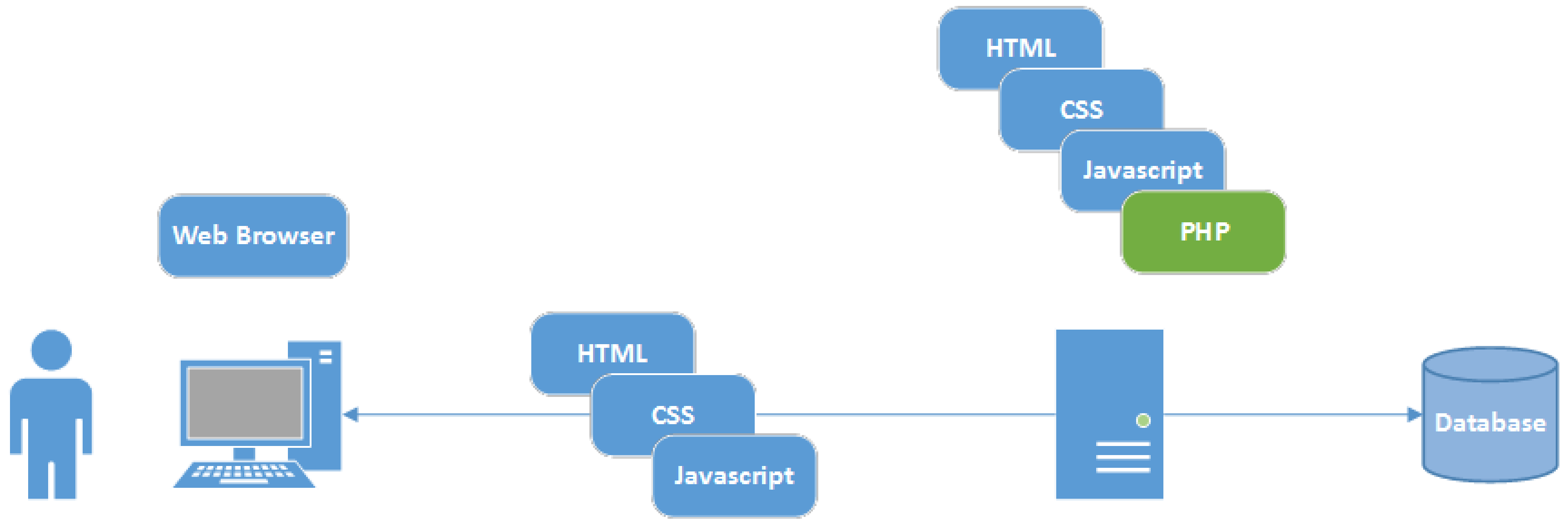
5. Scripting Languages

Outline

- JavaScript – client side scripting language
  - JavaScript Functions and Events
    - Event handling
  - Using JavaScript
    - Working with HTML Document Object Model (DOM)
    - Rollover graphics
    - Form validation
    - Pop-up windows and dialogs
- PHP – server side scripting language
  - PHP basics
    - Variables, arrays
    - Program control
    - File handling
  - Form handling

Web Applications



What makes a web site dynamic?

- Interactivity – adapt and react to the visitor’s actions as quick as possible.
- Synchronicity – bring together relevant information from a variety of sources.
- Flexibility – give the visitor different ways to find information in the site.
- Adaptability – adjust to cater to individual visitor’s needs.
- Activity – uses motion and sound to draw user’s attention to changes on the site.

Introduction to JavaScript

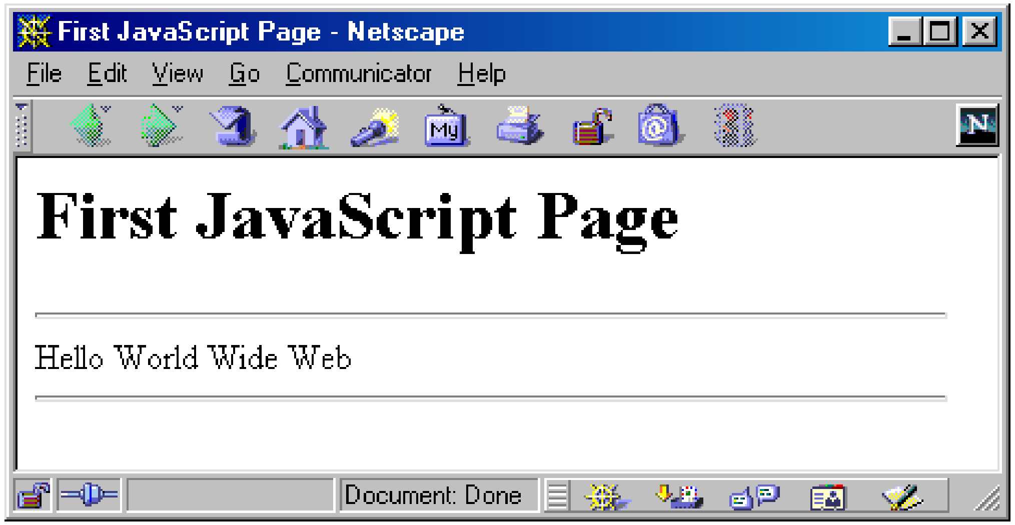
- **NOT Java**
  - JavaScript was developed by Netscape
  - Java was developed by Sun
- The growth of the WWW has resulted in a demand for dynamic and interactive web sites.
- Designed to ‘plug a gap’ in the techniques available for creating web-pages
  - Client-side dynamic content – support control of a browser and interactions with users
- JavaScript is a scripting language
  - a scripting language is a lightweight programming language
  - Interpreted: scripts execute without preliminary compilation
- There are many different kinds of scripting languages – JavaScript, Jscript, VBScript…
- JavaScript is *THE* most popular scripting language on the internet, and works in all major browsers.

JavaScript Capabilities

- JavaScript was designed to add interactivity to HTML pages
  - JavaScript is a simple language for fairly simple tasks
  - Best suited for tasks which run for a short time
  - Most commonly used to manipulate pieces of the Document Object Model (DOM)
- Everyone can use JavaScript without purchasing a license
- JavaScript gives HTML designers a programming tool to:
  - Improve the user interface of a website
  - Make your site easier to navigate
  - Easily create interactivity
  - Replace images on a page without reloading the page
  - Validate user input

A Simple JavaScript

```
<html>
<head>  <title>First JavaScript Page</title>
     </head>
<body>
<h1>First JavaScript Page</h1>
<script type="text/javascript">
<!--
document.write("<hr />");
document.write("Hello World Wide Web");
document.write("<hr />");
-->
</script>
</body>
</html>
```


First JavaScript Page - Netscape
File Edit View Go Communicator Help
Document: Done

First JavaScript Page

Hello World Wide Web

- To insert a script in an HTML page, we use the <script> tag.
- Use the type attribute to define the script language.
- Write output to the web page: document.write("text”) or document.writeln("text”).

Where is JavaScript placed?

- Scripts in the body section
  - Scripts will be executed when the page loads.
  - When a script is placed in the body section it can generate the content of the page
- Scripts in the head section
  - Scripts will be executed when they are called, or when an event is triggered.
  - When a script is placed in the head section, it is ensured that the script is loaded before anyone uses it.
- Scripts in both the body and the head section
  - You can place an unlimited number of scripts in your document.
  - You can have scripts in both the body and the head section.

Need for a Source File

- If the JavaScript code is fairly short, you are recommended to include the code in the XHTML/HTML document.
- Otherwise it is best to have it in a separate file:
  - To add clarity to an XHTML/HTML document.
  - To share JavaScript code across multiple HTML documents.
  - src attribute – specifies the location of an external script

```
<html>
<head>
<title>First JavaScript Program</title>
</head>
<body>
<script type="text/javascript" src="your_source_file.js"></script>
</body>
</html>
```

Hide JavaScript from Incompatible Browsers

```
<script type="text/javascript">
<!- begin hiding JavaScript
// single-line comment, /* … */ multiple-line comment
End hiding JavaScript
// -->
</script>
<noscript>
Your browser does not support JavaScript.
</noscript>
```

- Not all browsers support JavaScript.
  - e.g. NN1, IE2, character-based lynx.
- Adding comments
  - Single line comments start with //.
  - Multi-line comments start with /* and end with */.
- The **noscript** element is used to define an alternate content (text) if a script is NOT executed.

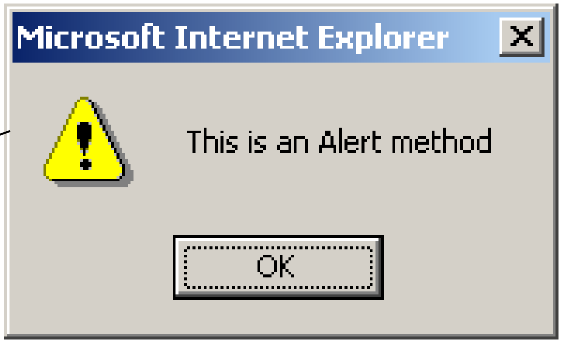
Screen Output and Keyboard Input

```
<head>
<script type="text/javaScript">
alert("An alert triggered by JavaScript");
</script>
</head>
```

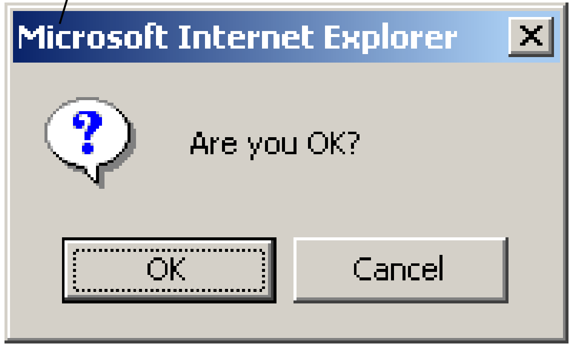
- The default object for JavaScript is the window object currently being displayed
- There are three methods that create dialog boxes for specific kinds of user interaction - alert(), prompt() and confirm().
- You can use alert() to display textual information to the user (simple and concise).
- String argument is the input
- The user can simply click **OK** to close it.

