JavaScript

What is JavaScript?

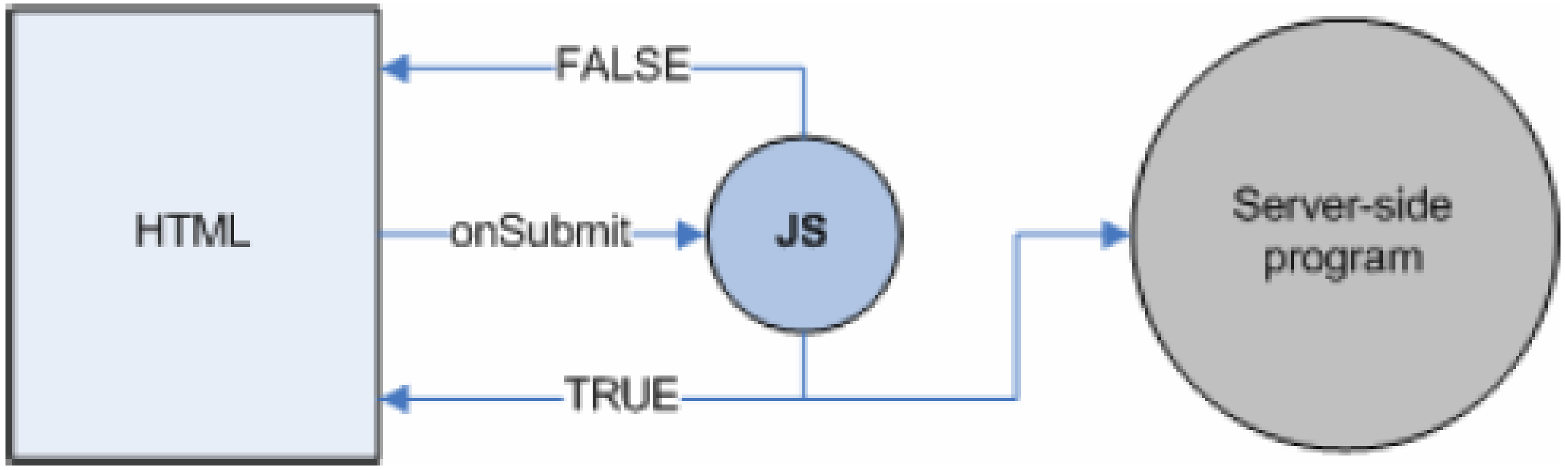
- A lightweight programming language that runs in a Web browser (client-side).
- Embedded in HTML files and can manipulate the HTML itself.
- Interpreted, not compiled.
- JavaScript is <u>not</u> Java.
  - Developed by Netscape, not Sun.
  - Only executed in a browser.
  - Is not a full-featured programming language.
  - However, the syntax is similar.

Why use JavaScript?

- To add dynamic function to your HTML.
  - JavaScript does things that HTML can't—like logic.
  - You can change HTML on the fly.

- To shoulder some of the form-processing burden.
  - JavaScript runs in the browser, not on the Web server.
    - Better performance
  - JavaScript can validate the data that users enter into the form, before it is sent to your Web application.

Form validation



Form validation

1. Add an **onSubmit** event for the form.
2. Use the `return` keyword to get an answer back from JavaScript about whether the data is valid or not.
   - **return false:** server-side program is not called, and the user must fix the field(s).
   - **return true:** the valid data is sent to the server-side program.

JavaScript is not Java

- JavaScript has some features that resemble features in Java:
  - – JavaScript has Objects and primitive data types
  - -- JavaScript has Events and event handlers
  - – Exception handling in JavaScript is almost the same as in Java
- JavaScript has some features unlike anything in Java:
  - – Variable names are untyped: the type of a variable depends on the value it is currently holding
  - – JavaScript has with statements and a new kind of for statement

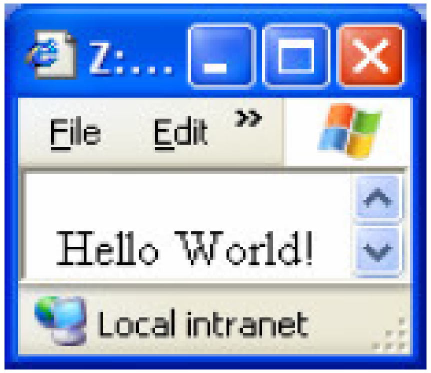
Where to Insert JavaScript in HTML?

- Scripts can be placed in the head section or the body section or both
- Scripts to be executed when they are called or when an event is triggered are placed in the head section
- Scripts to be executed when the page loads are placed in the body section (generate the contents of the page

document.write

- document.write is a standard JavaScript command for writing output to a page.
- Semicolons are optional when single statements are written on separate lines
- Semicolons are required when multiple statements are written on the same line

```
<html>
<body>

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

</body>
</html>
```

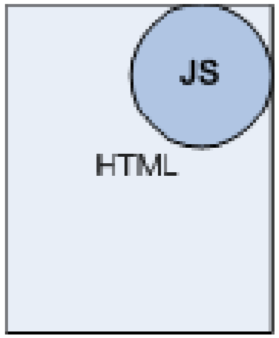


Hello World!

Add JavaScript to HTML

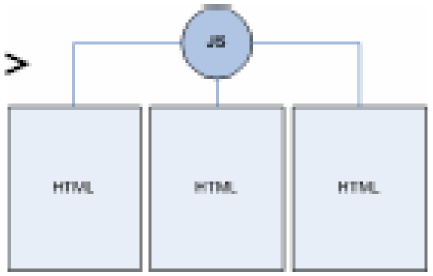
- In the HTML page itself:

```
<html>
<head>
<script language="javascript">
  // JavaScript code
</script>
</head>
```



- As a file, linked from the HTML page:

```
<head>
<script language="javascript" src="script.js">
</script>
</head>
```



External JavaScript

- If you have the same JavaScript written in different HTML pages, you may want to
  - write the script in a separate file,
  - give it the extension .js and
  - point to it in the SRC attribute of the <script> tag

- Note that the .js file cannot contain the <script> tag

```
<html>
<body>

<script src="hello.js">
</script>

</body>
</html>
```

hello.html

```
document.write("Hello World!")
```

hello.js

Comments

- Comments are as in C or Java:
  - Between // and the end of the line
  - Between /* and */
- Java's javadoc comments, /** ... */, are treated just the same as /* ... */ comments; they have no special meaning in JavaScript

JavaScript Variables

- Variables are used to store values
- The values may change during the script
- You may use var to declare a variable

```
var variable = value
variable = value
```

```
Var DMET601 = "JavaScript"

DMET601 = "JavaScript"
```

- Rules for variable names:
  - Variable names are case sensitive
  - They must begin with a letter or the underscore character

- JavaScript has untyped variables.

