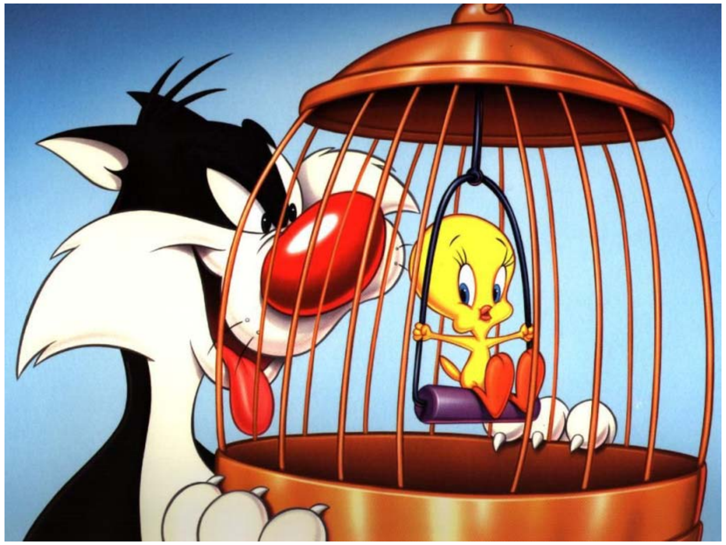
-
  ```
  <SCRIPT LANGUAGE=“Javascript”>
  <!-- Hide your script
  Script script script
  //Stop hiding now -->
  </SCRIPT>
  ```
- The noscript tag allows display for non-javascript aware browsers:
  ```
  <NOSCRIPT>
  You need javascript to read this page
  </NOSCRIPT>
  ```

JavaScript: The language



Javascript rules

- It is (generally) case sensitive
- Separate statements.
  - Individual lines
  - Semicolons (eg. A = 2; B =3
- Reserved word cannot be used as identifiers or variables: e.g. break, do, if, else….



Javascript Variables

- Cannot begin with a number
- Cannot be a reserved word
- Can only contain letters, numbers or underscores
- Should be declared by var statement (you can get away without it sometimes, but it’s better to do it as a matter of habit).

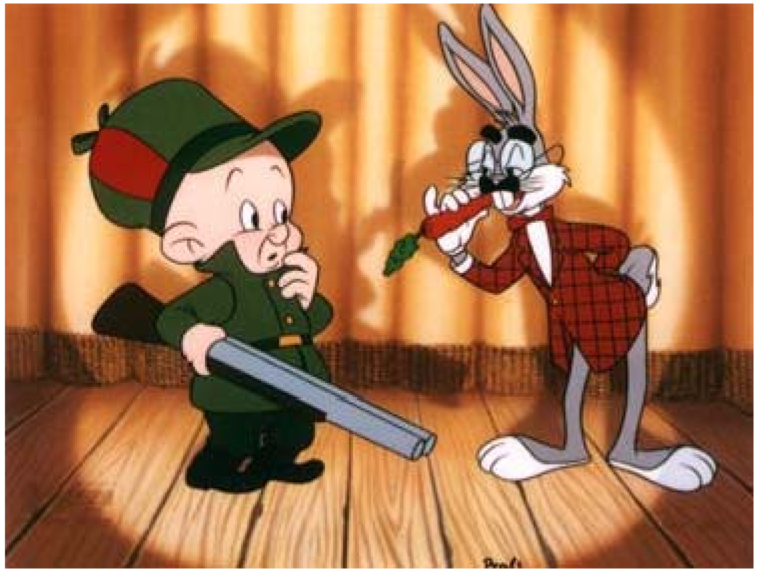
Coercion

- A variable of one type can be used as if it were another.
- If there's a conflict, javascript doesn't produce an exception
  - string+number goes to strings
  - boolean+string goes to strings
  - number+boolean goes to numbers

  - Explicit conversions
    - string to an integer, use the parseInt() method.
    - string to a number, use the parseFloat() method.

