Web Application Development

JavaScript

Why use client-side scripting?

- Server side languages already allows us to create dynamic web pages. Why also use client-side scripting?
  - Client-side scripting (Javascript) benefits:
    - **Usability:** can modify a page without having to post back to the server (faster UI)
    - **Efficiency:** can make small, quick changes to page without waiting for server
    - **Event-driven:** can respond to user actions like clicks and key presses

Why use server-side programming?

- Server-side programming (ASP.NET, PHP, etc.) benefits:
  - **Security:** has access to server's private data; client can't see source code
  - **Compatibility:** not subject to browser compatibility issues
  - **Power:** can write files, open connections to servers, connect to databases, ...

What is Javascript?

- A lightweight scripting language which follows all the programming language fundamentals e.g. data types, control statements, loops, overloading, etc.
  - Used to make web pages interactive
  - Insert dynamic text into HTML
  - React to events (ex: page load, user click)
  - Get information about a user's computer (ex: browser type, version, etc.)
  - Perform calculations on user's computer (ex: form validation)
  - Store data using Cookies

What Javascript can do?

- Run a VM inside a browser
- Run a game inside the browser
- Serve 300 million users without overloading CPU
- Help in making word documents, spreadsheets and PPTs online
- Make real time chat possible

Atwood's Law: “Any application that can be written in JavaScript, will eventually be written in JavaScript.”

How To Use JavaScript

- JavaScript is Usually Embedded Directly into Web Pages
  - Contained within a web page and integrates with its HTML/CSS content

The HTML <script> tag is used to insert a JavaScript into an HTML page.

```
<!DOCTYPE html>
<html lang="en-US">
<head>
   <title>Using JavaScript With Script Tag</title>
   <meta charset="UTF-8">
</head>
<body>
   <h1>Using JavaScript With Script Tag</h1>
   <p id="demo">Will Not Print This</p>

<script>
<!--
   document.getElementById("demo").innerHTML="Hello World!";
//-->
</script>

</body>
</html>
```

JavaScript in <head>

```
<!DOCTYPE html>
<html lang="en-US">
<head>
   <title>Using JavaScript in HTML Head</title>
   <meta charset="UTF-8">
   <script type="text/javascript">
   function changeText()
   {
        document.getElementById("demo").innerHTML="Hello World!";
   }
   </script>
</head>
<body>
   <h1>Using JavaScript in HTML Head</h1>

   <p id="demo">This Will Be Replaced By JavaScript</p>
   <button onclick="changeText()">Change Text</button>

</body>
</html>
```

See: Example 02

Using an External JavaScript

- JavaScript can also be placed in **external files**.
- External JavaScript files have the file extension **.js**.

```
<script type="text/javascript" src="ext.js"></script>
```

```
function changeText()
{
    document.getElementById("demo").innerHTML="Hello World!";
}
```

Reference: http://stackoverflow.com/questions/436411/where-is-the-best-place-to-put-script-tags-in-html-markup

Using an External JavaScript

```
<!DOCTYPE html>
<html lang="en-US">
<head>
   <title>Using External JavaScript</title>
   <meta charset="UTF-8">

   <script type="text/javascript" src="ext.js"></script>

</head>
<body>

   <h1>Using External JavaScript</h1>
   <p id="demo">This Will Be Replaced By JavaScript</p>
   <button onclick="changeText()">Change Text</button>

</body>
</html>
```

See: Example 03

SYNTAX

JavaScript Statements

- JavaScript is **Case Sensitive**
  - Therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.
- A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do.
  - It is normal to **add a semicolon** at the end of each executable statement. Using semicolons makes it possible to write multiple statements on one line.

JavaScript Comments

- Comments can be added to explain the JavaScript, or to make the code more readable.
  - **Single line comments** start with **//**
  - **Multi line comments** start with /* and end with */