Variables: Scope and Lifetime

- Variables declared within a function are local and can be accessed only within the function
- Local variables are destroyed once you exit the function

- Variables declared outside a function are accessed from any function within the page
- These variable are destroyed when the page is closed

JavaScript Operators

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- String Operator
- Conditional Operator

Arithmetic Operators

Operator	Description	Example	Result

+	Addition	x=2
		y=2
		x+y
			4

-	Subtraction	x=5
		y=2
		x-y
			3

*	Multiplication	x=5
		y=4
		x*y
			20

/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false

===	is equal to (checks for both value and type)	x=5
		y="5"
		x==y returns true
		x===y returns false

!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example

&&	and	x=6
		y=3
		(x < 10 && y > 1) returns true

||	or	x=6
		y=3
		(x==5 || y==5) returns false

!	not	x=6
		y=3
		!(x==y) returns true

String Operator

- + operator is used for string concatenation
- Example:

```
txt1="What a very"
txt2="nice day!"
txt3=txt1+" "+txt2
```

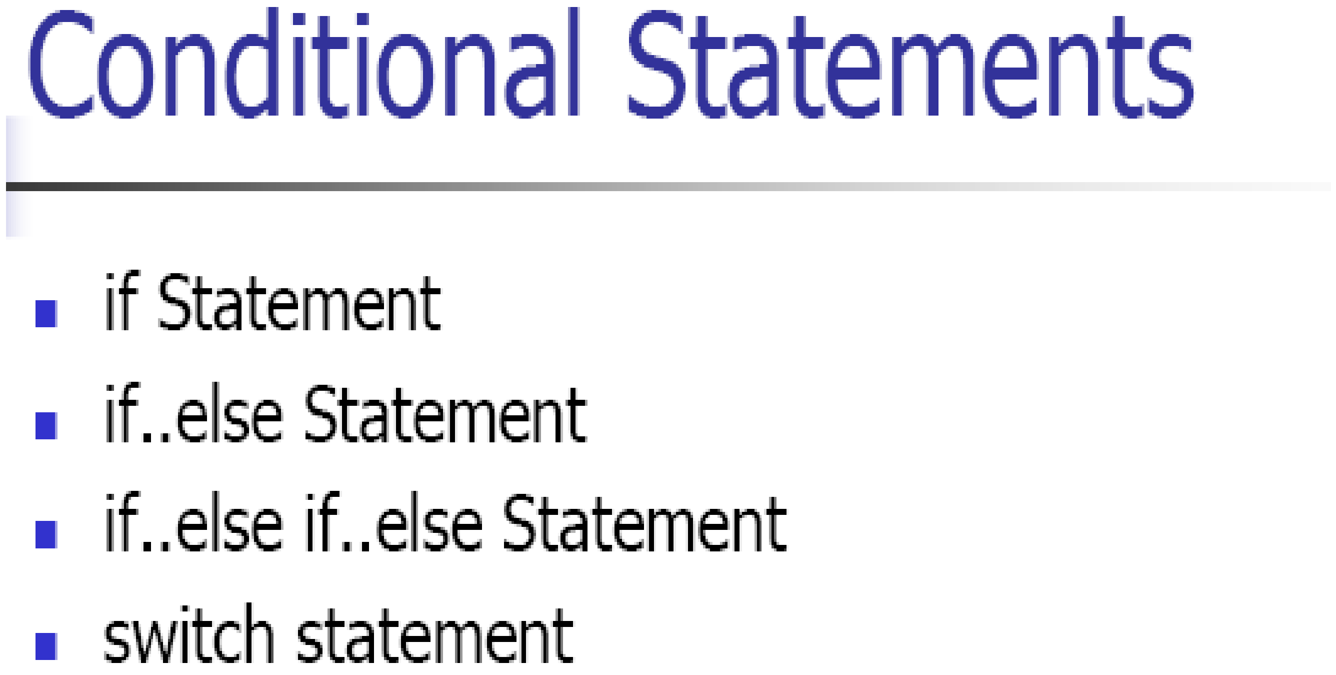
- `txt3` now contains `"What a very nice day!"`

Conditional Operator

- Syntax:

```
variablename=(condition)?value1:value2
```

- This means that:
  - If the condition is true assign `value1` to `variablename`
  - If the condition is false assign `value2` to `variablename`



Conditional Statements

- if Statement
- if..else Statement
- if..else if..else Statement
- switch statement

If Statement

- If statement is used to execute code only if some certain condition is true
- Syntax:

```
if (condition)
{
   code to be executed if condition is true
}
```

- Note that "if" is written in lowercase letters. Using "IF" will generate an error