Three Methods

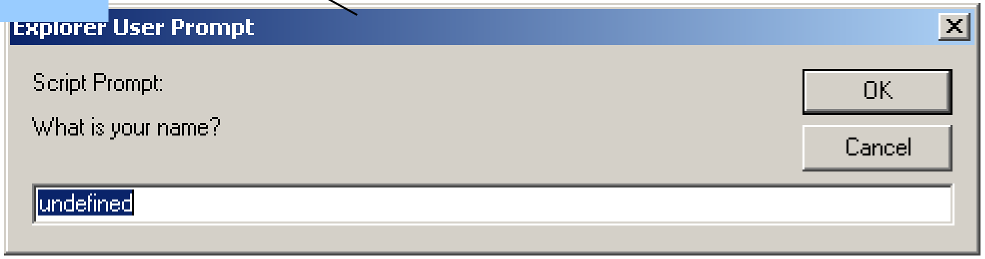
```
<script type="text/javascript">
alert("This is an Alert method");
confirm("Are you OK?");
prompt("What is your name?");
prompt("How old are you?","20");
</script>
```


Microsoft Internet Explorer

This is an Alert method

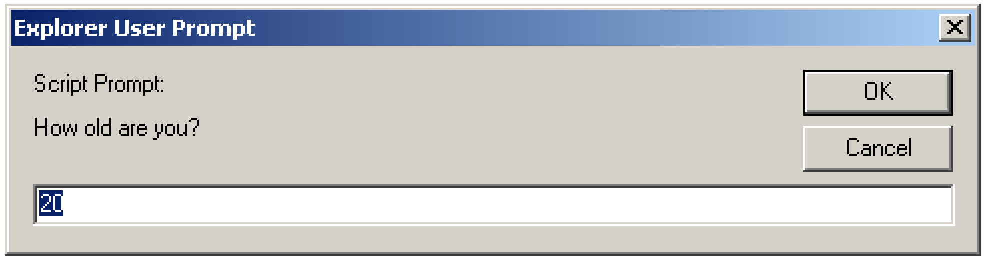

Microsoft Internet Explorer

Are you OK?


Explorer User Prompt

Script Prompt:
What is your name?

undefined


Explorer User Prompt

Script Prompt:
How old are you?

20

- These cause the browser to wait for a user response
- prompt() uses a second parameter to define the default value

JavaScript Basics

- JavaScript syntax is very similar to that of Java
- JavaScript has
  - variables, objects and functions (methods)
    - you **don’t have to assign a variable to a data type**
  - blocks of code {   }
  - input forms and dialog boxes
  - conditional structures (**`if...then...else`**)
  - iterative structures (**`while...`** and **`for...`**)
- JavaScript has events associated with HTML elements

The Basics: Arrays

- Arrays in JavaScript are objects that have some special functionality
- Array declarations:

```
var values = new Array(100);
var nums = [3, 6, 66, 3, 8, 10, 99];
Var days = [“Monday”,”Tuesday”,”Wednesday”];
```

- Example

```
var nums = new Array(1,2,3);
document.writeln(“first value: " + nums[0] + “<br>”);
document.writeln(“second value: " + nums[1] + “<br>”);
document.writeln(“third value: " + nums[2] + “<br>”);
```

- Note that arrays begin at index `0(zero)`
- Note that variables may be included in strings using **+**

The Basics: for loops

for(initialise counter; test condition; increment**)**

```
{
   do something;
}
```

```
var i;
var myArray = [1,1,2,3,5,8,13];
for(i=0; i<myArray.length(); i++) {
  document.writeln("value is " + myArray[i]);
  document.writeln("<br>");
}
```

Note that you can use the **`length()`** function to get the number of values in an array, e.g.

```
Var arrayLength = myArray.length();
```

while loops and do while loops

while (condition is true) {do something }

```
count=0;
while(count < 5) {
  document.writeln("value is " + count);
  document.writeln("<br>");
  count = count + 1;
}
```

do {something} **while** (condition is true) ;

```
count=0;
do {
  document.writeln("value is " + count);
  document.writeln("<br>");
  count = count + 1;
} while (count < 5);
```

The Basics: `if` statements

`if` (condition) {do something}

```
if(scoreA > scoreB) {
  document.writeln("The winner is A")
}
```

```
if(scoreA > scoreB) {
  document.writeln("The winner is A")
}
else if (scoreA < scoreB) {
  document.writeln("The winner is B")
}
else {
  document.writeln("Everyone's a winner")
}
```

Global and Local Variables

- Any variables declared within a function are ***local*** to that function
- Any variables declared outside a function are ***global*** – they may be used anywhere in the program/page
- You cannot use variables declared as local outside of the function they are declared in
- If you wanted you could use only global variables – but this is very bad programming practice.
- If a variable that is defined both as a local variable and as a global variable, appears in a function, <u>the local variable has precedence.</u>

Example

```
<head><script type="text/javascript">
    <!--
   var global_one = 1; var global_two = 100;
   function doSomething(){
    //local_one can only be used here!
    var local_one = 10; var global_two = 10;
    document.writeln("local_one="+local_one+"<br>");
    //can also use global_one - its a global variable
    document.writeln("global_one="+global_one+"<br>");
    document.writeln("global_two="+global_two+"<br>");
   }
   -->
   </script>
    </head><body>
   <script type="text/javascript">
    doSomething();
    //I can use global_one here - not local_one
    document.writeln("global_one="+global_one+"<br>");
    document.writeln("global_two="+global_two);
   </script></body>
```

What is the Value of global_two?

Class and Object

- One of the most important features of JavaScript - enables modular programs.
  - Objects are based on Classes; variables, functions, statements are contained in a structure called class.
- You can use the "new" operator to create instances of objects of a particular class or object type.
  - `variable = new objectType(parameters)`
- This objectType() is called constructor.
- E.g. Date is a predefined JavaScript class.
  - Assign the current date and time to objA `objA = new Date()`

Object	Description
Array	Creates new array objects
Boolean	Creates new Boolean objects
Date	Retrieves and manipulates dates and times
Error	Returns run-time error information
Function	Creates new function objects
Math	Contains methods and properties for performing mathematical calculations
Number	Contains methods and properties for manipulating numbers.
String	Contains methods and properties for manipulating text strings

Some More Basics

Can use `Math` class for powerful built in functions:

- `abs(value)` - returns the absolute value (modulus)
- `sin(value)/cos(value)/tan(value)` - trigonometric functions
- `random()` - returns a pseudorandom number between 0 and 1
- `pow(value,power)` – result of value raised to power
- `sqrt(value)` – returns square root of the number e.g. `var result = Math.sqrt(number);`

`String Manipulation`

- ❒ `charAt(index) e.g. myString.charAt(index);`
  - returns the character at position **`index`**
- ❒ `concat(string) e.g. both = str1.concat(str2);`
  - concatenates two strings
- ❒ `length() e.g. size = str1.length();`
  - returns the number of characters
- ❒ `split(separator) e.g. bits = str.split(“.”);`
  - breaks the string apart whenever it encounters the **`separator`** character (pieces returned in an array)

Functions

- Functions are integral to JavaScript and functions are typically called in response to user actions
  - This adds interactivity to pages
- Functions have to be declared within the `head` of a document
  - Can then be used anywhere in a document (in the body, in another function, etc)

```
<head>
 <title>Example 1</title>
 <script type="text/javascript">
 <!--
  function showMessage()
  {
   document.writeln("<h3>Hi There..</h3>");
  }
  //Can also write more functions here if required..
  -->
 </script>
</head>
<body>
 <script type="text/javascript">
   showMessage();
 </script>
</body>
```

Functions

- A function definition consists of the function’s header (having the reserved word *function*) and a compound (*body*) statement that describes its action.
- You can reuse functions within the same script, or in other documents.
- To create a function you define its name, any values (“arguments”), and some statement.
- Some functions return a value to the calling expression
    - Functions that will return a result must use the return statement.
- A function is not executed before it is called.
- Built-in and user-defined functions

Built-In Functions

- Functions provided by the language and you cannot change them to suit your needs.
- Some of the built-in functions in JavaScript are shown here:
  - eval - eval(expr)
    - eval takes a string as its argument and evaluates the expression or statements
    - eval("24*4/8")=12
  - isFinite
    - Determines if a number is finite
  - isNaN
    - Determines whether a value is "Not a Number"
  - parseInt
    - parseInt("124") will return 124.
    - Converts string literals to integers, no number → NaN.
  - parseFloat
    - Finds a floating-point value at the beginning of a string.

Events

- Use of JavaScript for Web programming is to detect certain activities of the browser and the user
- Events are the actions that occur as a result of browser activities or user interaction with web pages.
  - Such as the user performs an action (mouse click or enters data)
  - We can validate the data entered by a user in a web form
  - Communicate with Java applets and browser plug-ins
- Events are normally used in combination with functions, and the function will not be executed before the event occurs!
- Event Categories
  - Keyboard and mouse events
    - Capturing a mouse event is simple
  - Load events
    - The page first appears on the screen: “Loads”, leaves: “Unloads”, …
  - Form-related events
    - onFocus() refers to placing the cursor into the text input in the form.
  - Others
    - Errors, window resizing.

Events defined by JavaScript

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

Event Handlers

- When an event occurs, a code segment that is executed in response to a specific event is called “event handler”.
- Event handler names are quite similar to the name of events they handle.
- Different event triggers are associated with different HTML elements.
  - E.g the event handler for the “click” event is “onclick”.
- The process of connecting an event handler to an event is called registration.
- HTML elements support attributes for specifying code to run when a particular event happens.
  - Attribute name specifies event type.
  - Attribute value specifies code to run.
- Examples:

```
<a onclick=“processClick()”>My link</a>
<a onclick=“var a; a = 3; doThis(); doThat();”>Another Link</a>
```

Selected Event Handlers

- `onClick`: when the mouse button is clicked on an element (used with the **`button`** and **`link`** elements)
- `onMouseOver/onMouseOut`: when the mouse moves into/ out of an element (used with the **`link`**, **`image`** and **`layer`** elements).
- `onMouseDown/onMouseUp`: when the mouse button is pressed/released.
- `onload/onunload`: when browser loads/finishes with a document (used with the **`body`** element).
- `onFocus/onBlur` : when an element is selected/deselected (i.e. another element is selected) with the mouse (used with the **`input`**, **`select`**, **`textarea`** and **`button`** elements).
- `onSubmit`: when the submit button pressed in a form (used with the **`form`** element).