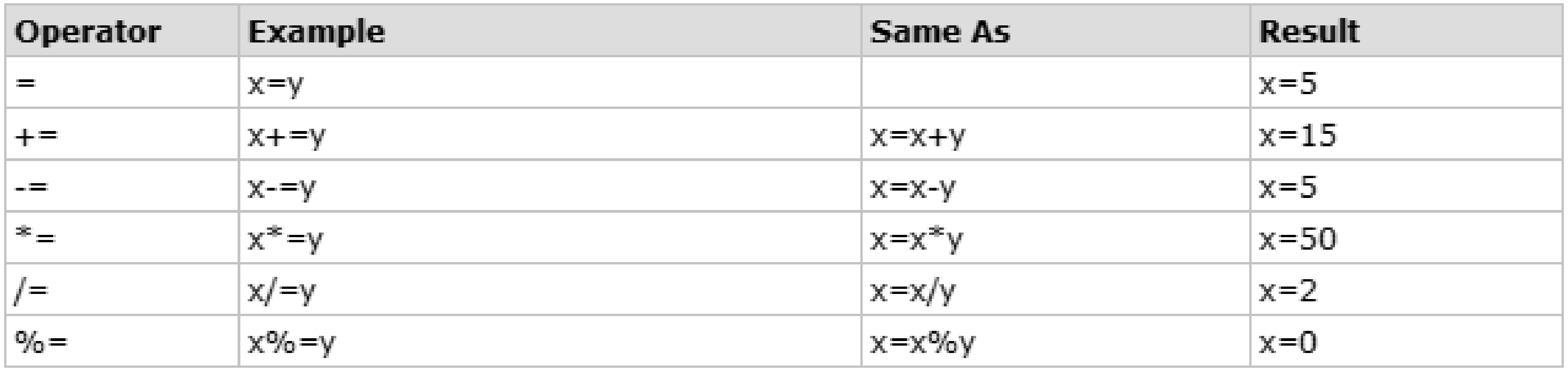
JavaScript Variables

- The general rules for constructing names for variables (unique identifiers) are:
  - Names can contain **letters**, **digits**, **underscores**, and **dollar signs**.
  - Names must begin with a **letter**
  - Names can also begin with **$** and **_**
  - Names are **case sensitive** (y and Y are different variables)
- You declare a JavaScript variable with the **var** keyword:
  - var carName;
    carName = "Volvo";
  - var carName = "Volvo";
  - var carName = "Volvo", carType="Sedan";

See: http://www.w3schools.com/js/js_variables.asp

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

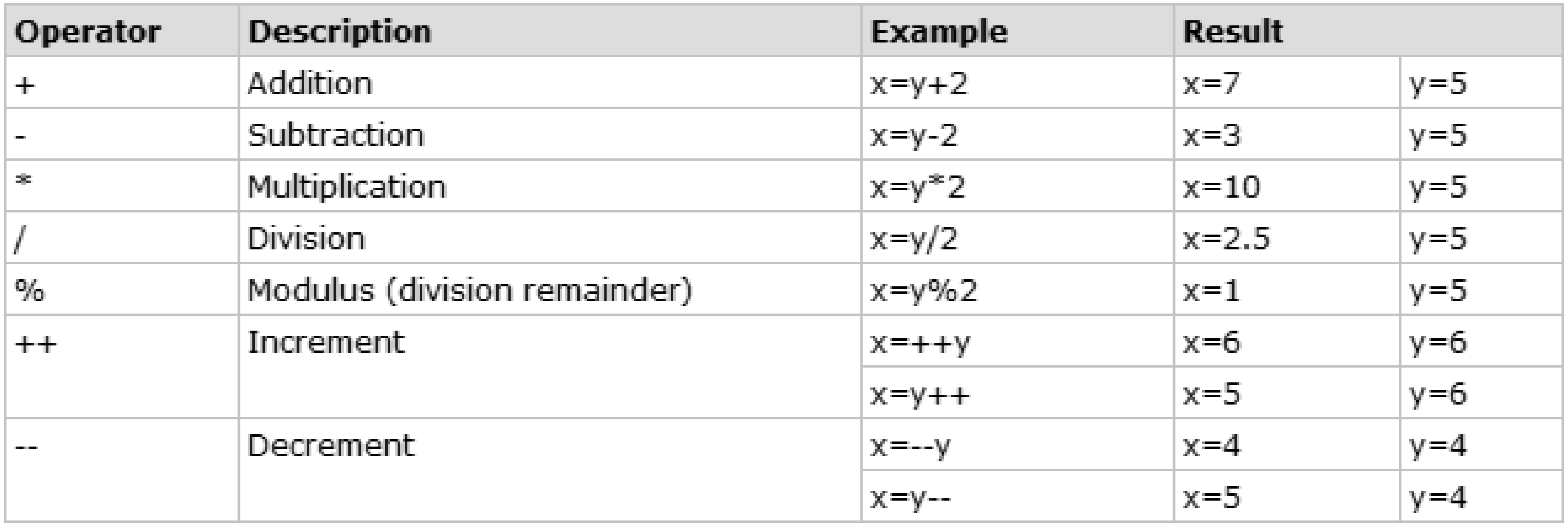


Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

See: http://www.w3schools.com/js/js_operators.asp

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.



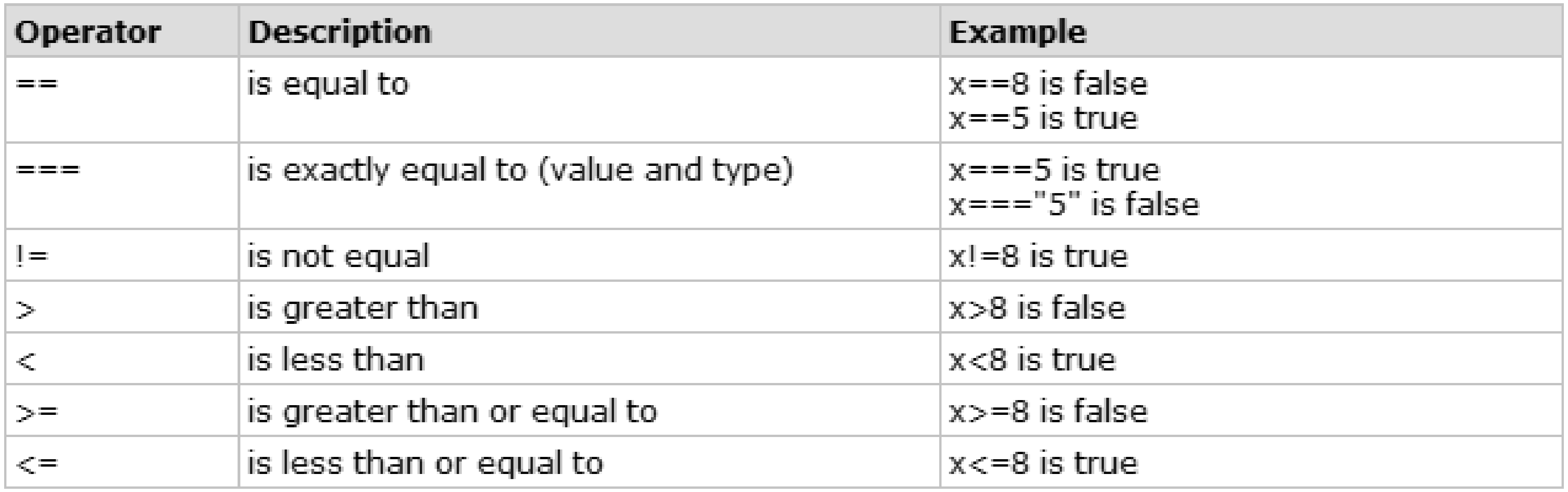
Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