If Statement: Example

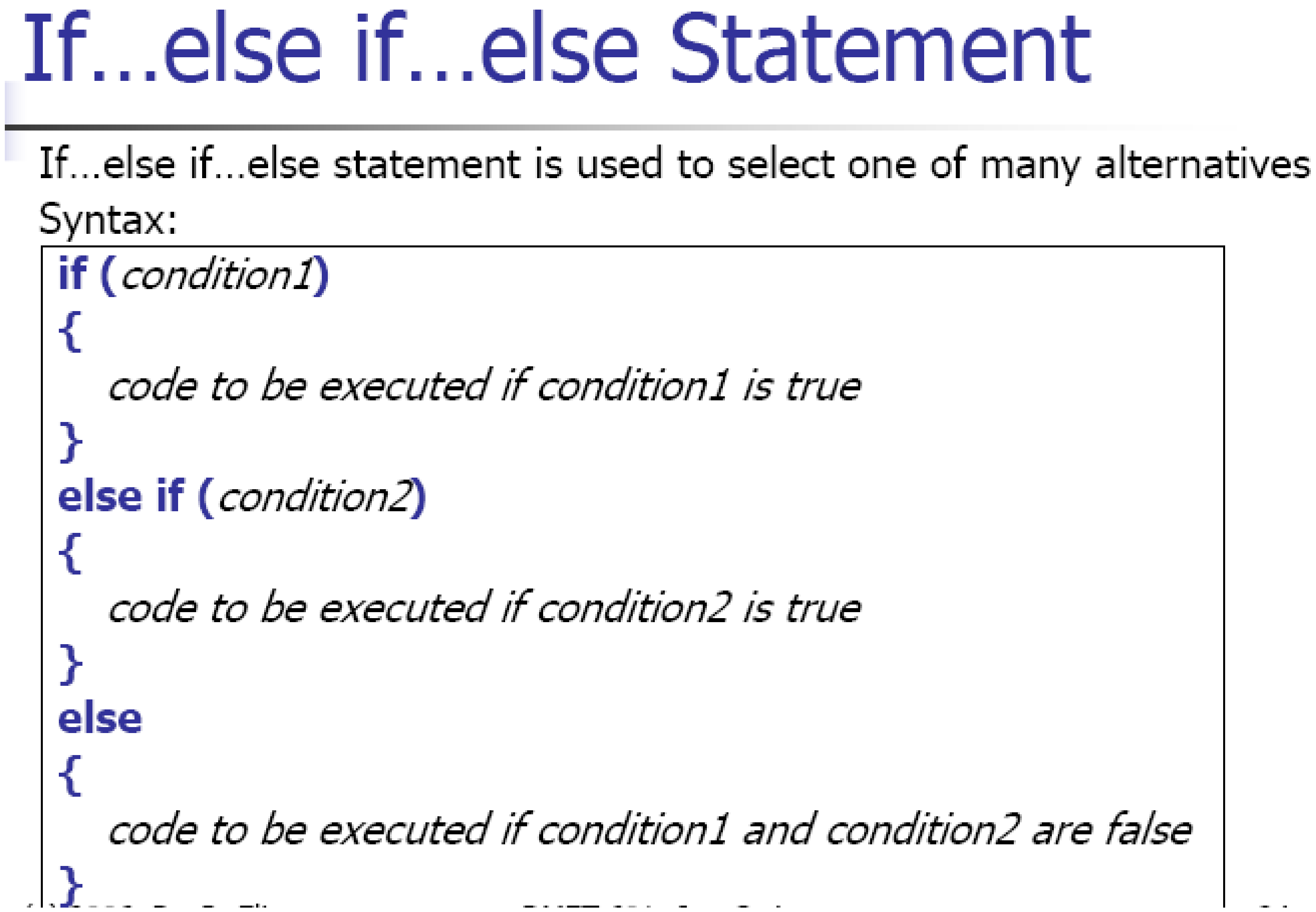
```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10

var d=new Date()
var time=d.getHours()

if (time<10) {
document.write("<b>Good morning</b>")
}
</script>
```

If…else Statement

- If…else statement is used to execute some code if some certain condition is true and another code if the condition is false
- Syntax:

```
if (condition)
{
   code to be executed if condition is true
}
else
{
   code to be executed if condition is false
}
```

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning"
// greeting. Otherwise you will get a "Good day" greeting.

var d = new Date()
var time = d.getHours()

if (time < 10) {
  document.write("Good morning!")
}
else
{
  document.write("Good day!")
}
</script>
```



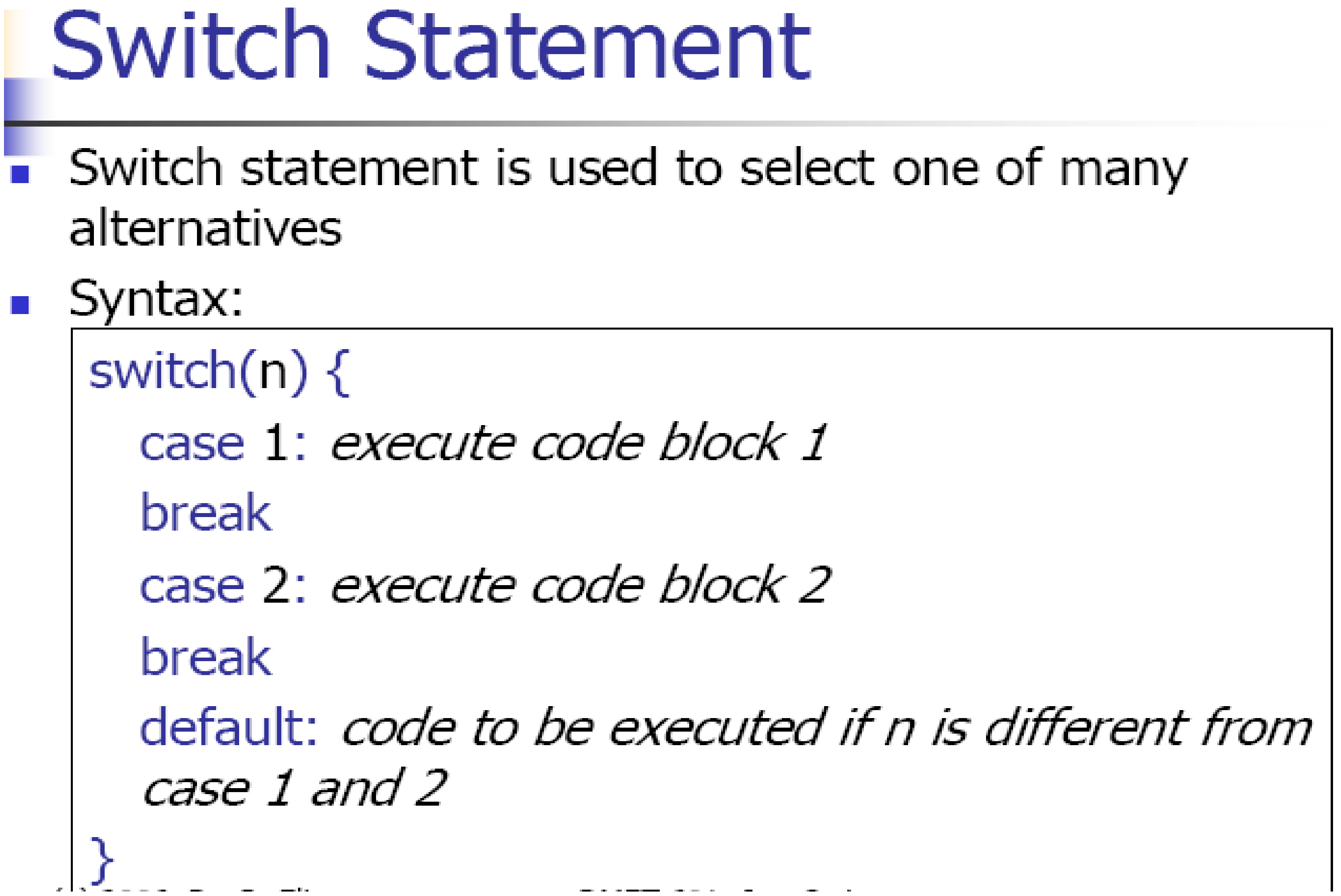
If...else if...else Statement

If...else if...else statement is used to select one of many alternatives

Syntax:

```
if (condition1)
{
    code to be executed if condition1 is true
}
else if (condition2)
{
    code to be executed if condition2 is true
}
else
{
    code to be executed if condition1 and condition2 are false
}
```

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning"
// greeting. Otherwise you will get a "Good day" greeting.

var d = new Date()
var time = d.getHours()

if (time < 10) {
   document.write("Good morning!")
}
else if (time>10 && time<16) {
   document.write("Good day!")
}
else {
   document.write("<b>Hello World!</b>")
}
</script>
```



Switch Statement

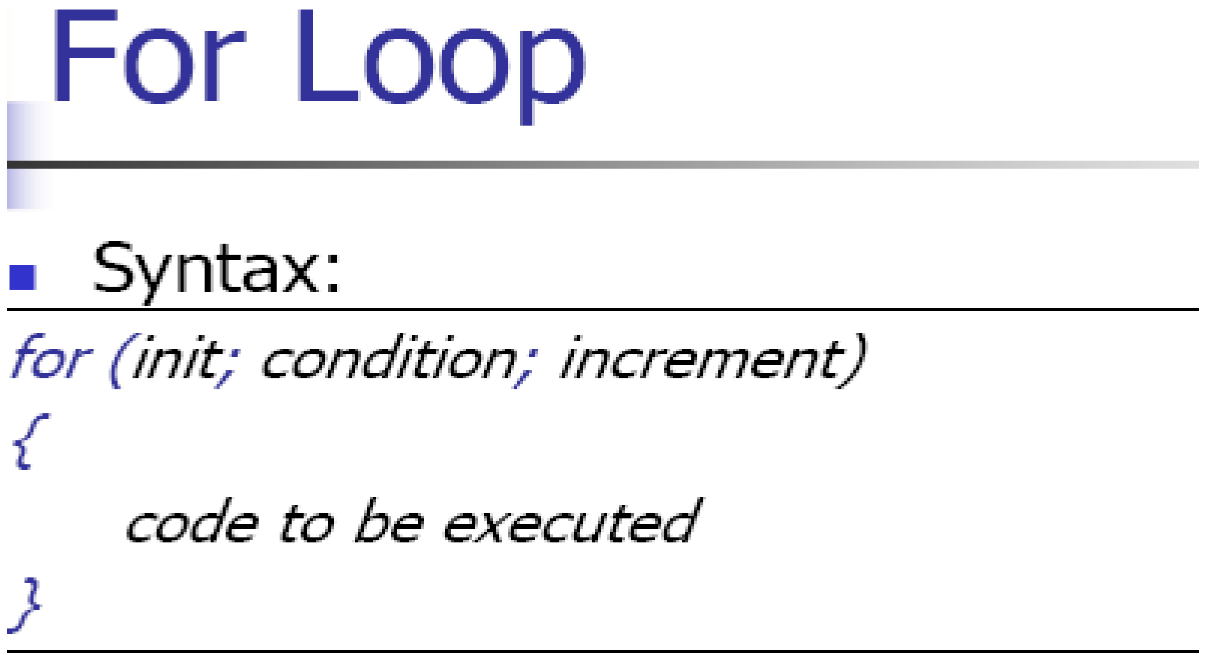
- Switch statement is used to select one of many alternatives
- Syntax:

```
switch(n) {
    case 1: execute code block 1
    break
    case 2: execute code block 2
    break
    default: code to be executed if n is different from
    case 1 and 2
}
```

```
<script type="text/javascript">
//You will receive a different greeting based on what day it is.
//Note that Sunday=0, Monday=1, Tuesday=2, etc.

var d=new Date()
theDay=d.getDay()
switch (theDay)
{
case 4:
  document.write("<b>Finally Thursday</b>")
  break
case 5:
  document.write("<b>Super Friday</b>")
  break
default:
  document.write("<b>I'm really looking forward to this
   weekend!</b>")
}
</script>
```

Loops in JavaScript

- In JavaScript, there are two different kind of loops:
  - **for** - loops through a block of code a specified number of times
  - **while** - loops through a block of code while a specified condition is true. The condition is tested at the beginning of the loop
  - **do..while** - loops through a block of code while a specified condition is true. The condition is tested at the end of the loop

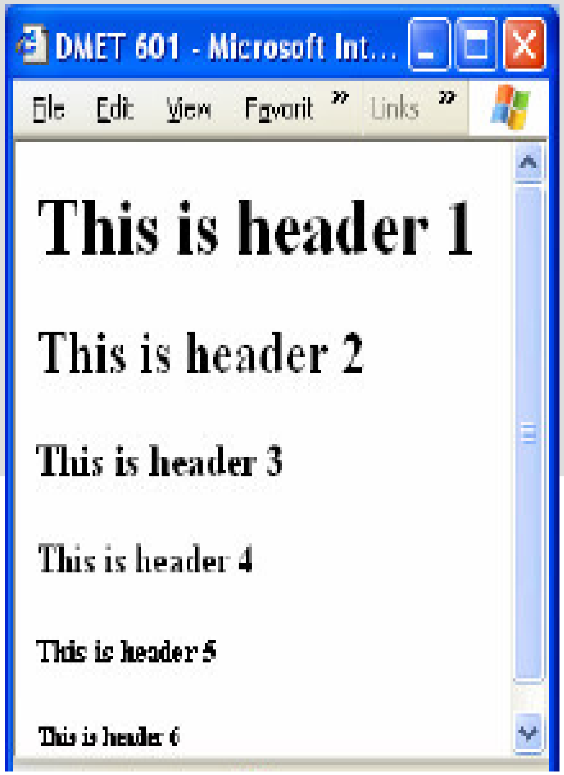


For Loop

- Syntax:

```
for (init; condition; increment)
{
    code to be executed
}
```

```
<html>
<body>

<script type="text/javascript">
   for (i = 1; i <= 6; i++){
       document.write("<h" + i + ">This is header " + i)
       document.write("</h" + i + ">")
   }
</script>

</body>
</html>
```



This is header 1

This is header 2

This is header 3

This is header 4

This is header 5

This is header 6

How to use the for loop to loop through the different HTML headers.