A simple event handler

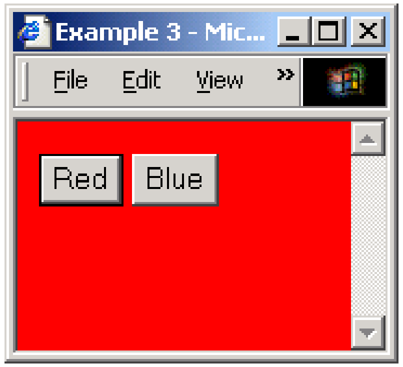
-
  ```
  <form method="post" action="">
   <input type="button"
              name="myButton"
              value="Click me"
              onclick="alert('You clicked the button!');">
  </form>
  ```
  - The button is enclosed in a form
    - method tells *how* to send the form data; action tells *where* to send it
  - The tag is input with attribute type="button"
  - The name can be used by other JavaScript code
  - The value is what appears on the button
  - onclick is the name of the event being handled
    - The value of the **onclick** element is the JavaScript code to execute
    - **alert** pops up an alert box with the given text

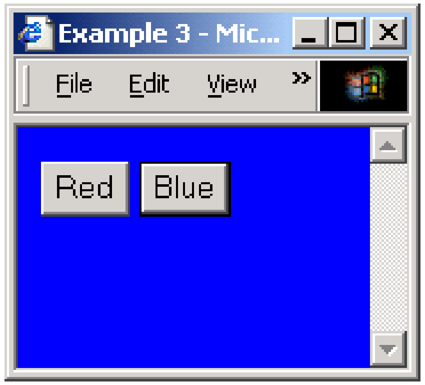
onClick Event Handler

```
<html>
<head>
<title>onClick Event Handler Example</title>
<script type="text/javascript”>
function warnUser(){   return confirm(“RHUL students?”); }
</script>
</head>
<body>
<a href=“reference.html”  onClick=“return warnUser();”>RHUL
Students access only</a>
</body>
</html>
```

onclick example

```
<head>
 <script type="text/javascript">
 <!--
  function change(col) {
    if(col=="red") {
      document.bgColor = col;
    }
    if(col=="blue") {
      document.bgColor = col;
    }
  }
 //-->
 </script>
</head>
<body>
 <form>
 <input type="button" value="Red" onClick=change("red")>
 <input type="button" value="Blue" onClick=change("blue")>
 </form>
</body>
```





onLoad Event Handler

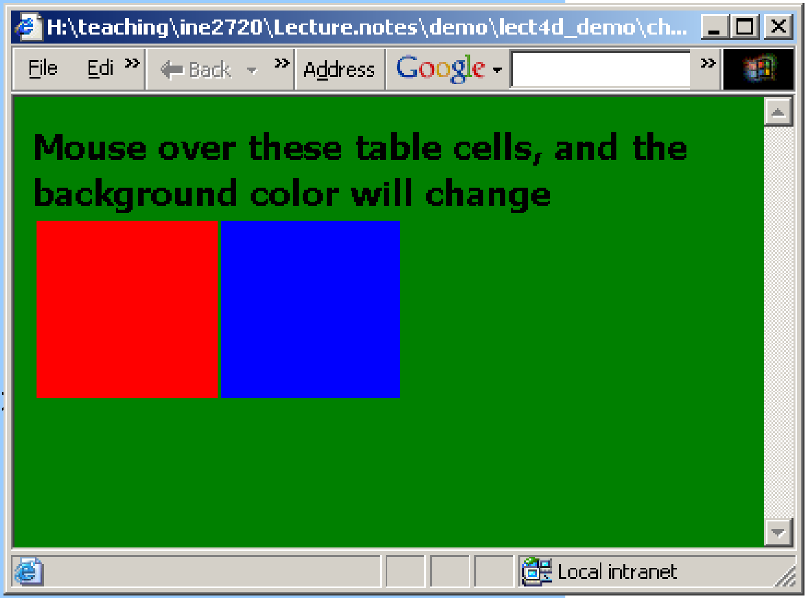
```
<html>
<head>
<title>onLoad and onUnload Event Handler Example</title>
</head>
<body onLoad="alert('Welcome User');"
onUnload="alert('Thanks for visiting the page');">
Load and UnLoad event test.
</body>
</html>
```

onMouseOver and onMouseOut Event Handlers

```
<html>
<head>
<title>onMouseOver and onMouseOut event handler</title>
</head>
<body>
<a href= "link.html"
onMouseOver = "status = 'Now mouse is over the link';  return true"
onMouseOut = "status = 'Mouse has moved out'; return true">
A Link Page
</a>
</body>
</html>
```

Change Background Color

```
<html><head>
<script type="text/javascript">
function bgChange(bg)
{      document.body.style.background=bg; }
</script></head>
<body><b>Mouse over these table cells, and the background color will change</b>
<table width="300" height="100">
 <tr>
  <td onmouseover="bgChange('red')"
      onmouseout="bgChange('transparent')" bgcolor="red">
  </td>
  <td onmouseover="bgChange('blue')"
      onmouseout="bgChange('transparent')" bgcolor="blue">
  </td>
  <td onmouseover="bgChange('green')"
      onmouseout="bgChange('transparent')" bgcolor="green"
  </td>
</tr>
</table>
</body></html>
```



Mouse over these table cells, and the background color will change

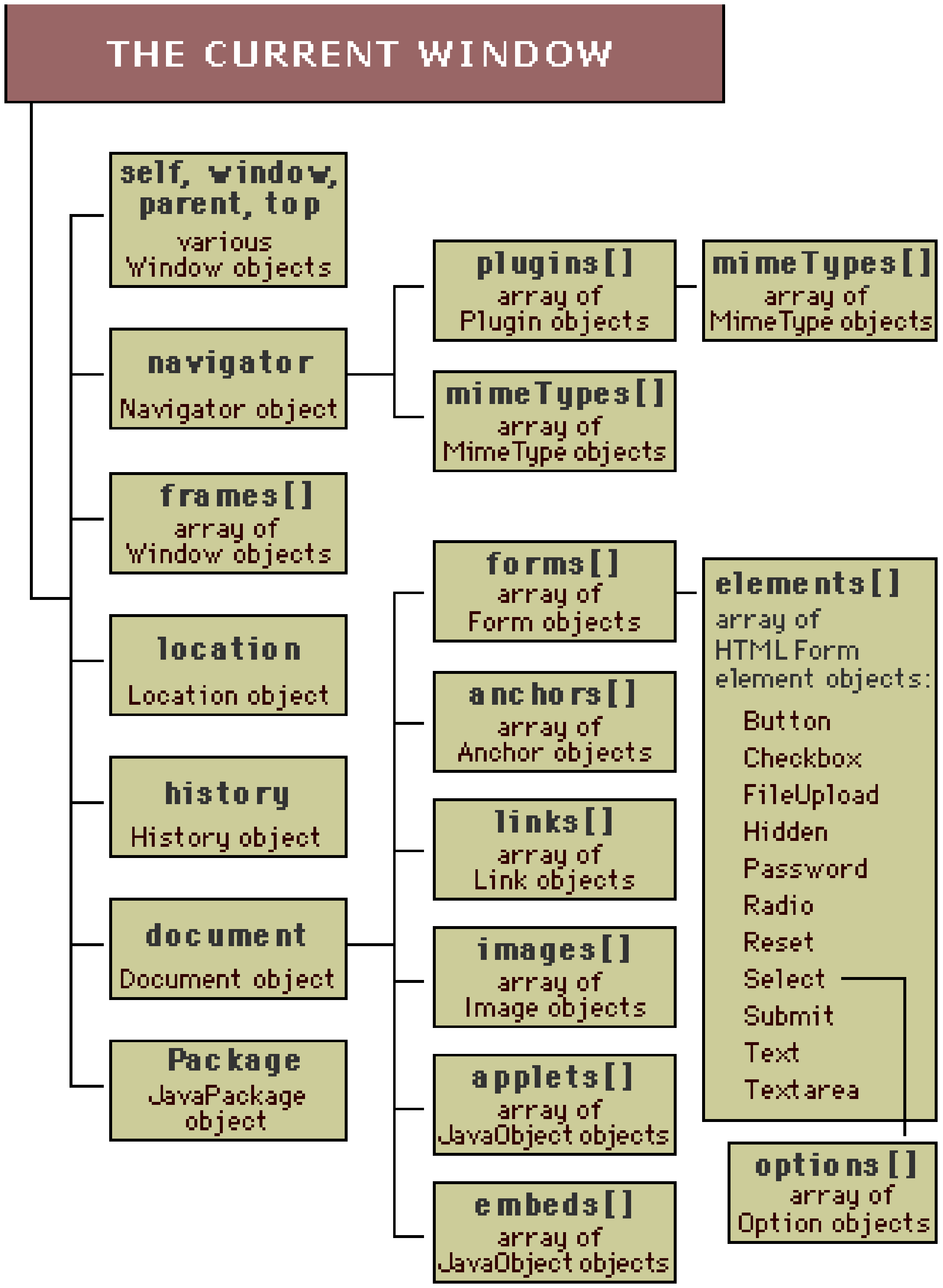
Document Object Model (DOM)

- Using JavaScript we can directly manipulate parts of a web page – making pages dynamic and interactive
- How is this done?
  - JavaScript regards a webpage as a collection of objects
  - We can manipulate object properties – thus manipulating the appearance of the web page
- Objects are arranged into a hierarchy of objects called the Document Object Model (DOM)
- At the top of the hierarchy is the window object – this is your browser window
- The document object is a child of the window object – this is the current webpage being displayed

DOM

- You can attach event handlers to HTML elements with very little knowledge of the DOM
- However, to *change* what is displayed on the page requires knowledge of how to refer to the various elements
- The basic DOM is a W3C standard and is consistent across various browsers
  - More complex features are browser-dependent
- The highest level element (for the current page) is window, and everything else descends from that
  - Every JavaScript variable is a field of some object
  - In the DOM, all variables are assumed to start with “window.”
  - All other elements can be reached by working down from there

The DOM hierarchy



Referring to Document Objects

- Each object has properties allowing us to access the lower level objects in the hierarchy.
  - e.g., `document.forms` returns an array of forms in the document.
- Normally an array of objects is returned, and we access a specific one (e.g., first form) using array subscripting.
  - document.forms[0]
- We can access the input elements using "elements":
  - document.forms[0].elements[0]
- DOM 1 has a set of very general methods for accessing any element on the page. E.g.,
  - document.getElementById(ID) – ID specified by id attribute
    - <p id=“para1”>Welcome to our web page!</p>
    - document.getElementById(‘para1').
  - document.getElementsByTagName(Name)
- Also general properties such as "parentNode”, “firstChild”, and “lastChild".