See: http://www.w3schools.com/js/js_operators.asp

JavaScript Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that x=5, the table below explains the comparison operators:



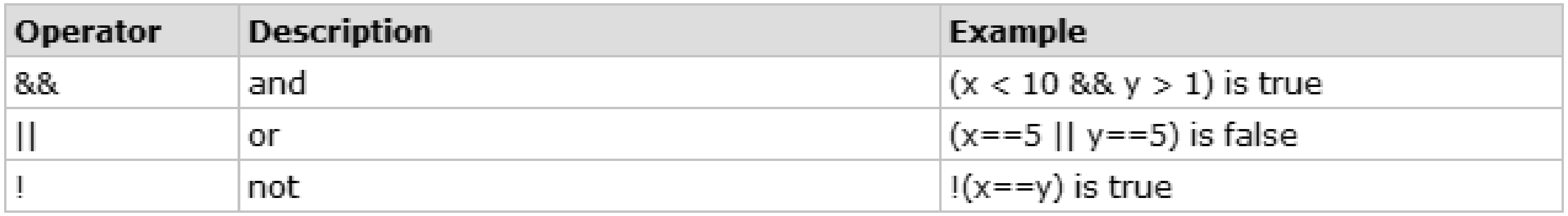
Operator	Description	Example
==	is equal to	x==8 is false x==5 is true
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

See: http://www.w3schools.com/js/js_comparisons.asp

JavaScript Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that x=6 and y=3, the table below explains the logical operators:



Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

See: http://www.w3schools.com/js/js_comparisons.asp

If...Else Statements

```
var d = new Date();
var time = d.getHours();
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>=10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
else
  {
  document.write("<b>Hello World!</b>");
  }
```

See: http://www.w3schools.com/js/js_if_else.asp

Switch Statement

```
var d=new Date();
var theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
```

See: http://www.w3schools.com/js/js_switch.asp

For Loop

```
var i=0;
for (i=0;i<=5;i++)
{
   document.write("The number is " + i);
   document.write("<br />");
}
```

While Loop

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

See: http://www.w3schools.com/js/js_loop_while.asp

Do While Loop

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

See: http://www.w3schools.com/js/js_loop_while.asp

JavaScript Functions

- A function contains code **that will be executed by an event or by a call to the function**.
  - You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).
  - Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.
- **Note:** A function with no parameters **must include the parentheses ()** after the function name.
- **Note:** The word **function** must be written in lowercase letters, otherwise a JavaScript error occurs! Also note **function name** is case-sensitive.

See: http://www.w3schools.com/js/js_functions.asp

JavaScript Functions

```
function product(a,b)
{
   return a*b;
}

C = product(4,3);
```

JavaScript Variable Scope

- Variables declared within a JavaScript function, become **local** to the function.
- A global variable has **global scope**: All scripts and functions on a web page can access it.
- If you assign a value to a variable that has not been declared, it will automatically become a **global** variable.

```
var x="For Everyone";                    // global
function dosomething()
{
  var y="For Function Only";             // local
  z="Also For Everyone";                 // global
}
```

JavaScript Variable Scope

```
<script>
    var x="Global X";
    var y="Global Y";
    function dosomething()
    {
        x="Global X Modified";
        var y="Local Y";
        z="Global Z";

        document.write("Value of x = "+x+ "<br />");
        document.write("Value of y = "+y+ "<br />");
        document.write("Value of z = "+z+ "<br />");

        document.write("-----------------------------------
   <br />");

        y="Modified Y";
        document.write("Value of Local y = "+y+ "<br />");
        document.write("Value of Global y = "+this.y+ "<br />");
       // Can also use window.y
        document.write("-----------------------------------
   <br />");
    }

</script>
```

JavaScript Variable Scope

```
<script>

    document.write("Value of x = "+x+ "<br />");
    document.write("Value of y = "+y+ "<br />");
    document.write("--------------------------------------<br />");
    dosomething();
    document.write("Value of x = "+x+ "<br />");
    document.write("Value of y = "+y+ "<br />");
    document.write("Value of z = "+z+ "<br />");
    document.write("--------------------------------------<br />");

</script>
```