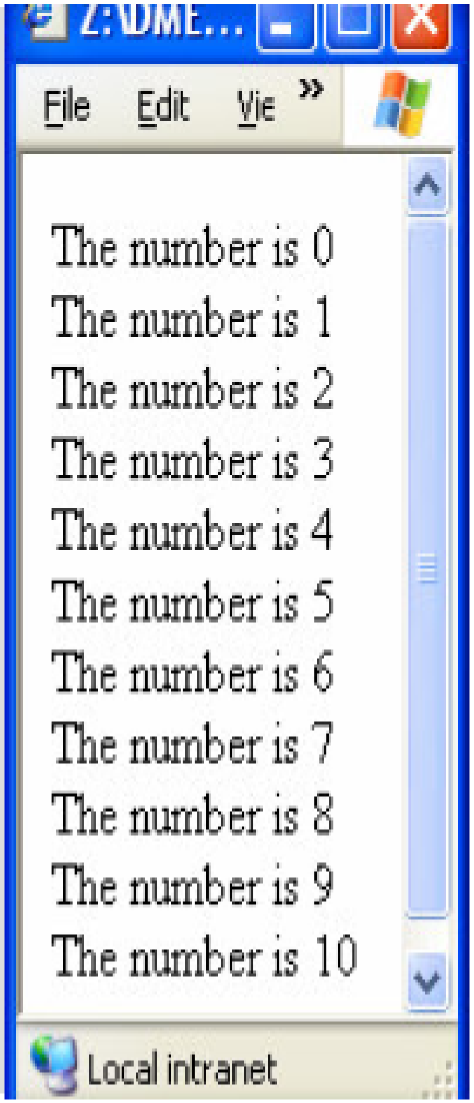
While Loop

- Syntax:

```
while (condition)
{
    code to be executed
}
```

The body of the loop will be executed repeatedly as long as the condition is true

```
<html>
<body>

<script type="text/javascript">
  var i=0
  while (i<=10)
  {
    document.write("The number is " + i)
    document.write("<br>")
    i=i+1
  }
</script>

</body>
</html>
```



```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

Do..while Loop

- Syntax:

```
do
{
    code to be executed
} while (condition)
```

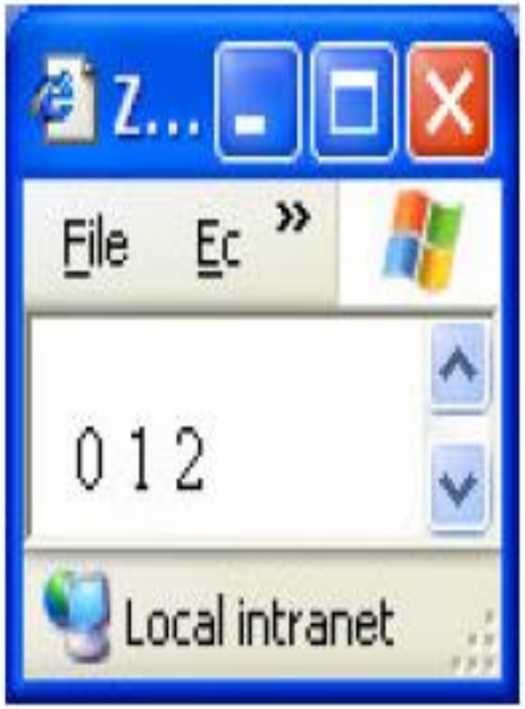
- The body of the loop will be executed at least one time. It will continue iterating as long as the condition is true

Break and Continue

- The break command will break the loop and continue executing the code that follows after the loop (if any).

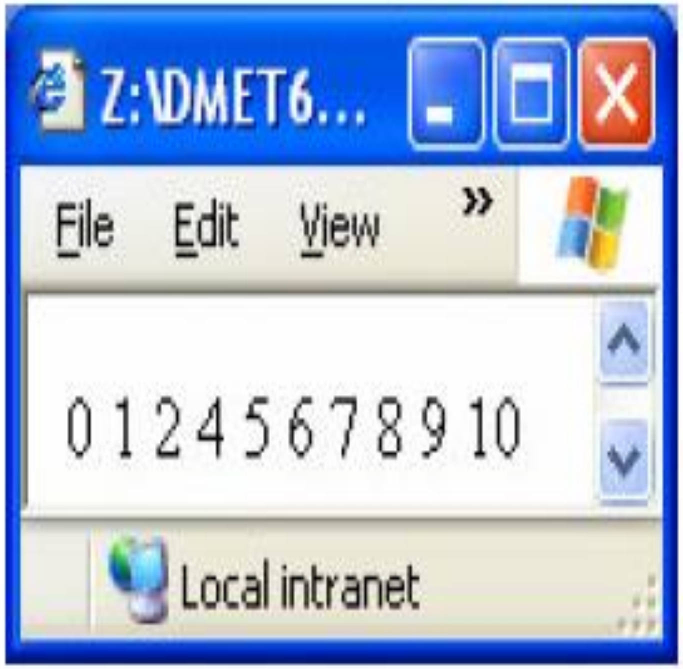
- The continue command will break the current loop and continue with the next value.

```
<html>
<body>

<script type="text/javascript">
  var i=0
  for (i=0;i<=10;i++) {
    if (i==3){break}
    document.write(i + " ")
  }
</script>
</body>
</html>
```



```
0 1 2
```

```
<html>
<body>

<script type="text/javascript">
  var i=0
  for (i=0;i<=10;i++) {
    if (i==3){continue}
    document.write(i + " ")
  }
</script>
</body>
</html>
```



```
0 1 2 4 5 6 7 8 9 10
```

Popup Boxes

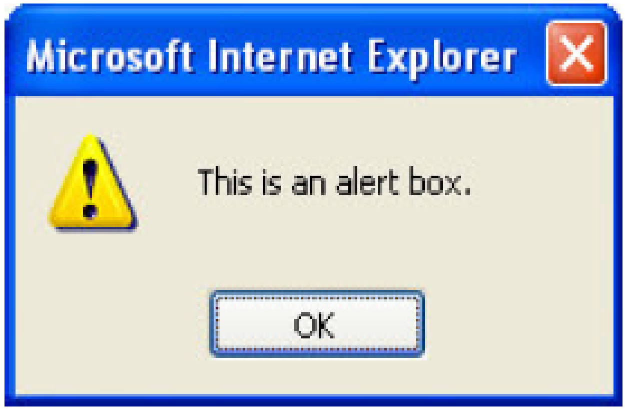
- There are three types of popup boxes:
  - Alert box
  - Confirm box
  - Prompt box

Alert Box

- Syntax:

```
alert("message")
```

```
alert("This is an alert box.")
```



This is an alert box.

- The user will have to click OK in order to proceed

- Are generally used for warnings.

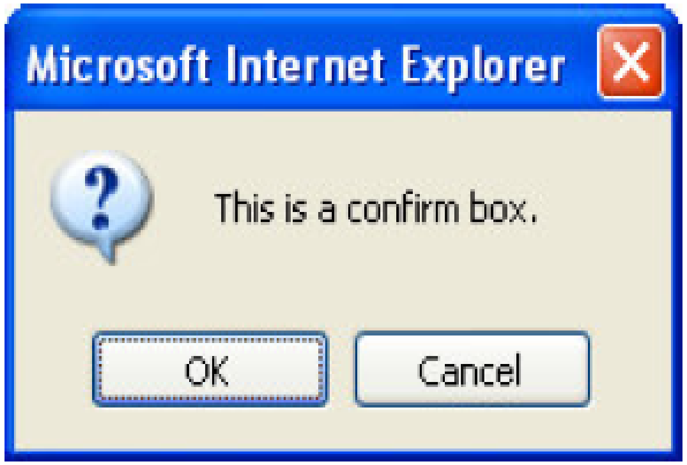
Confirm Box

.

- Syntax:

```
variable=confirm("message")
```

```
var=confirm("This is a confirm box.")
```



This is a confirm box.

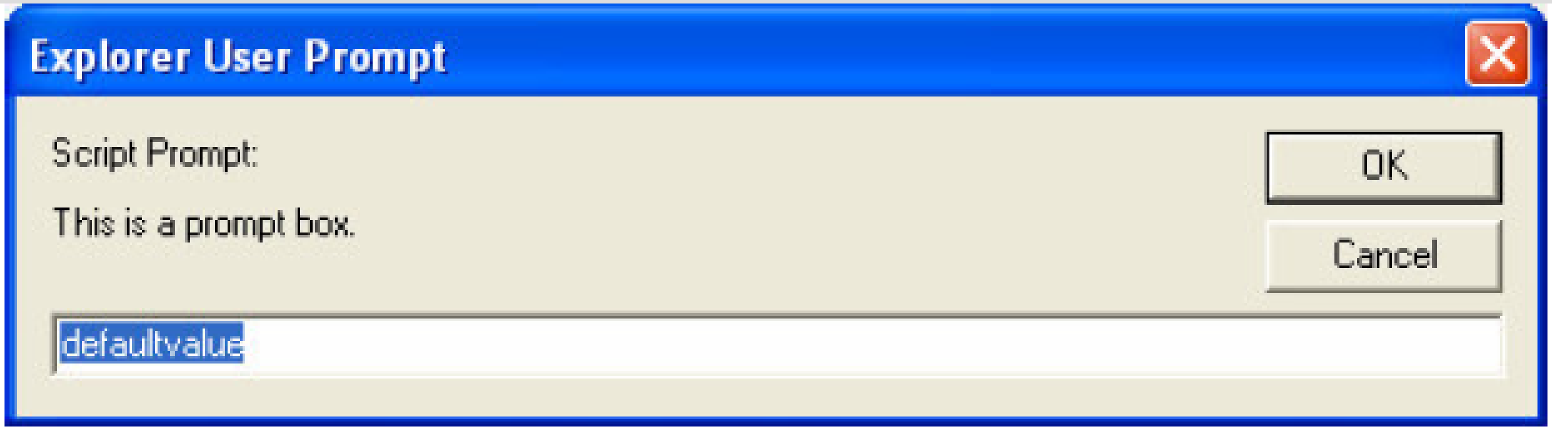
- The user will have to click OK or Cancel in order to proceed
- OK → returns true
- Cancel → returns false

Prompt Box

- Syntax:

```
variable=prompt("message","defaultvalue")
```

```
var=prompt("This is a prompt box.","defaultvalue")
```



Script Prompt:

This is a prompt box.

defaultvalue

OK → returns the input value

Cancel → returns null

Functions & Objects

Functions

- To keep the browser from executing a script as soon as the page is loaded, write your script as a function.
- A function contains some code that will be executed only by an event or by a call to that function.
- You may call a function from anywhere within the page
- Functions are defined at the beginning of a page, in the <head> section

Functions

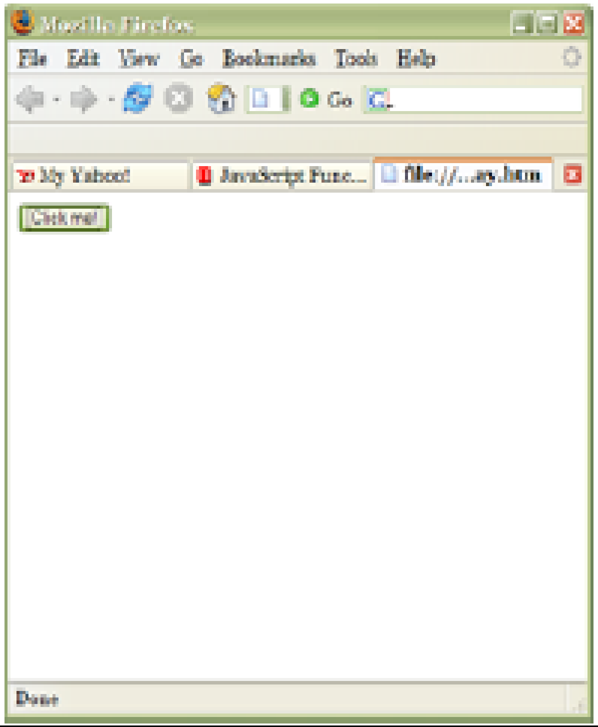
- A function is executed by an event, or when the function is called
- Syntax:

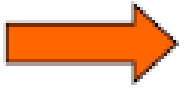
```
function functionname(var1, var2,...,varX)
{
    some code
}
```

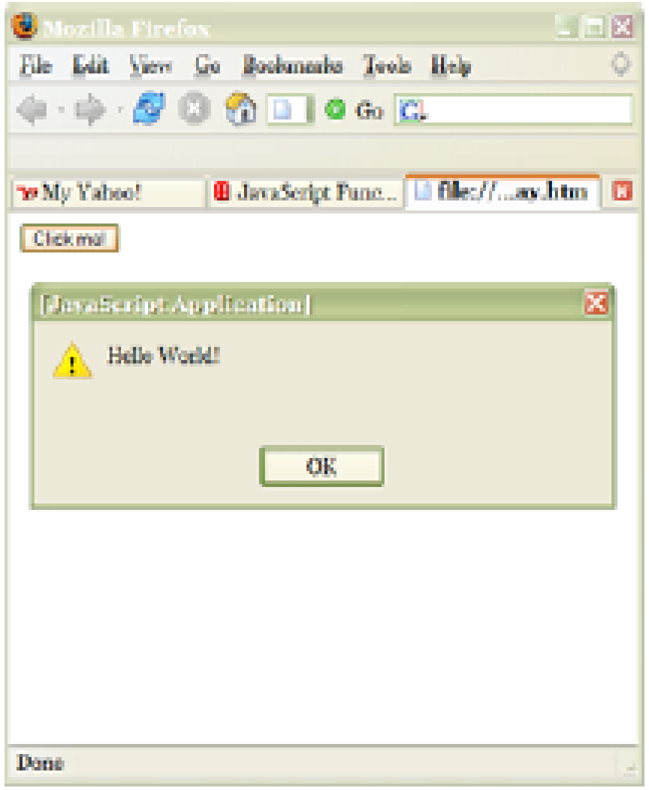
- Parameters are passed to the function through var1, var2, ..
- The parentheses are still needed even if no parameters are present

Function Example