Variables and Data Types

- JavaScript is a loosely typed language
  - Data types are converted during execution as needed
  - Data typing only matters during operations
    - “6” + “67” = “667” String
    - 6 + 67 = 73



Variables and Data Types

- Numbers
  - Integer and floating-point numbers.
- Booleans
  - True or false. Can not use as 1 or 0 but 0 = false;
- Strings
  - Anything surrounded by “” (double quotes) or ‘’ (single quotes)
    e.g. “My String” = ‘My String’
- Object
  - myObj = new Object();
- Null
  - Not the same as zero - no value at all.
- Undefined
  - The Variable has been created but no value has been assigned to it

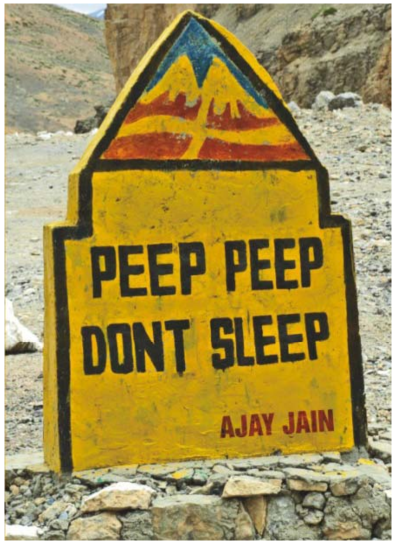
Operators

- Arithmetic (the usual suspects)

  +,-,*,/,%,--,++
- Comparison

  ==, !=, >, >=, <, <=
- Boolean

  &&, ||, !



Statements

- Conditionals

  -
    ```
    if(x < 0){
      alert("x is negative");
    }
    else{
      alert("x is positive");
    }
    ```

  -
    ```
    switch(favoriteProf){
      case "Yoon":
        statements;
        break;
      case "Lank":
        statements;
        break;
      default:
        statement;
    }
    ```

Statements

- Loops
  - for(var i =0; i < myArray.length;i++){
      document.write(i);
    }

  - do
    { statements;}
    while (condition)

  - while(condition){
      statements;
    }

Functions

- The function is a set of statements that perform some specific piece of work.
- The function describes its name, any values (known as "arguments") which it accepts incoming, and the statements of which the function is comprised.

```
function funcName(argument1,argument2,etc) {
  statements;
}
```

- Example:

```
function foo(myString){   // this is function definition
      document.write(myString);
}

foo(“Computers are fun”);         // calling the function
```

Function Example

```
<html>
<head>
<script type="text/javascript">
    function displaymessage()
    {
           alert(“Salaam ho gya g!");
    }
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()">
</form>
</body>
</html>
```

Events

- Events are actions that can be detected by JavaScript.
- Every element on a web page has certain events which can trigger JavaScript functions.
- For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.
- Examples of events:
  - A mouse click
  - A web page loading or an image loading
  - Mousing over a hot spot on the web page
  - Selecting an input box in an HTML form
  - Submitting an HTML form
  - A keystroke

Event Handlers

- JavaScript events are placed inside HTML Tags
- Examples:

onclick	onmousedown
onload	ondblclick
onunload	onkeypress
onmouseout	onkeydown
onmouseover	onkeyup
onmouseup	onselect
onchange	onsubmit
onfocus	onblur

Real Life Examples

- Form Validation

  Create function to run on form's onSubmit event

```
<form id="myForm" action="#" onsubmit="return validate()“
    method="get" >

    <input type="text" name="firstName" id=“fName“ />
    <input type=“text" name=“lastName" id=“lName“ />
    <input type="submit" />

</form>
```

Real Life Examples

- Form Validation

```
function validate()
{
    if(myform[“fname"].value==""){
      alert("You must enter your first name");
      myform[“fname"].focus();
      return false;
    }
    if(myform[“lname"].value==""){
      alert("You must enter your last name");
      myform[“lname"].focus();
      return false;
    }
    return true;
}
```

Real Life Examples

- Other ways to access a form
  - document.forms[0].elements[0]

  gets the first form and first element of the form
  - document.forms[“formName”].elements[“elementName”]
  - document.formName.elementName

Image Rollover

- Swap Images
  - On mouse over event swap the image
  - On mouse out event restore the image
- Code

```
<html>
<head>
</head>
<body>
<a href="#" onmouseover="myimage.src='image2.jpg'"
    onmouseout="myimage.src='image1.jpg'">
<img src="image1.jpg" id="myimage" name="myimage" width="130"
    height="15" /></a>
</body>
</html>
```

Assignment 3

- You need to develop web site that will contain a home page (e.g. index.html) and at least 3 (but no more than 6) content pages. You are required to create an **external style sheet** (.css file) that configures text, font, color, border, margins and page layout. **(No font tags, embedded CSS, or inline CSS may be used.)**
- The content pages will include at least:
  - One page containing a form with at least three elements. Use proper data validation using JavaScript.
  - Consistent banner logo area
  - Consistent main navigation
  - Association with external style sheet (.css file)
  - You can use any tool (e.g. Fireworks or Photoshop) to design your home page.