Anonymous Functions

```
var x = function (a,b) {
   return a*b;
};
```

Anonymous Functions

```
var x = function (a,b) {
   return a*b;
};

var y = x(4,5);
```

Functions in Expressions

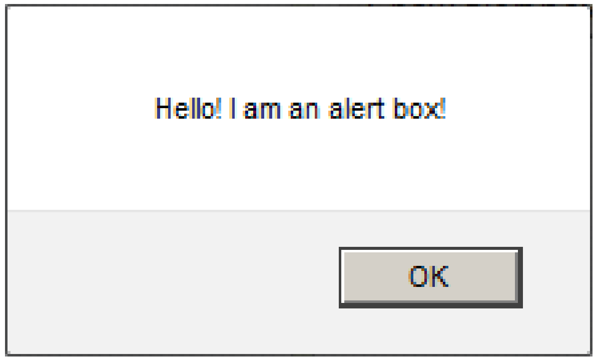
```
function myFunction(a, b) {
    return a * b;
}

var x = myFunction(4, 3) * 2;
```

Self-Invoking Functions

```
(function () {
    var x = "Hello!!";
})();
```

JavaScript Popup Boxes

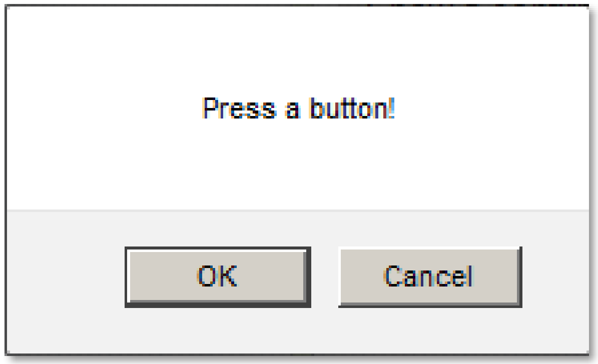
- JavaScript has three kind of popup boxes:
  - Alert Box
  - Confirm Box
  - Prompt Box


Hello! I am an alert box!

See: http://www.w3schools.com/js/js_popup.asp

JavaScript Popup Boxes

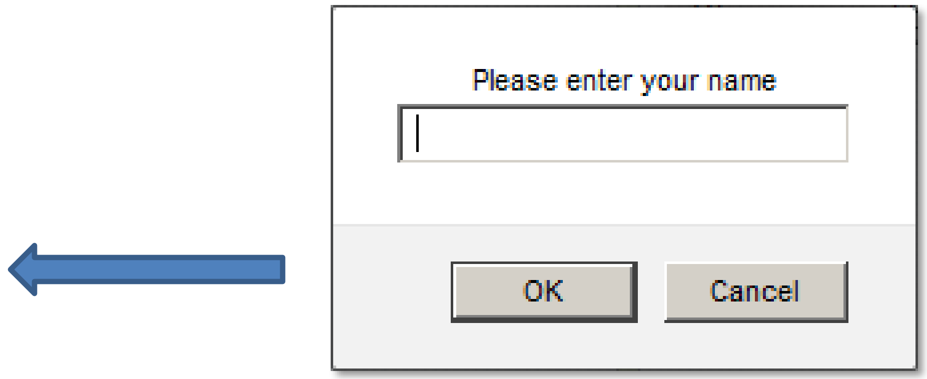
- JavaScript has three kind of popup boxes:
  - Alert Box
  - Confirm Box
  - Prompt Box


Press a button!

See: http://www.w3schools.com/js/js_popup.asp

JavaScript Popup Boxes

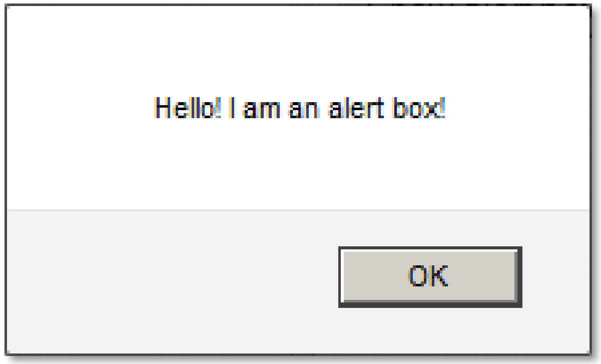
- JavaScript has three kind of popup boxes:
  - Alert Box
  - Confirm Box
  - Prompt Box



Alert Box

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.
Alert Box

```
alert("I am an alert box!");
```

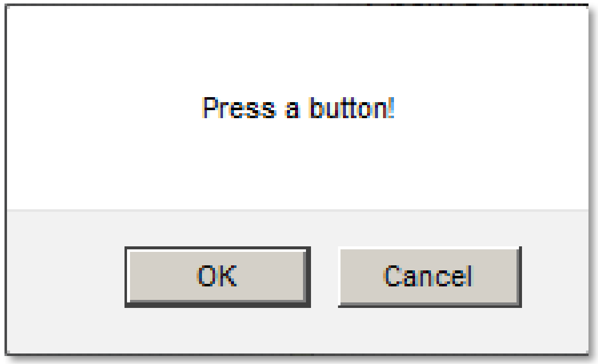

Hello! I am an alert box!

See: http://www.w3schools.com/js/js_popup.asp

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**.