```
<html> <head>
<script type="text/javascript"> function displaymessage() {
   alert("Hello World!") } </script> </head>
<body> <form>
<input type="button" value="Click me!" onclick="displaymessage()" >
</form> </body> </html>
```







Hello World!

16

Functions: return Statement

- Syntax:

```
return value
```

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

The called function

```
var = product(2,3)
```

The calling statement

```
<html>
<head>
<script language="javascript">
 function add()
{
   var a,b,c;
   a=document.calc.val1.value;
   b=document.calc.val2.value;
   c=parseInt(a)+parseInt(b);
   document.calc.result.value=c;
}
</script>
</head>
```

```
<body>
<form  name="calc">
enter text1:
 <input type="text" name="val1" size=20><br>

enter text2:
 <input type="text" name="val2" size=20><br>
<input type="button" value="SUBMIT"
   onclick="add()"><br>

result  :
<input type ="text" name="result"><br>
</form></body></html>
```

Exercises

- Find out factorial of a given number without using Recursion
- Same function Using Recursion
- Maximum of 3 numbers using Functions and also using predefined function Math.max()

Objects

- Objects have attributes and methods.
- Many pre-defined objects and object types.
- Using objects follows the syntax of Java:

```
objectname.attributename
objectname.methodname()
```

The `Math` Object

- **Access to mathematical functions and constants.**
- **Constants: `Math.PI`**
- **Methods:**

	`Math.random()`
`Math.abs(x),`	`Math.ceil(x)`
`Math.sin(x),`	`Math.floor(x)`
`Math.cos(x),`	`Math.exp(x)`
`Math.max(x,y),`	`Math.log(x)`
`Math.min(x,y),`	`Math.round(x),`
`Math.sqrt(x),`	`Math.pow(x,y)`

Math object in use

```
// returns an integer between 1 and 6
function roll() {
  var x = Math.random();

  // convert to range [0,6.0)
  x = x * 6;
  // add 1 and convert to int
  return parseInt(1+x );
}

document.writeln("Roll is “ + roll() );
```

The **String** Object

- **Access to String functions**

Methods:

var s1="Information",s2="Technology"

charAt(index), Ex: s1.charAt(2)

concat(string),Ex: s1.concat(s2)

slice(start,end), Ex: s1.slice(3,8)

Substr(start,length), Ex: s2.substr(1,4)

toLowerCase()Ex: s2.toLowerCase()

toUpperCase()Ex: s2.toUpperCase()

The Date Object

- Access to Date functions

Methods:

```
 var d= new Date()
getDate(); Ex: d.getDate();
getDay();        getSeconds();
getFullYear(); getTime();
getHours();      getMonth();
getMilliseconds();getMinutes();
```

Predefined Objects

- **JavaScript also includes some objects that are automatically created for you (always available).**
  - **`document`**
  - **`navigator`**
  - **`window`**

The **document** object

`Methods:`

- **`document.write()`** like a print statement – the output goes into the HTML document.

```
document.write("My title is" +
  document.title+ “URL is”
  +document.URL);
```

string concatenation!

JavaScript Example

```
<HEAD>
<TITLE>JavaScript is Javalicious</TITLE>
</HEAD>
<BODY>
<H3>I am a web page and here is my
  name:</H3>
<SCRIPT>
document.write(document.title);
</SCRIPT>
<HR>
```

The **`navigator`** Object

- Represents the browser. Read-only!
- Attributes include:

`appName`

`appVersion`

`platform`

often used to determine what kind of browser is being used (Netscape vs. IE)

`navigator` Example

- **alert(navigator.appName);**
- **alert(navigator.appVersion);**
- **alert(navigator.platform);**

The `window` Object

- Represents the current window.

- There are possible many objects of type **`Window`**, the predefined object **`window`** represents the current window.
- Access to, and control of, a number of properties including position and size.

some **`window`** methods

```
alert()
close()
prompt()
moveTo()    moveBy()
open()
scroll()    scrollTo()
resizeBy()  resizeTo()
```

Arrays

Array literals

- 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 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;

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**

Array example code

- **<script language="javascript">**
- **var a = [8,7,6,5];**

- **b = a.reverse();**
- **document.writeln(b);**
- **</script>**

The with statement

- **with (*object*) *statement* ; uses the *object* as the default prefix for variables in the *statement***
- **For example, the following are equivalent:**
  - **with (document.myForm) {**

    result.value = compute(myInput.value) ;

    }
  - **document.myForm.result.value =**

    compute(document.myForm.myInput.value);

for .. in statement

- **You can loop through all the properties of an object with for (variable in object) statement;**