The “window” Object

- It is the highest-level object in the JavaScript browser object hierarchy.
  - It is also the default object and is created automatically when a page is loaded.
- Since it is the default object, we may omit writing window explicitly.
  - document.write(“a test message”);
  - window.document.write(“a test message”);
- It also includes several properties and methods for us to manipulate the page.

Property	Description
length	An integer value representing the number of frames in the window
name	A string value containing the name of a window
parent	A string value containing the name of the parent window
status	A string value representing status bar text

Method	Description
alert(text)	Pop up a window with "text” as the message
close()	Closes the current window
open(url)	Open a new window populated by a URL.
setTimeout(expression, time)	Executes an expression after the elapse of the interval time.

The “document” Object

- It is one of the important objects in any window or frame.
- The document object represents a web document or a page in a browser window.

Property	Description
bgColor	A string value representing the background color of a document
alinkColor	A string value representing the color for active links
Location	A string value representing the current URL
Title	A string value representing the text specified by <title> tag

Method	Description
clear()	Clears the document window
write(content)	Writes the text of content to a document
writeln()	Writes the text and followed by a carriage return
open()	Open a document to receive data from a write() stream
close()	Closes a write() stream

Manipulating the `document` object

- Each object has certain *properties* and *methods*
- The **`document`** object has the following properties/methods:
  - **`writeln`** – method to write out to the page
  - **`write`** – method to write out to the page
  - **`bgColor`** – change the body background colour property
  - **`fgColor`** – change the body foreground colour property
- The following program accesses and manipulates properties of the **`document`** object

```
<html>
   <head></head>
      <body bgColor="red">
         <script type="text/javascript">
           <!--
            var backcolour = document.bgColor;
            document.write("The background colour is: "+backcolour);
            alert("Press to change from "+backcolour+" to yellow");
   document.bgColor = "yellow";
   document.writeln("<br>Colour is now "+document.bgColor);
           -->
         </script>
      </body>
</html>
```

Example: Simple Calculator

```
<head><script type="text/javascript">
 <!--
  function addNumbers() {
    var num1 = document.forms[0].elements[0].value;
    var num2 = document.forms[0].elements[1].value;
    var result = parseInt(num1) + parseInt(num2);
    alert("Result: " + result);
}
 //-->
 </script></head><body>
 <form method="POST">
  Number 1 <input type="text" name="number_1"><br>
  Number 2 <input type="text" name="number_2"><br>
  <input type="button" value="Add“
   onClick="addNumbers()">
 </form></body>
```

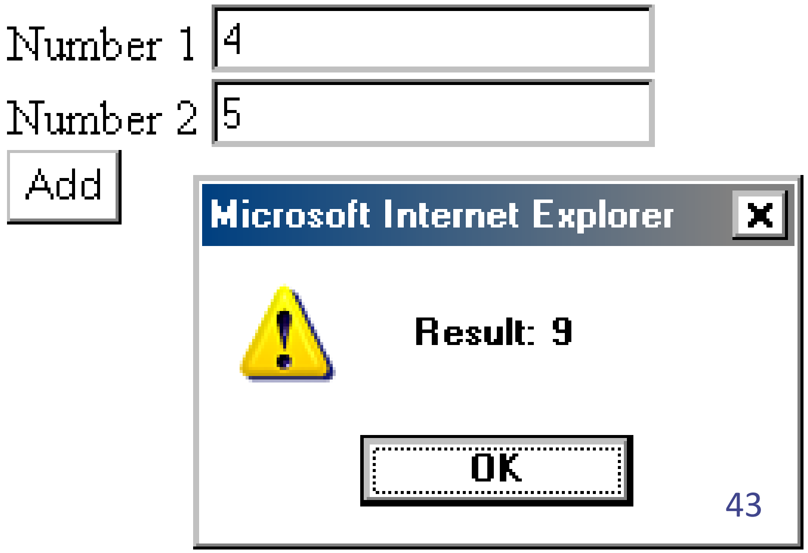
Example: Simple Calculator

- Pressing the button fires event **`onClick`**
- This calls **`addNumbers()`**

- Values entered into the text boxes are read using:

```
var num1 = document.forms[0].elements[0].value;
var num2 = document.forms[0].elements[1].value;
```

- These values are initially strings
- They are converted to Integers using **`parseInt()`**


Number 1 4
Number 2 5

Result: 9

Further modification of DOM objects

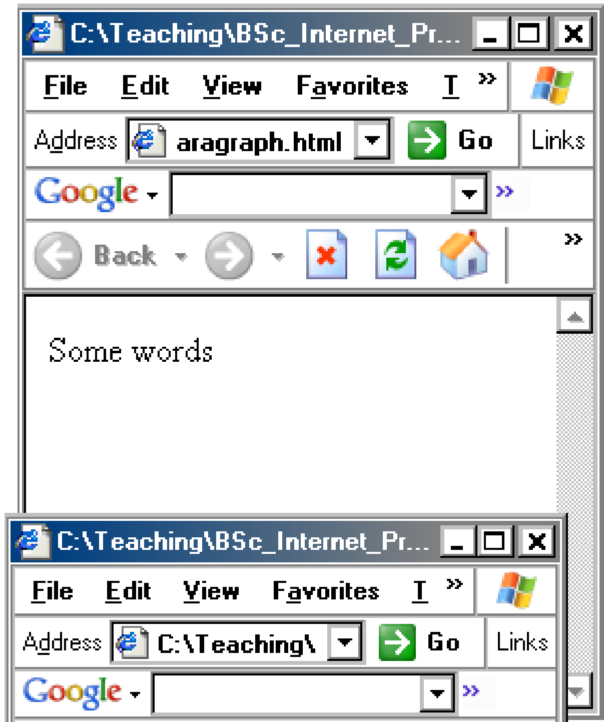
- We can also access, modify and set the style attribute of page elements
- We first give an element a unique id
- We modify style attributes with reference to this id
- Through the id we can also alter the document content of an element e.g. we may change the text between some <p> tags or <b> tags, etc

- Note that we can also associate events with tags such as <p> and <b>, etc

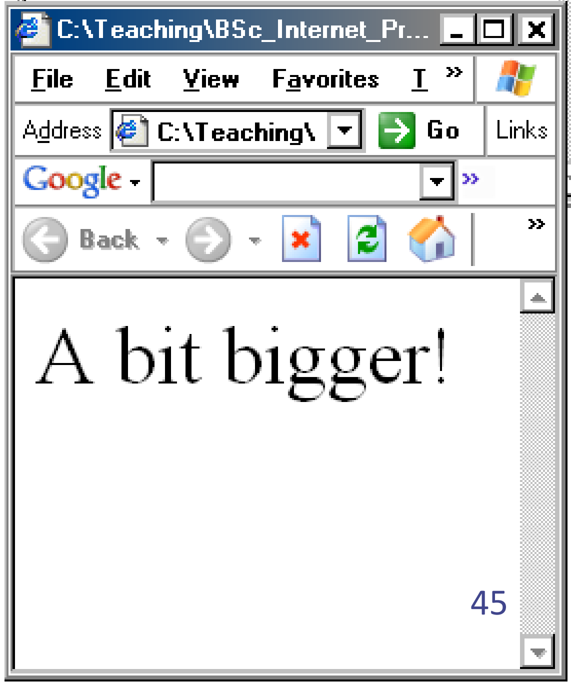
Example 1

```
<html>
 <head>
  <script type="text/javascript">
  <!--
    function move(x,y) {
      para1.innerHTML="A bit bigger!";
      para1.style.fontSize = 40;
    }
  -->
  </script>
 </head>
 <body>
  <p id="para1" onClick="move(50,100)">
    Some words
  </p>
 </body>
</html>
```

- **innerHTML** *sets or gets all of the markup and content within a given element.*
- **style** *object represents an individual style statement.*



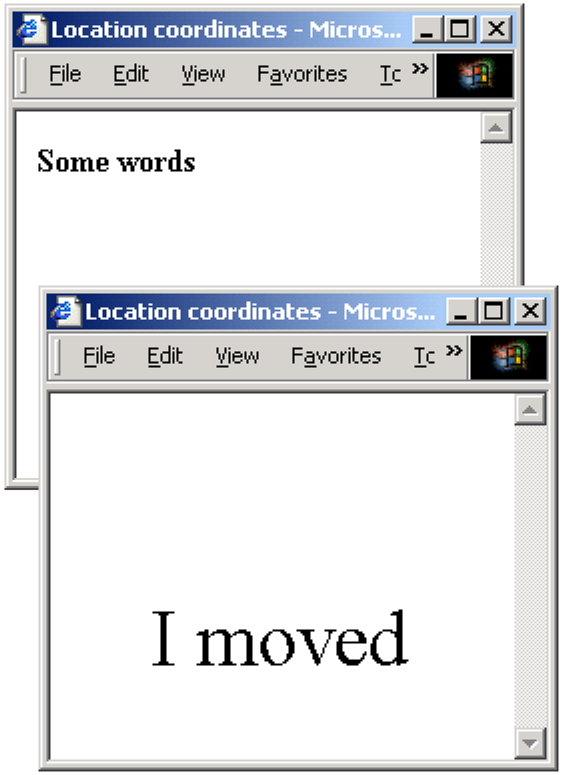
Some words



A bit bigger!