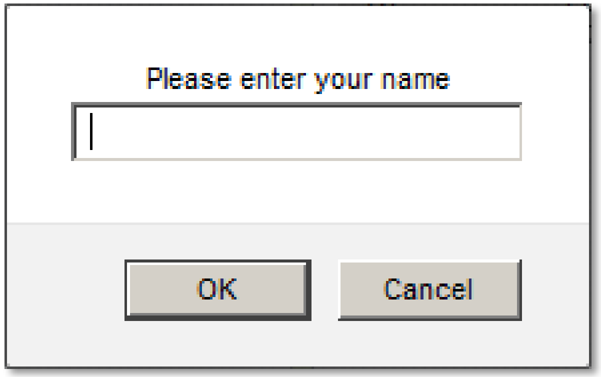
```
var r=confirm("Press a button");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
```



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**.

```
var name=prompt("Please enter your name","");
if (name!=null && name!="")
  {
  document.write("Hello " + name);
  }
```



JavaScript Events

- HTML elements have **special attributes called event attributes**
- JavaScript functions can be set as **event handlers**
- When you interact with the element, the function will execute
- **onclick** is just one of many HTML event attributes
- For a comprehensive list of **Event Attributes**:
    - http://www.w3schools.com/jsref/dom_obj_event.asp

See: http://www.w3schools.com/js/js_events.asp

Event Attributes

- onblur An element loses focus
- onchange The content of a field changes
- onclick Mouse clicks an object
- ondblclick Mouse double-clicks an object
- onfocus An element gets focus
- onload A page or image is finished loading
- onmousedown A mouse button is pressed
- onmouseup A mouse button is released
- onselect Text is selected
- onunload The user exits the page

See: http://www.w3schools.com/jsref/dom_obj_event.asp

Mouse Events

Some of the mouse events:

- **onmousedown** : user presses down mouse button on this element
- **onmouseup** : user releases mouse button on this element
- **onclick** : user presses/releases mouse button on this element
- **ondblclick** : user presses/releases mouse button twice on this element
- **onmouseover** : mouse cursor enters this element's box
- **onmouseout** : mouse cursor exits this element's box
- **onmousemove** : mouse cursor moves within this element's box

See: http://www.w3schools.com/jsref/dom_obj_event.asp

JAVASCRIPT BUILT IN OBJECTS
STRING OBJECT

JavaScript String Object

Methods:

- charAt() Returns the character at the specified index
- substr() Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
- toLowerCase() Converts a string to lowercase letters
- toUpperCase() Converts a string to uppercase letters

Property:

– length

See: http://www.w3schools.com/jsref/jsref_obj_string.asp

JavaScript String Object

```
var txt = "Hello World!";

document.write("Length of String:" + txt.length + "<br />");

document.write(txt.toLowerCase() + "<br />");
document.write(txt.toUpperCase());
```

See: Example 05

JavaScript String Object

Escape sequences behave as in Java

```
\' \" \& \n \t \\
```

For Example:

```
str="We can have \"double quotes\" in strings";

document.write(str + "<br />");
```

See: Example 07

JAVASCRIPT BUILT IN OBJECTS ARRAY OBJECT

JavaScript Array Object

Three ways to initialize an array in JavaScript:

```
var stooges = new Array();
stooges[0] = "Larry";
stooges[1] = "Moe";
stooges[2] = "Curly";

var stooges = new Array("Larry", "Moe", "Curly");
```

```
var stooges = ["Larry", "Moe", "Curly"];
```

JavaScript Array Object

Methods:

- concat() Joins two or more arrays, and returns a copy of the joined arrays
- join() Joins all elements of an array into a string
- reverse() Reverses the order of the elements in an array
- slice() Selects a part of an array, and returns the new array
- sort() Sorts the elements of an array
- toString() Converts an array to a string, and returns the result

JAVASCRIPT BUILT IN OBJECTS DATE OBJECT

JavaScript Date Object

Four ways of instantiating a date:

```
var today = new Date();

// milliseconds since 1970/01/01
var d1 = new Date(1000000);

// date string
var d2 = new Date("October 17, 1979 11:13:00");

// year, month, day, hours, minutes, seconds, milliseconds
var d3 = new Date(79,9,17,11,33,0);
```

JavaScript Date Object

Methods:

- getDate() Returns the day of the month (from 1-31)
- getDay() Returns the day of the week (from 0-6)
- getFullYear() Returns the year (four digits)
- getHours() Returns the hour (from 0-23)
- getMilliseconds() Returns the milliseconds (from 0-999)
- getMinutes() Returns the minutes (from 0-59)
- getMonth() Returns the month (from 0-11)
- getSeconds() Returns the seconds (from 0-59)
- getTime() Returns the number of milliseconds since midnight Jan 1, 1970

JavaScript Date Object

```
var currentDate = new Date();

var month = currentDate.getMonth() + 1;
var day = currentDate.getDate();
var year = currentDate.getFullYear();

document.write(currentDate + "<br />");
document.write(month + "/" + day + "/" + year);
```

See: Example 04 Example 08 (Difference)

JAVASCRIPT BUILT IN OBJECTS
MATH OBJECT

JavaScript Math Object

- The Math object allows you to perform mathematical tasks.

```
var x=Math.PI;
var y=Math.sqrt(16);
```