```
<html>   <body>
     <script type = "text/javascript">
         var aProperty;
          document.write("Window Object Properties<br /> ");
          for (aProperty in window) {
             document.write(aProperty);
             document.write("<br />");
          }
          document.write ("Exiting from the loop!");
     </script>
     <p>Set the variable to different object and then try...</p>
   </body> </html>
```

Form Validation

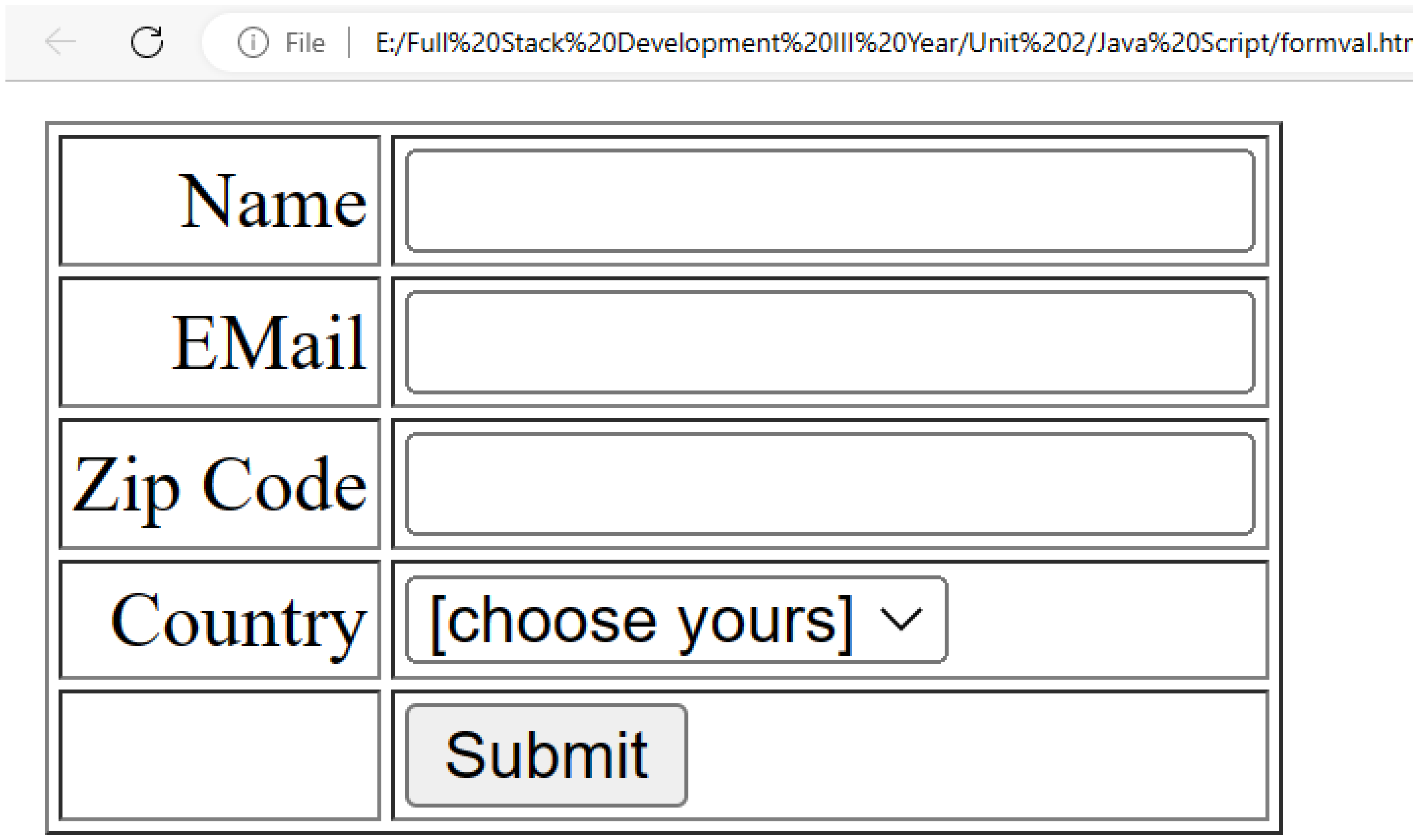
```
<html>
<head>
<title>Form Validation</title>
<script type="text/javascript">
function validate()
{

  if( document.myForm.Name.value == "" )
  {
    alert( "Please provide your name!" );
    document.myForm.Name.focus() ;
    return false;
  }
var emailID = document.myForm.EMail.value;
  atpos = emailID.indexOf("@");
  dotpos = emailID.lastIndexOf(".");
  if (atpos < 1 || ( dotpos - atpos < 2 ))
  {
     alert("Please enter correct email ID")
     document.myForm.EMail.focus() ;
     return false;
  }
```

```
if( document.myForm.Zip.value == "" ||
        isNaN( document.myForm.Zip.value ) ||
        document.myForm.Zip.value.length != 6 )
  {
    alert( "Please provide a zip in the format
######." );
    document.myForm.Zip.focus() ;
    return false;
  }
  if( document.myForm.Country.value == "-1" )
  {
    alert( "Please provide your country!" );
    return false;
  }
  return( true );
}

</script>
</head>
```

Form Validation

```
<body>
 <form action="/cgi-bin/test.cgi"
name="myForm"
         onsubmit="return(validate());">
 <table cellspacing="2" cellpadding="2"
border="1">
 <tr>
   <td align="right">Name</td>
   <td><input type="text" name="Name"
/></td>
 </tr>
 <tr>
   <td align="right">EMail</td>
   <td><input type="text" name="EMail"
/></td>
 </tr>
 <tr>
   <td align="right">Zip Code</td>
   <td><input type="text" name="Zip" /></td>
 </tr>
<tr>
 <td align="right">Country</td>
 <td>
 <select name="Country">
   <option value="-1" selected>[choose
yours]</option>
   <option value="1">USA</option>
   <option value="2">UK</option>
   <option value="3">INDIA</option>
 </select>
 </td>
 </tr>
 <tr>
   <td align="right"></td>
   <td><input type="submit" value="Submit"
/></td>
 </tr>
 </table>
 </form>
 </body>
 </html>
```

Form Validation



File | E:/Full%20Stack%20Development%20III%20Year/Unit%202/Java%20Script/formval.htr

Name	
EMail	
Zip Code	
Country	[choose yours]
	Submit

III UNIT

jQuery

- jQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.

jQuery

- Many of the biggest companies on the Web use jQuery, such as:
    - » Google
    - » Microsoft
    - » IBM
    - » Netflix
- **jQuery Features**
    - » HTML manipulation
    - » DOM manipulation
    - » DOM element selection
    - » CSS manipulation
    - » Effects and Animations
    - » Utilities

Any questions?