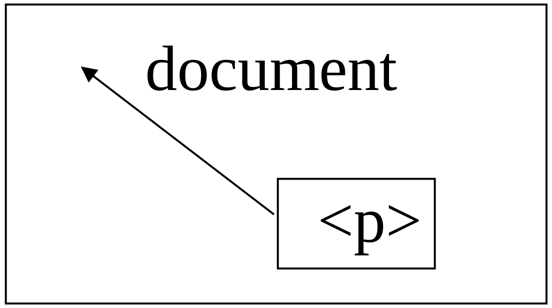
Example 2

```
<html>
 <head>
  <title>Location</title>
  <style>
    #para1 {position: absolute}
  </style>
  <script type="text/javascript">
  <!--
    function move(elementid,x,y) {
      elementid.style.left = x;
      elementid.style.top = y;
      elementid.innerHTML="I moved";
      elementid.style.fontSize = 40;
    }
  -->
  </script>
 </head>
 <body>
  <p id="para1" onMouseOver="move(this,50,100)">
    Some words
  </p>
 </body>
</html>
```


Location coordinates - Micros...
File Edit View Favorites Tc
Some words
Location coordinates - Micros...
File Edit View Favorites Tc
I moved

Event Propagation (DOM2)

- An event object is created at a node in the document tree.
  - For that event, the node is called the target node.
- Capturing phase: the event starts at the document root node and propagates down the tree to the target node
  - An event may be pre-processed, handled or redirected by any intervening object.
- Execution of any appropriate handlers at the target node
- Bubbling phase: in which events fired in child elements “bubble” up to the child’s event handler, then to the parent’s event handler.
  - If you intend to handle an event in a child element, you might need to cancel the bubbling of the event in the child element’s event-handling code by using the cancelBubble property of the event object.
- Example



`setInterval` and `clearInterval`

- Used to call JavaScript code repeatedly at a specified time interval

```
setInterval(code, interval)
```

  - `code` is a string of JavaScript code (in quotes)
  - `interval` is time in milliseconds

```
clearInterval(interval_Id)
```

  - `interval_Id` is the value returned by **`setInterval`**

❐ To delay the execution of code, use `setTimeout` and `clearTimeout` (same parameters)

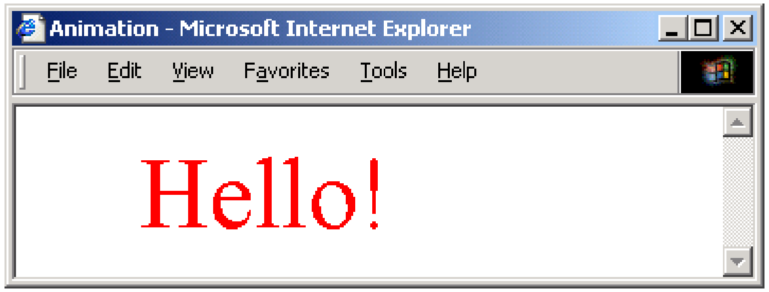
❐ These four methods are all window methods, and provide a solid backbone for most animation techniques in DHTML.

Example 3 Animation

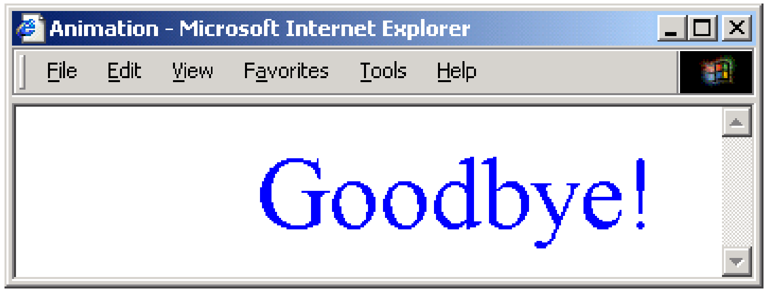
```
<style>
   #para1 {position: absolute;
     color : red;
     font-size : 40pt}
 </style>
 <script type="text/javascript">
   var x = 20, xshift = 1;
   function move(elementId) {
     x = x + xshift;
     elementId.style.left = x;
     if (x > 150) {
       xshift = -1;
       elementId.innerHTML = "Goodbye!";
       elementId.style.color = "blue";
     }
     if (x < 20 ) {
       xshift = 1;
       elementId.innerHTML = "Hello!";
       elementId.style.color = "red";
     }
   }
 </script>
```

Animation

```
<body onLoad="intervalId = setInterval('move(para1)',10)"
      onClick="clearInterval(intervalId)">
 <p id="para1" >
    Hello!
 </p>
</body>
```


Animation - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Hello!


Animation - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Goodbye!

Roll-over Graphics

- Perhaps the most visible (and popular) use of JavaScript involves roll-over graphics
- A rollover is an image that changes its appearance when the mouse moves over it, and returns to its original state when the mouse moves away
- We can access the filename for the **first** image on a page like this:

```
Var imageFileName=document.images[0].src;
```

- We can therefore change the image like this:

```
document.images[0].src = newFileName;
```

- We can use the `onMouseOver` and `onMouseOut` event handlers to update the image file name:

```
<html>
<head></head>
<body>
<img src="cat_1.jpg"
onMouseOver="document.images[0].src='cat_2.jpg' "
onMouseOut="document.images[0].src='cat_1.jpg' ">
</body>
</html>
```

Rollovers example

```
<html>
<head></head>
<body>
<a href="catwebpage.html”>
<img src="cat_1.jpg"
onMouseOver="document.images[0].src='cat_2.jpg' "
onMouseOut="document.images[0].src='cat_1.jpg' ">
</a>
</body>
</html>
```

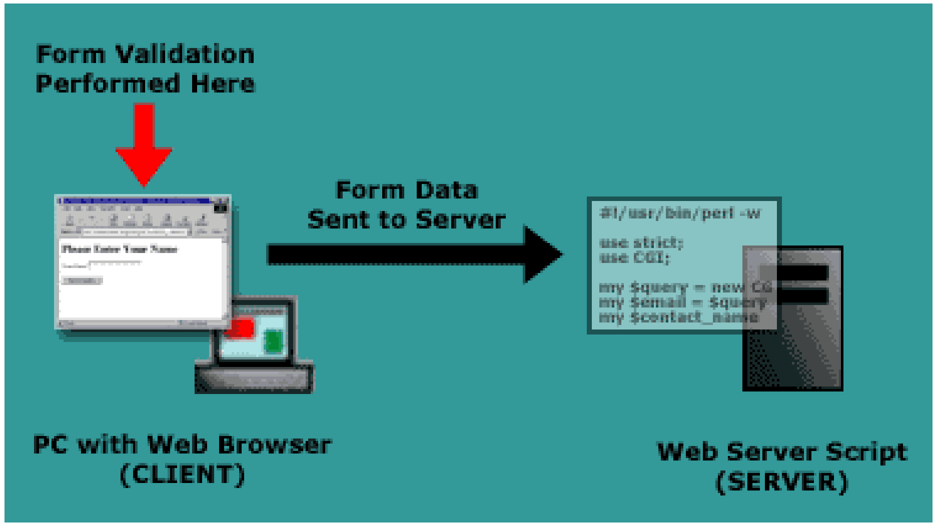
Google Maps API

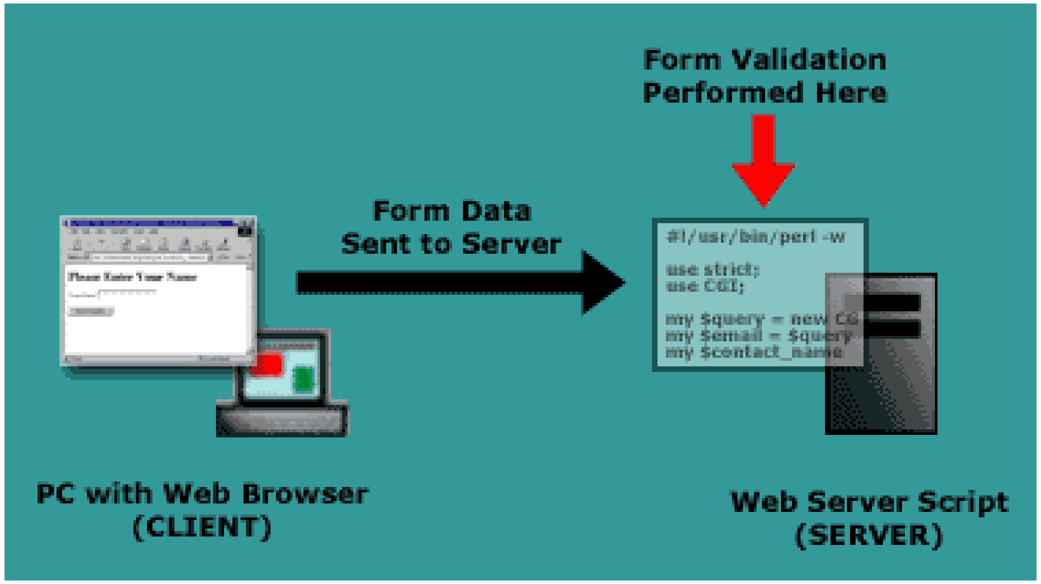
- A web mapping service application and technology provided by Google
- Bringing the power of Google Maps to your website or application
- Google map API is available to you in JavaScript
- Main reference:
  - https://developers.google.com/maps/documentation/javascript/reference
- Samples:
  - https://developers.google.com/maps/documentation/javascript/examples/
- Other APIs are available to be used, some of which are free.

Form Validation in JavaScript

- The right way to use JavaScript is as an optional extra to improve user-friendliness
- All form submissions are validated on the server
- If the reader has JavaScript available, an initial validation is done on the client, to check for simple errors such as missing information or non-numeric characters in a numeric field.
- The reader gets faster feedback than waiting for a response from the server.
- The essence is thus that JavaScript validation is always optional.
- Since the validation must be on the server anyway, it is not necessary to insist that the user must have JavaScript.

Form Validation





Do’s and Don’ts

- There are two ways of implementing validation, of which the better one is hardly ever used!

1. Acceptable: validate the entered data when the submit button is pressed.
2. Better in most cases, especially with large forms: validate on a field-by-field basis, and revalidate the whole form when the submit button is pressed. This also makes it easy to alter the user in cases where entered data is perhaps, but not certainly, incorrect – for example unusual characters in an email address, or a birth year of 1891.