- **Note:** Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

JavaScript Math Object

Methods:

- abs(x) Returns the absolute value of x
- ceil(x) Returns x, rounded upwards to the nearest
- floor(x) Returns x, rounded downwards to the nearest integer
- max(x,y,z,...,n) Returns the number with the highest value
- min(x,y,z,...,n) Returns the number with the lowest value
- random() Returns a random number between 0 and 1
- round(x) Rounds x to the nearest integer

See: http://www.w3schools.com/jsref/jsref_obj_math.asp

JavaScript Math Object

```
x=Math.random();

// random number between 0 and 1
document.write(x + "<br />");

x=Math.floor(x*11);

// random integer between 0 and 10
document.write(x + "<br />");
```

See: Example 06

Document Object Model (DOM)

- The DOM defines a standard for accessing and manipulating HTML documents.
  - A representation of the current web page as a **tree of JavaScript objects**
  - Allows you to **view/modify page elements** in script code
- The DOM defines the **objects** and **properties** of all document elements, and the **methods** (interface) to access them.

See: http://www.w3schools.com/htmldom/

HTML Nodes

According to the DOM, everything in an HTML document is a node.

The DOM says:

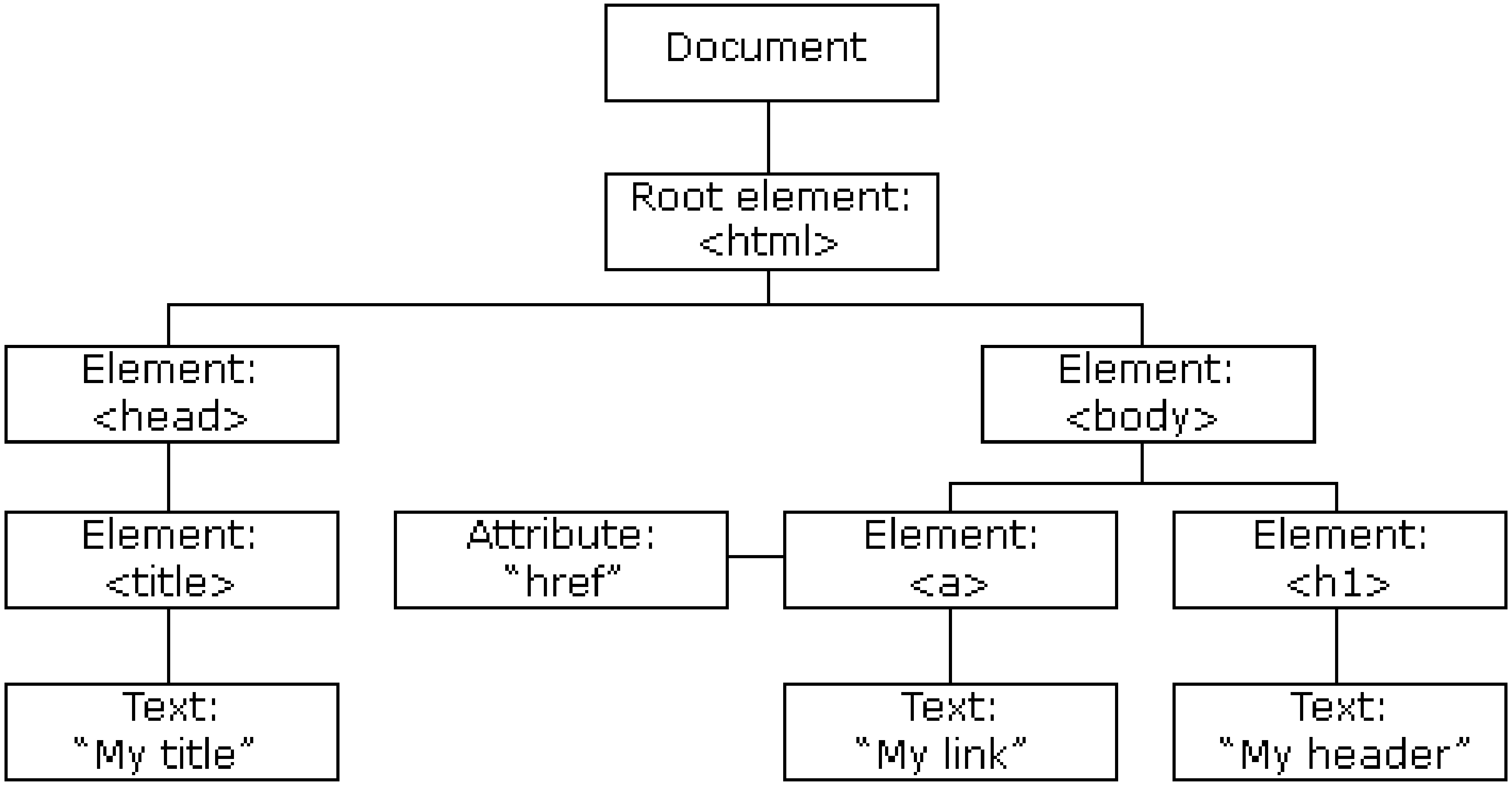
- The entire document is a **document node**
- Every HTML element is an **element node**
- The text in the HTML elements are **text nodes**
- Every HTML attribute is an **attribute node**
- Comments are **comment nodes**