Displaying an alert box

- An alert box is often used if you want to make sure information comes through the user.
- When an alert box pops up, the user will have to click "OK" to proceed.
- **Syntax:**
- alert("sometext");

- The following code displays an alert box when a button is clicked:

-
  ```
<form>
    <input type="button" name="Submit" value="Alert!“
              onclick="alert('Ohh, something happened!');">
</form>
```

Confirm Box

- **Confirm Box**
- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.
- **Syntax:**
- confirm("sometext");

Prompt Box

- **Prompt Box**
- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
- **Syntax:**
- prompt("sometext”,”defaultvalue”);

JavaScript Objects

- JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.
- We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail.
- **Note that an object is just a special kind of data. An object has properties and methods.**

JavaScript Objects

- To declare an object
  var myObj = new Object();
- To set properties
  myObj.name = “blah”

JavaScript Objects

- **Properties**
  - Properties are the values associated with an object.
  - In the following example we are using the length property of the String object to return the number of characters in a string:

```
<Html>
<Head> <Title> My Page</Title>
<script type="text/javascript">
var txt="Hello BIT-9!"
document.write(txt.length)
</script>
</Head>
<Body> Hello BIT-9 </Body>
</Html>
```

The output of the code above will be: 12

JavaScript Objects

- **Methods**
  - Methods are the actions that can be performed on objects.
  - In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<Html>
<Head> <Title> My Page</Title>
<script type="text/javascript">
var str="Hello world!"
document.write(str.toUpperCase())
</script>
</Head>
<Body> Hello BIT-9 </Body>
</Html>
```

The output of the code above will be: HELLO WORLD!

String Objects

- The String object is used to manipulate a stored piece of text.

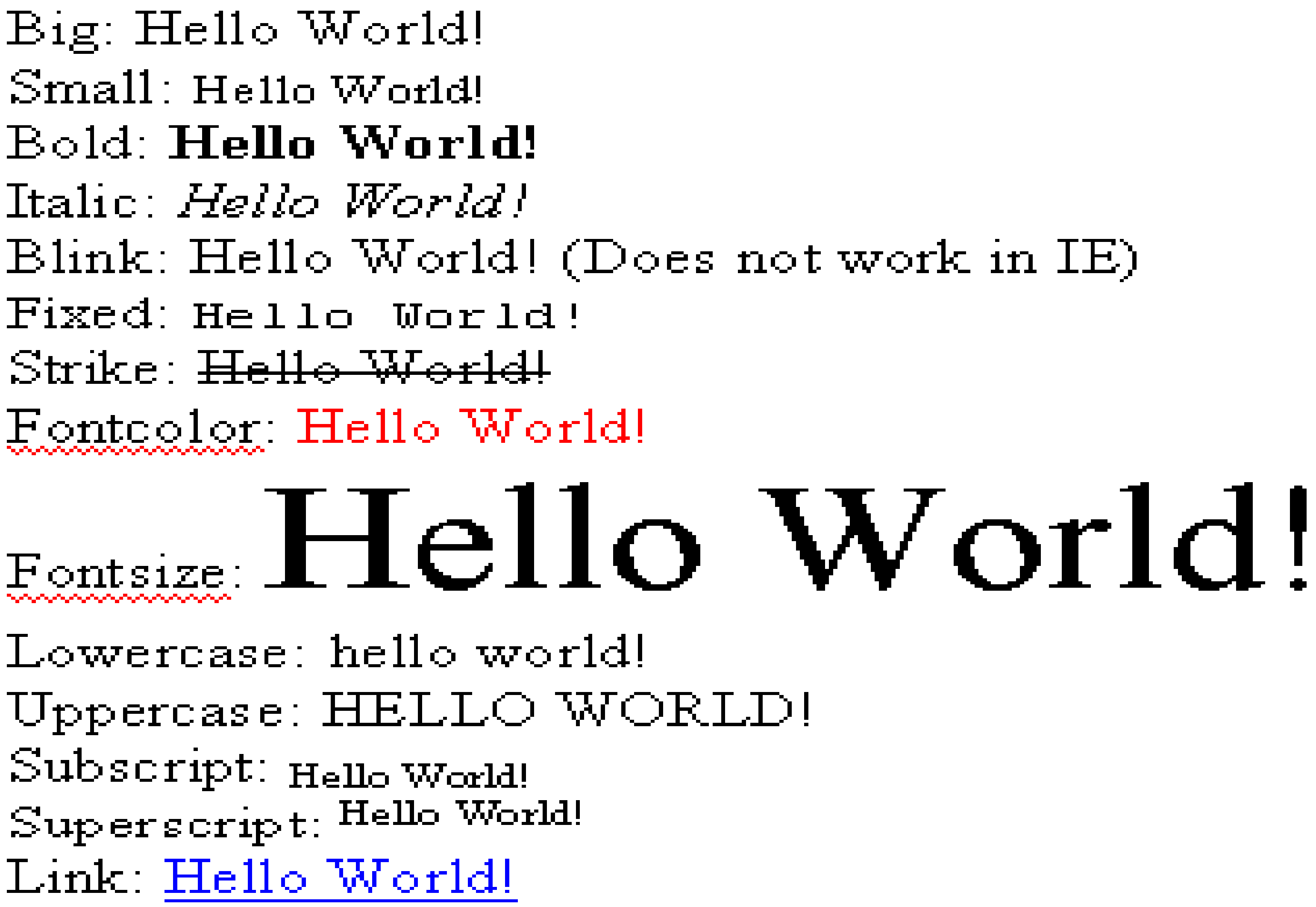
- **Examples**
  - **Length**

    How to use the Length property of a string
  - **Styling**
  - **The indexOf() method**
    How to use the indexOf() method to return the position of the first occurrence of a specified string value in a string
  - **The match() method**
    How to use the match() method to search for a specified string value within a string and return the string value if found
  - **Replace characters in a string - replace()**
    How to use the replace() method to replace some characters with some other characters in a string.

Styling Strings