DON’T

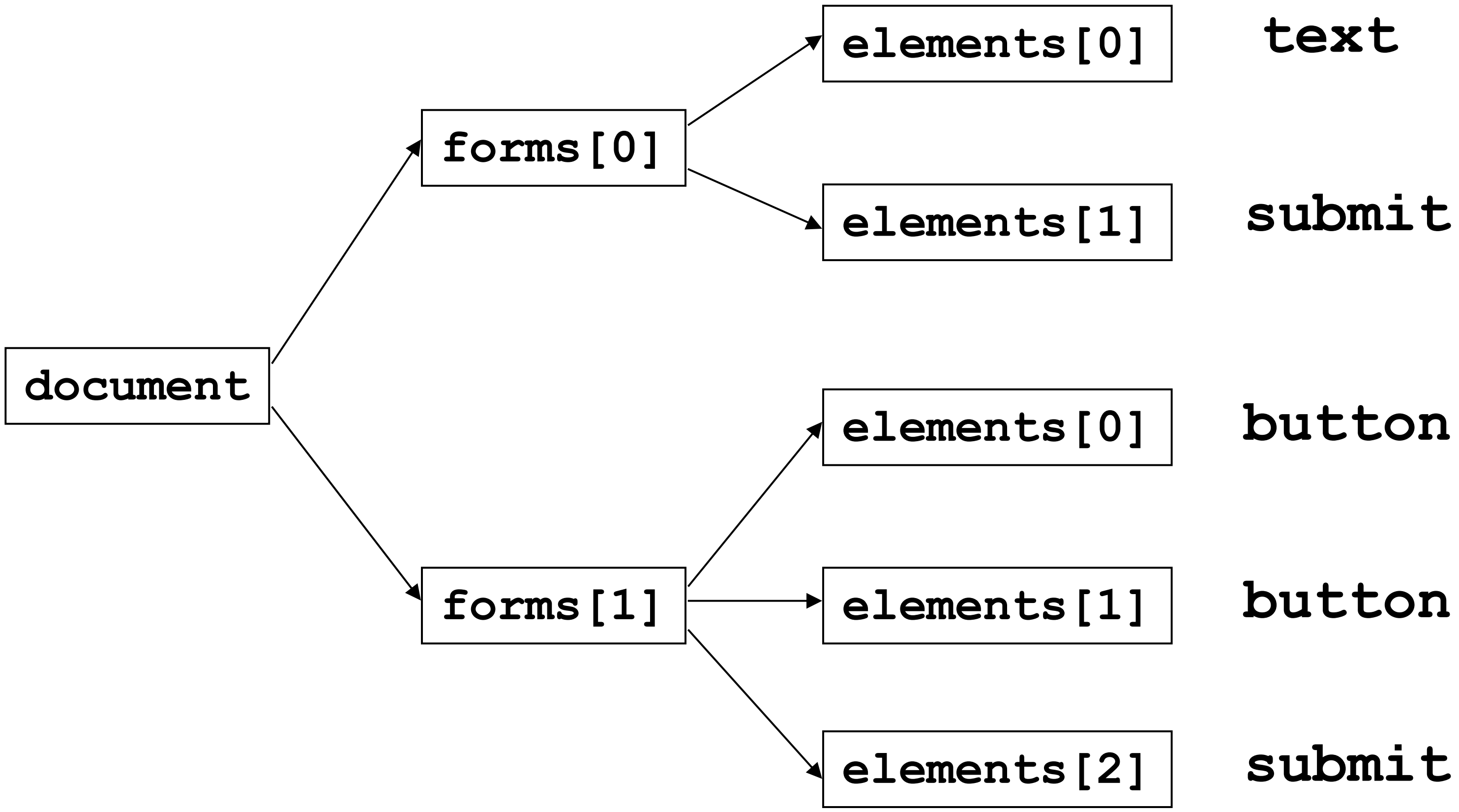
- Have the validation in JavaScript only, and not on the server.
- Simply assume the user has JavaScript

`form` objects

- Any data entered into a HTML form is contained in a `form` object
- An array `forms[]` contains all the form objects on a page
- Data from the first form on the page is contained in the first array element, e.g. `forms[0]`
- Data from the second form on the page is contained in the second element `forms[1]`
- A `forms` object contains another array, called `elements[]`
- An element refers to an input type, e.g. text, password, checkbox, radio,…
- Input types are organised inside `elements[]` in the order they appear in the specific form

`form` and `element` objects

- For example, a page has two forms. The first form has a text input and a submit button. The second form has two standard buttons and a submit button.



Where Does This Lead Us?

- We now have access to data entered into a form.
- JavaScript is typically used to do initial checks on form data before sending it to a server, e.g.:
  - Ensures all fields are filled in
  - Ensures that valid email addresses/URLs are entered

- The JavaScript program taking the form input can also be standalone – it doesn't have to send the data onto a server (see Simple Calculator Example).
- NB: Server side code may be written in PHP or Perl (and others)

Form Validation in JavaScript

Process of form validation

1. User fills in details and clicks on submit
2. A JavaScript function is called to check the form
   - ❒ Have all necessary fields been filled in?
   - ❒ Do fields have correct values?
3. Result of function is returned through onclick event handler
   - ❒ If return value is true – form is submitted
   - ❒ If return value is false – form is not submitted

Identifying by Names

- It is possible to access elements by name
- We can give names to specific HTML elements
- We can then access the properties of that element by referencing its name
- This is an alternative to referencing an element via its array location.

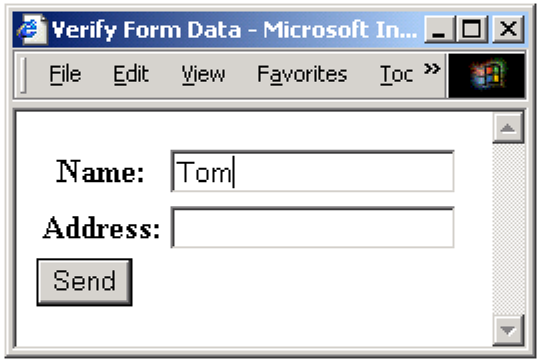
```
<head><script type="text/javascript">
 <!--
  function addNumbers() {
    var num1 = document.myForm.number_1.value;
    var num2 = document.myForm.number_2.value;
    var result = parseInt(num1) + parseInt(num2);
    alert("Result: " + result);
  }
 //-->
 </script></head><body>
 <form name = "myForm" method="POST">
  Number 1 <input type="text" name="number_1"><br>
  Number 2 <input type="text" name="number_2"><br>
  <input type="button" value="Add" onClick="addNumbers()">
 </form></body>
```

Example: Verifying Form Input

- We can use JavaScript to check forms before sending data to a server
- In this example:
    - The `onSubmit` event handler is fired when the submit button is pressed
    - This calls function `verifyForm()`
    - This returns `false` if there is missing data and alerts the user
    - If it **does not return false** then the form data is sent to `processForm.php`

Example

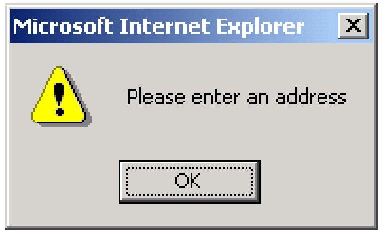
```
<head>
<script type="text/javascript">
 <!--
  function verifyForm() {
   if(myForm.username.value == "" ) {
     alert("Please enter a name");
     return false;}
   if(myForm.address.value == "" ) {
     alert("Please enter an address");
     return false;}
  }
 //-->
 </script>
</head>
<body>
 <form name="myForm" method="POST" action="processForm.php"
onSubmit="return verifyForm()">
  Name: <input type="text" name="username"><br>
  Address:<input type="text" name="address"><br>
  <input type="submit" value="Send">
 </form></body>
```



Name: Tom

Address:

Send



Please enter an address

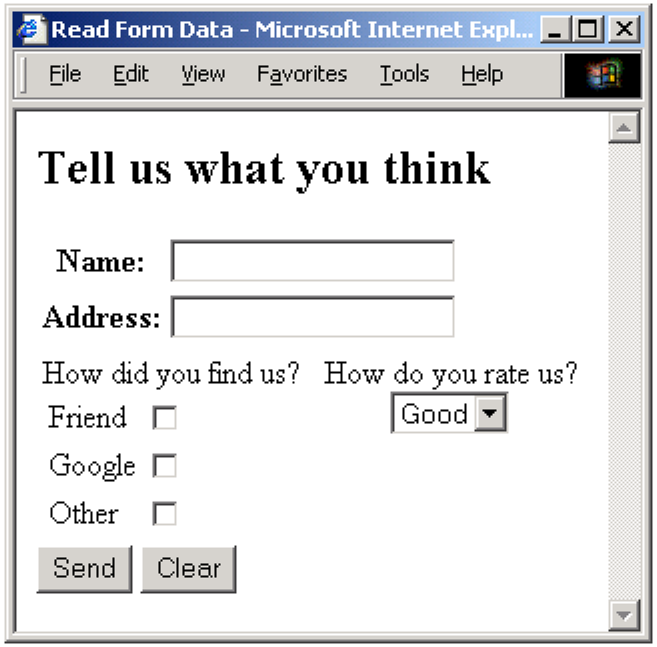
OK

A More Advanced Form

- Lets look at a more advanced form
  - Multiple input types
  - Passes form object to function **`readForm`** as a parameter
- The script reads the form data and writes it to a ***new window.*** We name this window **`win`**
- Also note:
  - **`checked`** is used to determine whether or not a check box is selected or not
  - The **`select`** element called **`rating`** contains an array of **`options[]`**
  - **`selectedIndex`** is used to record the option selected

The Form

```
<form name="feedbackForm"
  onSubmit="readForm(this)">
<h2>Tell us what you think</h2>
Name: <input type="text" name="username"><br>
Address: <input type="text" name="address" size="35"><br>
<table><tr><td>
How did you find this web site?:<br>
Friend<input type="checkbox" name="how1" value="friend"><br>
Google<input type="checkbox" name="how2" value="google"><br>
Other <input type="checkbox" name="how3" value="other"><br>
</td><td>
How do you rate this site?
<select name="rating">
<option value="good">Good
<option value="bad">Bad
<option value="ugly">Ugly
</select>
</td></tr></table>
<h3>Thank you</h3>
<input type="submit" value="Send">
<input type="reset" value="Clear">
</form>
```


Read Form Data - Microsoft Internet Expl...
File Edit View Favorites Tools Help

Tell us what you think

Name:

Address:

How did you find us? How do you rate us?