DOM Example

Consider Following Code:

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>My header</h1>
    <a href="http://fc.riphah.edu.pk">My link</a>
  </body>
</html>
```

DOM Tree Example



Node Parents, Children & Siblings

- The nodes in the node tree have a hierarchical relationship to each other
  - In a node tree, the top node is called the **root**
  - Every node, except the root, has exactly one **parent node**
  - A node can have any number of **children**
  - A **leaf** is a node with no children
  - **Siblings** are nodes with the same parent

HTML DOM Properties

Some DOM properties:

- x.**innerHTML** - the text value of x
- x.**nodeName** - the name of x
- x.**nodeValue** - the value of x
- x.**parentNode** - the parent node of x
- x.**childNodes** - the child nodes of x
- x.**attributes** - the attributes nodes of x

Note: In the list above, x is a node object (HTML element).

HTML DOM Properties

The nodeName Property (x.**nodeName)**

- nodeName is read-only
- nodeName of an element node is the same as the tag name
- nodeName of an attribute node is the attribute name
- nodeName of a text node is always #text
- nodeName of the document node is always #document

HTML DOM Properties

The nodeValue Property (x.**nodeValue**)

- nodeValue for element nodes is undefined
- nodeValue for text nodes is the text itself
- nodeValue for attribute nodes is the attribute value

HTML DOM Methods

Some DOM methods:

- x.**getElementById(id)** - get the element with a specified id
- x.**getElementsByTagName(name)** - get all elements with a specified tag name
- x.**appendChild(node)** - insert a child node to x
- x.**removeChild(node)** - remove a child node from x

Accessing Nodes

You can access a node in three ways:

- By using the **getElementById()** method
- By using the **getElementsByTagName()** method
- By **navigating the node tree**, using the node relationships

There are two special document properties that allow access to the tags:

- **document.documentElement** - returns the root node of the document
- **document.body** - gives direct access to the <body> tag

The getElementById() Method

```
<p id="intro">Hello World!</p>

<p>This example demonstrates the <b>getElementById</b> method!</p>


<script>

    x=document.getElementById("intro");

    document.write("<p>The text from the intro paragraph: " +
    x.innerHTML + "</p>");

</script>
```

See: Example 01

The getElementsByTagName() Method

```
<p>Hello World!</p>

<p>The DOM is very useful!</p>


<script>

    x=document.getElementsByTagName("p");

    document.write("Text of second paragraph: " + x[1].innerHTML);

</script>
```

See: Example 02

DOM Node List Length

```
<p>Hello World!</p>

<p>The DOM is very useful!</p>

<p>This example demonstrates the <b>length</b> property.</p>


<script>

   x=document.getElementsByTagName("p");

   for (i=0;i<x.length;i++)

   {

     document.write(x[i].innerHTML);

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

   }

</script>
```

See: Example 03

Change the Text of an HTML Element

```
document.getElementById("p1").innerHTML="New text!";
```

Change Style of an HTML Element

```
document.body.style.backgroundColor="#cccccc";

document.getElementById("p1").style.color="#00cc00";

document.getElementById("p1").style.fontFamily="Arial";
```

See: http://www.w3schools.com/jsref/dom_obj_style.asp

GLOBAL DOM OBJECTS

Global DOM Objects

- Every JavaScript program can refer to the following global objects:
  - **window** : the browser window
  - **navigator** : info about the web browser you're using
  - **screen** : info about the screen area occupied by the browser
  - **history** : list of pages the user has visited
  - **location** : URL of the current HTML page
  - **document** : current HTML page object model

The "window" Object

- Represents the entire browser window
- The top-level object in the DOM hierarchy
- Technically, all global variables become part of the window object
- **Methods:**
  - alert, blur, **clearInterval, clearTimeout**, close, confirm, focus, moveBy, moveTo, open, print, prompt, resizeBy, resizeTo, scrollBy, scrollTo, **setInterval, setTimeout**
- **Properties:**
  - document, history, location, name

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

The “window” Object

```
function delayedMessage() {
    var myTimer = setTimeout("alert('Booyah!');", 5000);
}

<h2 onclick="delayedMessage();">Click me now!</h2>
```

- **setTimeout** executes a piece of code once after a given number of milliseconds
- the function returns an object representing the timer
- to cancel the timer, call **clearTimeout** and pass the timer object
  - clearTimeout(myTimer); // cancel self-destruct sequence!

See: Example 04

The “window” Object

```
<input type="text" id="clock" />

<script type="text/javascript">
var myTimer=setInterval("clock()",1000);
function clock()
{
  var d=new Date();
  var t=d.toLocaleTimeString();
   document.getElementById("clock").value=t;
}
</script>

<button
   onclick="window.clearInterval(myTimer)">Stop</button>
```

See: Example 05

The “navigator” Object

- Information about the web browser application
- **Methods:**
  - javaEnabled()
- **Properties:**
  - appCodeName, appName, appVersion, ,cookieEnabled, **geolocation**, language, onLine, platform, product , userAgent

The “navigator” Object

```
<div id="example"></div>

<script type="text/javascript">
<!--
txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Language: " + navigator.language + "</p>";
txt+= "<p>Online: " + navigator.onLine + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>Product: " + navigator.product + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";

document.getElementById("example").innerHTML=txt;