```
<html>
<body>
<script type="text/javascript">
var txt="Hello World!"
document.write("<p>Big: " + txt.big() + "</p>")
document.write("<p>Small: " + txt.small() + "</p>")
document.write("<p>Bold: " + txt.bold() + "</p>")
document.write("<p>Italic: " + txt.italics() + "</p>")
document.write("<p>Blink: " + txt.blink() + " (does not work in IE)</p>")
document.write("<p>Fixed: " + txt.fixed() + "</p>")
document.write("<p>Strike: " + txt.strike() + "</p>")
document.write("<p>Fontcolor: " + txt.fontcolor("Red") + "</p>")
document.write("<p>Fontsize: " + txt.fontsize(16) + "</p>")
document.write("<p>Lowercase: " + txt.toLowerCase() + "</p>")
document.write("<p>Uppercase: " + txt.toUpperCase() + "</p>")
document.write("<p>Subscript: " + txt.sub() + "</p>")
document.write("<p>Superscript: " + txt.sup() + "</p>")
document.write("<p>Link: " + txt.link("http://www.niit.edu.pk") + "</p>")
</script>
</body>
</html>
```

Output



Big: Hello World!

Small: Hello World!

Bold: **Hello World!**

Italic: *Hello World!*

Blink: Hello World! (Does not work in IE)

Fixed: Hello World!

Strike: ~~Hello World!~~

Fontcolor: Hello World!

Fontsize: Hello World!

Lowercase: hello world!

Uppercase: HELLO WORLD!

Subscript: Hello World!

Superscript: Hello World!

Link: Hello World!

IndexOf

```
<html>
<body>
<script type="text/javascript">
var str="Hello world!"
document.write(str.indexOf("Hello") + "<br />")
document.write(str.indexOf("World") + "<br />")
document.write(str.indexOf("world"))
</script>
</body>
</html>
```

Output:

```
0
-1
6
```

Matching

```
<html>
<body>
<script type="text/javascript">
var str="Hello world!"
document.write(str.match("world") + "<br />")
document.write(str.match("World") + "<br />")
document.write(str.match("worlld") + "<br />")
document.write(str.match("world!"))
</script>
</body>
</html>
```

Output:

```
world
null
null
world!
```

Replace

```
<html>
<body>
<script type="text/javascript">
var str="Visit NIIT!"
document.write(str.replace("NIIT","NUST"))
</script>
</body>
</html>
```

Output: Visit NUST

Date Object

- **The Date object is used to work with dates and times.**
- **Examples**
  - **Return today's date and time**
    How to use the Date() method to get today's date.
  - **get Time()**
    Use getTime() to calculate the years since 1970.
  - **setFullYear()**
    How to use setFullYear() to set a specific date.
  - **getDay()**
    Use getDay() and an array to write a weekday, and not just a number.
  - **Display a clock**
    How to display a clock on your web page.

For a complete list of Date Object methods:

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

Getting the date

```
<Html>
<Head> <Title> My Page</Title>
<script type="text/javascript">
    var d = new Date()
    document.write(d.getDate() + "/")
    document.write((d.getMonth() + 1) + "/")
    document.write(d.getFullYear())
</script>
</Head>
</Html>
```

OUTPUT: **26/5/2009**

Math Objects

- **The Math object allows you to perform common mathematical tasks.**
- **Examples**
  - **round()**
    How to use round().
  - **random()**
    How to use random() to return a random number between 0 and 1.
  - **max()**
    How to use max() to return the number with the highest value of two specified numbers.
  - **min()**
    How to use min() to return the number with the lowest value of two specified numbers.

Getting a random number

- The following code gets a random floating-point number between 0 and 1:

-
  ```
  <script type="text/javascript">
      document.write(Math.random())
  </script>
  ```

```
0.728762788388911
```

Getting a random integer

- The following code gets a random integer between 1 and 10:

```
<script type="text/javascript">
      var max = 10;
      number=Math.random()*max + 1;
      document.write(Math.floor(number));
  </script>
```

```
5
```

Text Literals

- Double and Single Quotes
- Escape Sequences begin with a \
  - \b is backspace, \n is newline, \r is CR
  - \' is single quote (so you can use it in a string as a literal)
  - \\ is backslash

Math

- Standard Math functions supported (+, -, *, /, etc.)
- Use parseInt() or parseFloat() method to treat variables as numbers
- Precedence is important!

```
var x = prompt ("What Year Is It?");
var y = prompt ("What Year Were You Born?”);
document.write ("You're " +  (parseInt(x)-
parseInt(y)) + " years old");
```

Array literals

- You don’t declare the *types* of variables in JavaScript
- JavaScript has array *literals,* written with brackets and commas
  - Example: color = ["red", "yellow", "green", "blue"];
  - Arrays are *zero-based:* color[0] is "red"
- If you put two commas in a row, the array has an “empty” element in that location
  - Example: color = ["red", , , "green", "blue"];
    - color has 5 elements
  - However, a single comma at the end is ignored
    - Example: color = ["red", , , "green", "blue”,]; still has 5 elements

Four ways to create an array

- You can use an array literal:
  var colors = ["red", "green", "blue"];
- You can use new Array() to create an empty array:
  - var colors = new Array();
  - You can add elements to the array later:
    colors[0] = "red"; colors[2] = "blue"; colors[1]="green";
- You can use new Array(*n*) with a single numeric argument to create an array of that size
  - var colors = new Array(3);
  - You can add elements to the array later:
    colors[0] = "red"; colors[2] = "blue"; colors[1]="green";
- You can use new Array(*…)* with two or more arguments to create an array containing those values:
  - var colors = new Array("red","green", "blue");

The length of an array

- If myArray is an array, its length is given by `myArray.length`
- Array length can be changed by assignment beyond the current length
  - Example: `var myArray = new Array(5); myArray[10] = 3;`
- Arrays are sparse, that is, space is only allocated for elements that have been assigned a value
  - Example: `myArray[50000] = 3;` is perfectly OK
  - But indices must be between 0 and 2^{32}-1

Array functions

- If myArray is an array,
  - myArray.sort() sorts the array alphabetically
  - myArray.sort(function(a, b) { return a - b; }) sorts numerically
  - myArray.reverse() reverses the array elements
  - myArray.push(…) adds any number of new elements to the end of the array, and increases the array's length
  - myArray.pop() removes and returns the last element of the array, and decrements the array's length
  - myArray.toString() returns a string containing the values of the array elements, separated by commas

Thank You

- Thank you