Friend ☐

Google ☐

Other ☐

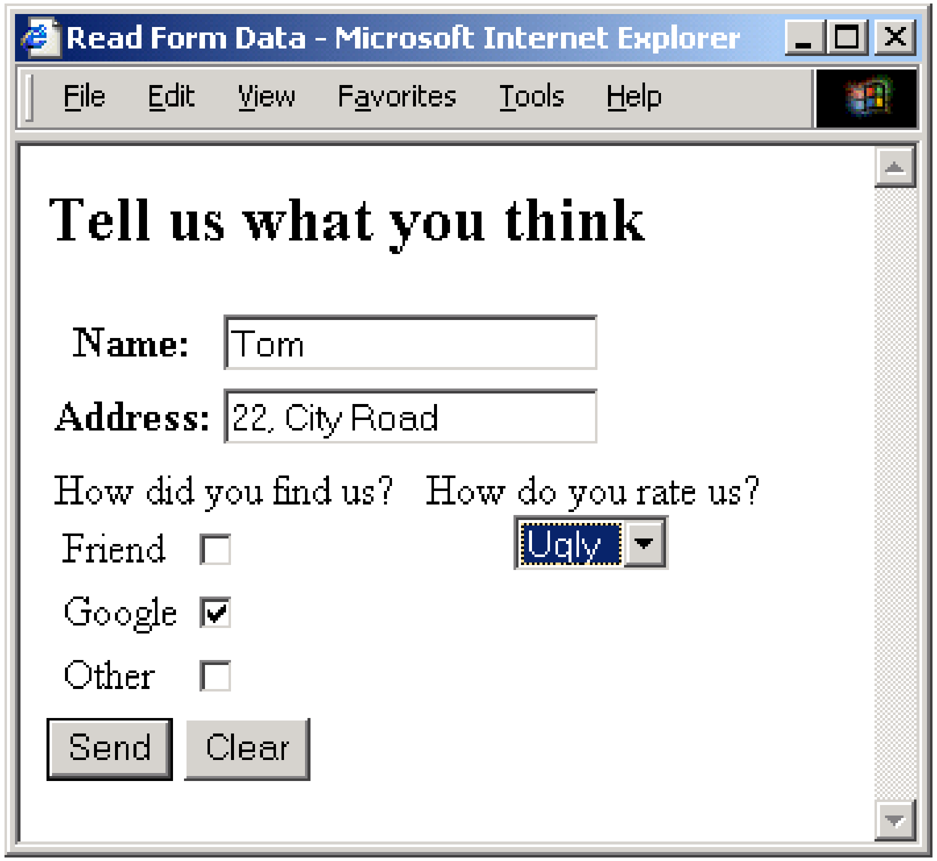
Good

Send Clear

The Script

```
function readForm(theForm)
{
  var i, optIdx, rating;
  win = window.open("","messageWin");
  win.document.writeln("<h2>The name is: "
                 +theForm.username.value+ "</h2>");
  win.document.writeln("<h2>The address is: "
+theForm.address.value+ "</h2>");
  win.document.writeln("<h2>You found us by: </h2>");
  if(theForm.how1.checked){
  win.document.writeln("<br>"+theForm.how1.value);
  }
  if(theForm.how2.checked){
  win.document.writeln("<br>"+theForm.how2.value);
  }
  if(theForm.how3.checked){
  win.document.writeln("<br>"+theForm.how3.value);
  }
  opIdx = theForm.rating.selectedIndex;
  rating = theForm.rating.options[opIdx].value;
  win.document.writeln("<h2>Rating: " +rating+ "</h2>");
}
```

Example Output


Read Form Data - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Tell us what you think

Name: Tom

Address: 22, City Road

How did you find us? How do you rate us?

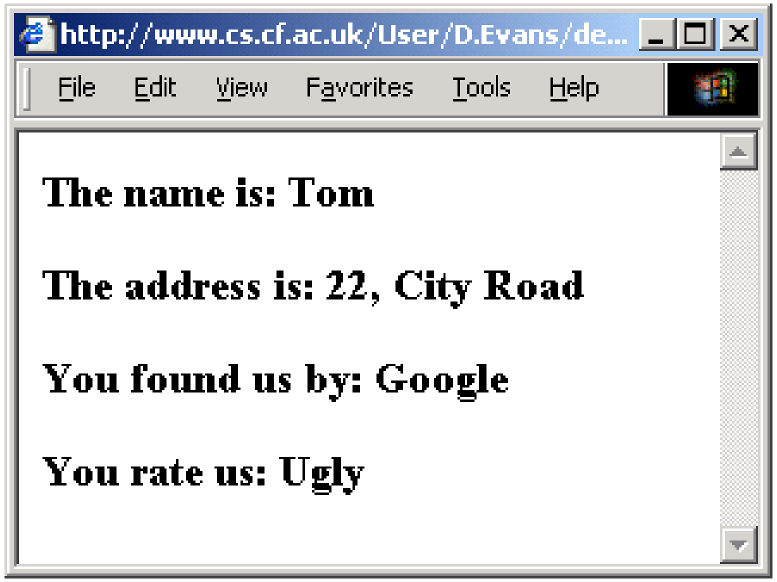
Friend ☐ Ugly

Google ✓

Other ☐

Send Clear

A message window created in response to user input


http://www.cs.cf.ac.uk/User/D.Evans/de...
File Edit View Favorites Tools Help

The name is: Tom

The address is: 22, City Road

You found us by: Google

You rate us: Ugly

Using Windows

- In the previous JavaScript validation examples, we used alert() to inform the user that an error had occurred.
- This might not always be suitable
- JavaScript lets you open, write to and close additional windows.
- Much more user friendly.
- To open a new window:

  myRef=open(url, name, features)

  - myRef: a reference to the new window
  - url: url of contents of the window (use “” if you want the window to be blank
  - name: this can be used with the HTML TARGET attribute
  - features: a list of desired browser features
    - e.g. toolbar, location, status, scrollbars, resizable, width, height, top, left

```
var myWin=open(“help.html”, “help”, “scrollbars, width=400,
   height=200”);
```

Opening New Windows

- We can also use the **`open`** method to open pages in new windows – often used for advertising/pop-ups

```
<html><head>
<script type="text/javascript">
<!--
open("http://www.bbc.co.uk");
-->
</script>
</head>
<body>
<h1>This page loads another page</h1>
</body></html>
```

Using Windows

- Once you have opened a (blank) window, you can use the document.write() method to add content

myWin=open("", "Hello", "scrollbars");

myWin.document.write("<h1>HELLO!!!</h1><br>");

- Use the close() method to close a window.
- Helpful if the user is presented with a close button.

myWin.document.write("<form><input type=button value=‘Close Window’ onClick=‘self.close()’></form>“);

JavaScript Summary

The basics of JavaScript:

- Variable, Data Types, Flow Control, Loops
- Function, Event, Objects

- JavaScript permits you to:
  - Make pages more dynamic
  - Validate HTML form input
  - Manipulate objects
  - Combine with DOM to build DHTML pages

Javascript Libraries

- As with most other programming languages, Javascript allows usage of external libraries.
- Many exist, one of the most popular ones is jQuery
  - http://jquery.com/
  - There are many examples available on the web, and once again w3schools.com has a whole section about it
  - Check and then try out some examples

jQuery
write less, do more.

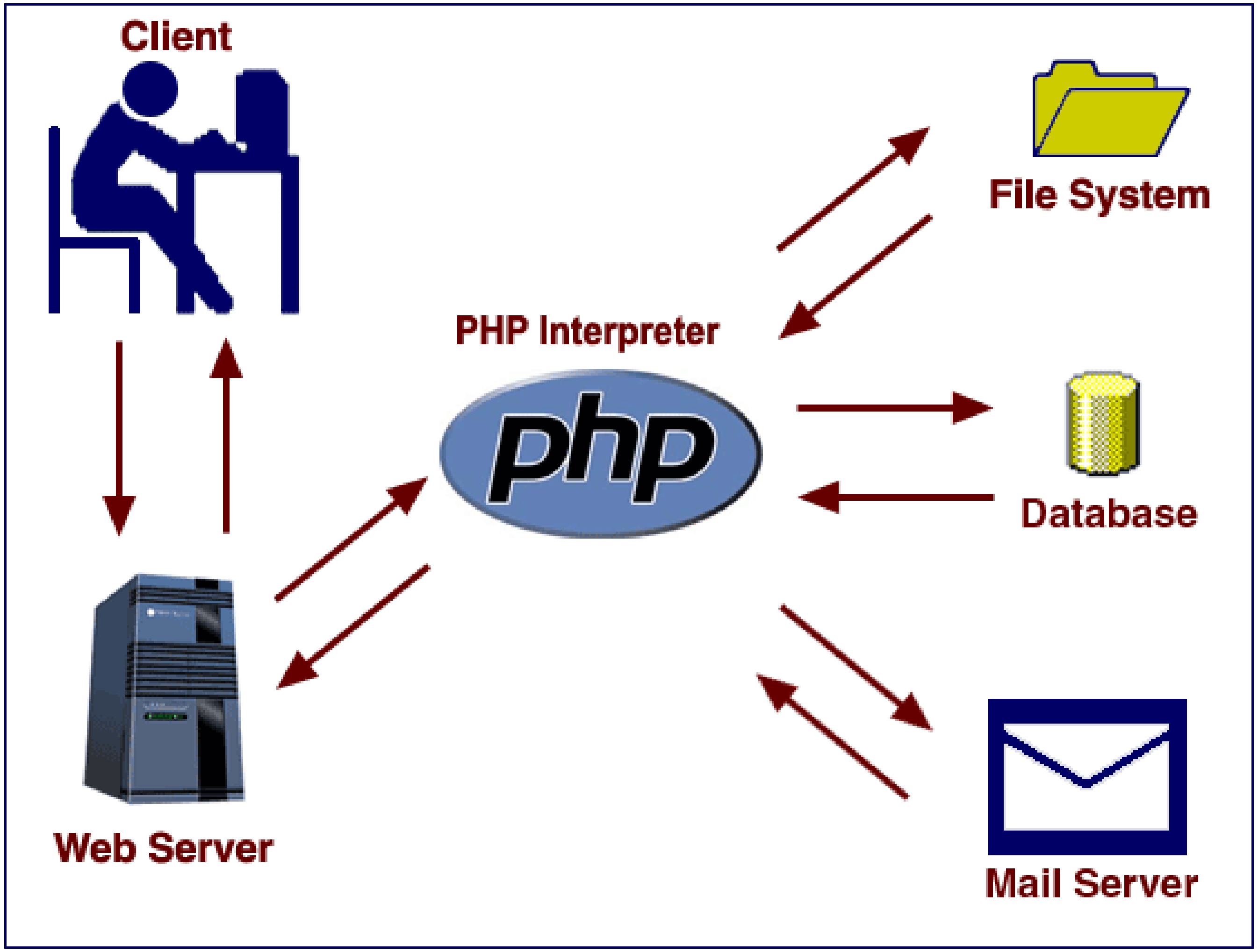
PHP Hypertext Preprocessor (PHP)