//-->
</script>
```

See: Example 06

The “navigator.geolocation” Object

```
<button onclick="getLocation()">Try It</button>

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

<script>
var x = document.getElementById("demo");

function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}

function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

See: Example 10 Example 11

The “screen” Object

- Information about the client's display screen
- **Properties:**
  - availHeight, availWidth, colorDepth, height, pixelDepth, width

See: http://www.w3schools.com/jsref/obj_screen.asp

The “screen” Object

```
document.write("Total width/height: ");
document.write(screen.width + "*" + screen.height);
document.write("<br />");
document.write("Available width/height: ");
document.write(screen.availWidth + "*" + screen.availHeight);
document.write("<br />");
document.write("Color depth: ");
document.write(screen.colorDepth);
document.write("<br />");
document.write("Color resolution: ");
document.write(screen.pixelDepth);
```

See: Example 07

The “history” Object

- List of sites the browser has visited in this window
- **Properties:**
  – length
- **Methods:**
  – back, forward, go

See: http://www.w3schools.com/jsref/obj_history.asp

The “history” Object

```
<script type="text/javascript">
function goBack()
  {
  window.history.back()
  }
</script>


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

See: Example 08

The “location” Object

- Represents the URL of the current web page
- **Properties:**
  - host, hostname, href, pathname, port, protocol, search
- **Methods:**
  - assign, reload, replace

See: http://www.w3schools.com/jsref/obj_location.asp

The “location” Object

```
<script type="text/javascript">
function newDoc()
  {
  window.location.assign("http://www.w3schools.com")
  }
</script>

<input type="button" value="Load new document"
   onclick="newDoc()" />
```

See: Example 09

The “document” Object

- Represents current HTML page object model
- **Properties:**
  - anchors, body, cookie, domain, forms, images, links, referrer, title, URL
- **Methods:**
  - getElementById, getElementsByName, getElementsByTagName, write, writeln

See: http://www.w3schools.com/jsref/dom_obj_document.asp

EXAMPLE
FORM VALIDATION

Form Validation

- JavaScript can be used to validate data in HTML forms before sending off the content to a server.
- Form data that typically are checked by a JavaScript could be:
  - Required Field
  - Valid Email Address
  - Numeric Field

Form Validation

```
<script type="text/javascript">
function validateForm()
{
...
}
</script>

<form name="myForm" action="somescript.php"
   onsubmit="return validateForm()" method="post">
...
<input type="submit" value="Submit">
</form>
```

Form Validation

```
function validateForm() {
   var formErrors="";
   var x;
   x=document.forms["myForm"]["fullname"].value;
   if (x==null || x=="") {
         formErrors=formErrors+"Name Required \n";
   }
   if (formErrors!="") {
         alert(formErrors);
         return false;
   }
}
```

See: Example 03

EXAMPLE
CREATE & RETRIEVE “COOKIES”

Create & Retrieve “Cookies”

- A cookie is a variable that is stored on the visitor's computer.
- Each time the same computer requests a page with a browser, it will send the cookie too.
- With JavaScript, you can both create and retrieve cookie values.
- A cookie is nothing but a small text file that's stored in your browser. It contains some data:
  - A name-value pair containing the actual data
  - An expiry date after which it is no longer valid
  - The domain and path of the server it should be sent to

Create & Retrieve “Cookies”

Cookies can be created, read and erased by JavaScript. They are accessible through the property **document.cookie**.

```
document.cookie = "wpone=test; expires=Tue, 31 Dec 2013
   23:59:59 UTC; path=/";
```

1. First the name-value pair (`wpcookie=test`)
2. then a semicolon and a space
3. then the expiry date in the correct format (`expires=Tue, 31 Dec 2013 23:59:59 UTC`)
4. again a semicolon and a space
5. then the path (`path=/`)

Create & Retrieve "Cookies"

```
document.cookie = "wpone=test; expires=Tue, 31 Dec 2014
   23:59:59 UTC; path=/";

document.cookie = "wptwo=another test; expires=Tue, 31
   Dec 2014 23:59:59 UTC; path=/";
```

The second one is added to document.cookie, so if we read it out we get

```
document.write(document.cookie);
wpone=test; wptwo=another test
```

See: Example 01

Create & Retrieve “Cookies”

```
function setCookie(name, value, days) {

    var expDate = new Date();
    var expTime = expDate.getTime() + (days*24*60*60*1000);
    expDate.setTime(expTime);
    var expires = "; expires="+expDate.toUTCString();

   document.cookie = name+"="+value+expires+"; path=/";
}
```

Create & Retrieve "Cookies"

```
function getCookie(name) {
   var cname = name + "=";
   var carray = document.cookie.split(';');
   for(var i=0;i < carray.length;i++) {
       var c = carray[i];
       while (c.charAt(0)==' ')
            c = c.substring(1,c.length);
       if (c.indexOf(cname) == 0)
            return c.substring(cname.length,c.length);
   }
   return null;
}
```

See: Example 02

Minimum Objectives

- ✓ Create **static** websites using HTML & CSS
- ✓ Design & develop **dynamic/interactive** websites & web applications using HTML, CSS, JavaScript.
- ✓ Develop **dynamic database driven** web applications using NODE JS

Q & A