- Rasmus Lerdorf – 1994 developed to allow him to track visitors to his Web site
- PHP is an open-source product
- PHP is an acronym for *Personal Home Page*, or *PHP: Hypertext Preprocessor*
- PHP is used for form handling, file processing and database access

Overview of PHP

- PHP is a server-side scripting language whose scripts are embedded in HTML documents
  - PHP files may contain text, HTML tags and scripts
  - PHP files are returned to the browser as plain HTML
  - PHP files have a file extension of ".php", ".php3", or ".phtml"
- PHP syntax is similar to that of JavaScript
  - PHP is dynamically typed and purely interpreted

How PHP works



General Characteristics

- PHP code can be specified in a HTML document internally or externally:
  - Internally: <?php ...?>
    - A PHP scripting block can be placed anywhere in the document.
    - On servers with shorthand support enabled, you can start a scripting block with <? and end with ?>.
  - Externally: <?php include ("myScript.php"); ?>
- A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.
- *Comments* - three different kinds (like Java)

```
// ...comments
# ... comments
/* ...comments */
```

- Compound statements are formed with braces
- Each code line in PHP must end with a semicolon.
- Compound statements cannot be blocks

Variables

- Every variable name begin with a $ and then followed by a letter or an underscore "_"
  - $variable_name = Value;
- Variable names are case-sensitive and cannot start with a digit.
- Variable Types
  - Integer, Double, Boolean (TRUE or FASLE), String and (Array and Object)
- In PHP a variable does not need to be declared before being set.
- PHP automatically converts the variable to the correct data type, depending on how they are set.
- String variables are used for values that contains character strings.
  - Text enclosed in either single (‘ ‘) or double (“ “) quotes.
  - Using double quotes, the variable $hello is evaluated within the quotes

```
<?php
$hello = "Hello World!";
$a_number = 4; $anotherNumber = 8.5;
echo "My answer is $hello<br>";
//result is: My answer is Hello World!
?>
```

Arrays

- Each element in the array has its own ID so that it can be easily accessed.
- There are three different kind of arrays:
  - **Numeric array** - An array with a numeric ID key
    - Array operations are exactly the same as those found in JavaScript, e.g. `$array[0]="Smith"; $array[1] = "Brown"; $array[2]="Red";`
  - **Associative array** - An array where each ID key is associated with a value
    - Each item is indexed with a key value and the key must be a unique string, e.g. `$array["friend"]="Smith"; $array["son"] = "Peter";` or `$array=array("friend" => "Smith", "son"=>"Peter");`
  - **Multidimensional array** - An array containing one or more arrays
    - e.g. `$array[0][0]="Fido"; $array[0][]="Brown"; $array[0][2]="quite"; $array[1][0]="Rover"; $array[1][]="White"; $array[1][]="noisy";`
  - Example

Array Functions

- PHP supplies a rich set of build-in functions which can be used to operate on arrays.
  - array_keys(array) returns an array containing all of keys from the associated array
  - array_merge(arr1, arr2, …) merges all of the arrays which are given as parameters
  - array_reverse(array) returns a new array which contains all the elements in reverse order
  - asort(array) sorts an associated array and in doing so preserves the association between each key and its value.
  - each(array) returns the next key:value pair from an array
  - sizeof(array) returns the number of elements in the array
  - sort(array) sorts the elements in the array into ascending order and returns the sorted list

Output

- Output from a PHP script is HTML that is sent to the browser
- HTML is sent to the browser through standard output
- There are three ways to produce output: echo, print, and printf
  - echo and print take a string, but will coerce other values to strings
  - echo "whatever"; # Only one parameter
  - echo("first <br />", $sum) # More than one
  - print "Welcome to my site!"; # Only one
  - <?php $str = "Hello"; $number = 213; printf("%s world. Day number %u",$str,$number); ?>
- PHP code is placed in the body of an HTML document

Program Control

- Logical operations (text and numerical)
- Provided the same set of branching and looping constructs as many other language (e.g. Java)
  - if …[elseif …] else
  - while
  - for
  - foreach
    - foreach ($array as $value)
    - foreach ($array as $key => $value)
  - break
  - switch
    - switch (expression) { case label; statement; … [default: statement;] }
    - Selects between a number of choices depending upon the value of the expression.
    - The choices are identified by case statements, each has a label.

User-defined Functions

***Syntactic form*:**

```
function function_name(formal_parameters) {
     …
   }
```

General Characteristics

- A function is a block of code that can be executed whenever we need it.
- Function overloading is not supported
  - If you try to redefine a function, it is an error
  - Functions can have a variable number of parameters
- Function names are **NOT** case sensitive
- The return function is used to return a value; If there is no return, there is no returned value

PHP - Files

- Deal with any files on the server, on the Internet, using either http or ftp
- There are 3 steps to using data in a file
  - *Open the file. If the file doesn't already exist create it or catch the error gracefully.*
  - *Write/Read data from the file.*
  - *Close the file.*
- A file has a file pointer (where to read or write)

  $fptr = fopen(filename, use_indicator)

  Use indicators:

r	read only, from the beginning
r+	read and write, from the beginning
w	write only, from the beginning (also creates the file, if necessary)
w+	read and write, from the beginning (also creates the file, if necessary)
a	write only, at the end, if it exists (creates the file, if necessary)
a+	read and write, read at the beginning, write at the end

- Use file_exists(filename) to determine whether file exists
- Use fclose(file_var) to close a file

```
$ourFileName = "testFile.txt";
$ourFileHandle = fopen($ourFileName, 'w') or die("can't open file");
fclose($ourFileHandle);
```

Reading Files

- Read all or part of the file into a string variable

  $str = fread(file_var, #bytes)

  To read the whole file, use filesize(file_name)
- Read the lines of the file into an array

  $file_lines = file(file_name)

  Need not open or close the file
- Read one line from the file

  $line = fgets(file_var, #bytes)

  Reads characters until eoln, eof, or #bytes characters have been read
- Read one character at a time

  $ch = fgetc(file_var)

  - Control reading lines or characters with eof detection using feof (TRUE for eof; FALSE otherwise)

```
while(!feof($file_var)) {
    $ch = fgetc($file_var);
   }
```

Writing and Closing files

- Writing to a file in PHP is easy. You can either use the function:
  - **fwrite()** …file write
  - **fputs()** …file put sting (an alias to fwrite)

```
$fp = fopen("orders.txt", "a");
fwrite($fp, "adding something to the file");
```

- All that is left is to tidy everything up by closing the file

```
fclose($fp);
```

Viewing Client/Server Environment Variables

- Knowledge of a client’s execution environment is useful to system administrators.
- Environment variables contain information about a script’s environment, such as the client’s Web browser, the HTTP connection.
- Global Arrays

• $_SERVER	data about the currently running server
• $_ENV	data about the client’s environment
• $_GET	data sent to the server by the get method
• $_POST	data sent to the server by the post method
• $GLOBALS	array containing all global variables

- Note: *by default many of the latest installs (and associated php.ini’s) of PHP will have this option set to ‘GPCS’, which stands for Get, Post, Cookie and Built-in variables respectively*.

Form Handling

- Any form element in an HTML page will **automatically** be available to your PHP scripts.
- It does not matter whether GET or POST method is used to transmit the form data
- PHP builds an array of the form values
  - $_GET for the GET method: an array of variable names and values sent by the HTTP GET method.
  - $_POST for the POST method: an array of variable names and values sent by the HTTP POST method.
- Examples
  - Gathering user input
  - Verifying a user name and password

Time inside PHP

- **Time()** -- gives you the current UNIX timestamp
- Returns the current time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).
- You find yourself using this all the time once you have a complicated site. Common examples are:
  - Keeping track of login times
  - Time how long processes are taking
- Time is very useful but not very pretty – who wants to know the number of seconds since the UNIX epoch !?
- **Date()** outputs the current time into whatever format you specify:

Print date(“jS F Y”);	25th April 2008
Print date("F j, Y, g:i a");	April 25, 2008, 9:46 pm
Print date("H:i:s");	09:46:17
Print date("D M j G:i:s T") ;	Fri Apr 25 9:46:17 GMT

Date Format for date month year and time

a	'am' or 'pm'
A	'AM' or 'PM'
d	day of the month, 2 digits with leading zeros; i.e. '01' to '31'
D	day of the week, textual, 3 letters; i.e. 'Fri'
F	month, textual, long; i.e. 'January'
g	hour, 12-hour format without leading zeros; i.e. '1' to '12'
G	hour, 24-hour format without leading zeros; i.e. '0' to '23'
h	hour, 12-hour format; i.e. '01' to '12'
H	hour, 24-hour format; i.e. '00' to '23'
i	minutes; i.e. '00' to '59'
I	'1' if Daylight Savings Time, '0' otherwise.
j	day of the month without leading zeros; i.e. '1' to '31'
l	day of the week, textual, long; i.e. 'Friday'
L	boolean for whether it is a leap year; i.e. '0' or '1'
m	month; i.e. '01' to '12'
M	month, textual, 3 letters; i.e. 'Jan'

n	month without leading zeros; i.e. '1' to '12'
s	seconds; i.e. '00' to '59'
S	English ordinal suffix, textual, 2 characters; i.e. 'th', 'nd'
t	number of days in the given month; i.e. '28' to '31'
T	Timezone setting of this machine; i.e. ‘GMT'
U	seconds since the epoch
w	day of the week, numeric, i.e. '0' (Sunday) to '6' (Saturday)
Y	year, 4 digits; i.e. '1999'
y	year, 2 digits; i.e. '99'
z	day of the year; i.e. '0' to '365'

PHP documentation and libraries

- PHP is one of the most well documented languages on the web
- Its main reference is at:
  - http://www.php.net/docs.php

- Additionally, many libraries exist that can be used for a number of frequently implemented features (e.g. logging).