Web Engineering

Dr. Saleem Ahmed
DUET

Technical Skills

- HTML
- CSS
- JavaScript
- Search Engine Optimization (SEO)
- Web Design Tools
  - HTML Editors (such as Brackets, Komodo Edit or Dreamweaver) and graphics applications (such as GIMP or Photoshop).
- Content Management Systems (CMS)
  - Most famous CMS -> WordPress, Drupal, Joomla (HW-1, Comparison between three) no copy paste…I will check
- Designers: Graphic design, multimedia, art
- Developers: programming, databases, scripting languages such as PHP

Grading (tentative)

- Mid term 20%
- Quizes, assignments and project 20%
- Final exam 60%

syllabus

Introduction to Internet, HTML and Web Designing: Internet, Internet Protocols, Internet Services, Common internet applications, Basic HTML Concepts, Basic Structure of HTML document, Elements and tags, attributes, lists, Forms, Tables, Arrays, links, Designing CSS, CGI

Scripting Languages: JavaScript programming language: Syntax of various basic operations: conditional statements, Loops, Arrays, Objects, Events, Alerts, Prompts and Confirms. DOM,

PHP programming language: Operators, Functions, Control structures, Arrays, Database (MySQL) access with PHP

***Introduction to ASP**: ASP, Role of ASP, ASP Objects.

Books:

- H. M. Deitel, P. J. Deitel, Internet and World Wide Web How to Program, 4th and 5th edition

- Guide to HTML, Javascript and PHP by David R Brooks

- Web Standards Programmer's Reference: HTML,CSS, Javascript, Perl, Python and PHP by Steven M. Schafer

Internet, Packets and Routing

At the sender, data is **broken into packets** and sent to the nearest node (**router**)

At each router, it sends the packet to another router that is closer to the final destination

At the receiver, packets are **reassembled** to get the original data

A simple analogy: mailing system

Internet, Packets and Routing

Internet is a network of computer networks

Data is transmitted by packet switching using the standard **Internet Protocol** (**IP**)

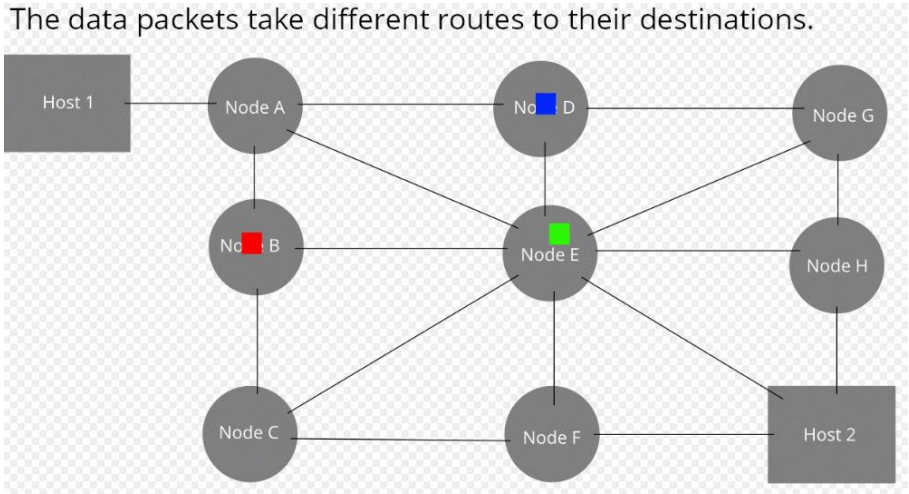
Packet – a unit of information carriage (longer messages broken in packets)

- Contains header and relevant info. for routing etc.

Packet switching – process of moving packets from one node (computer device) to another

Packet switching is a digital networking communications method that groups all transmitted data into suitably sized blocks, called *packets*, which are transmitted via a medium that may be shared by multiple simultaneous communication sessions

-



Packet switching

Advantages

Links can be shared

Suitable for computer generated traffic

- Virtual Circuit

Circuit switching

Dedicated communication path required between nodes

Path follows a fixed sequence of intermediate links

Three steps

Connection establishment

Data transfer

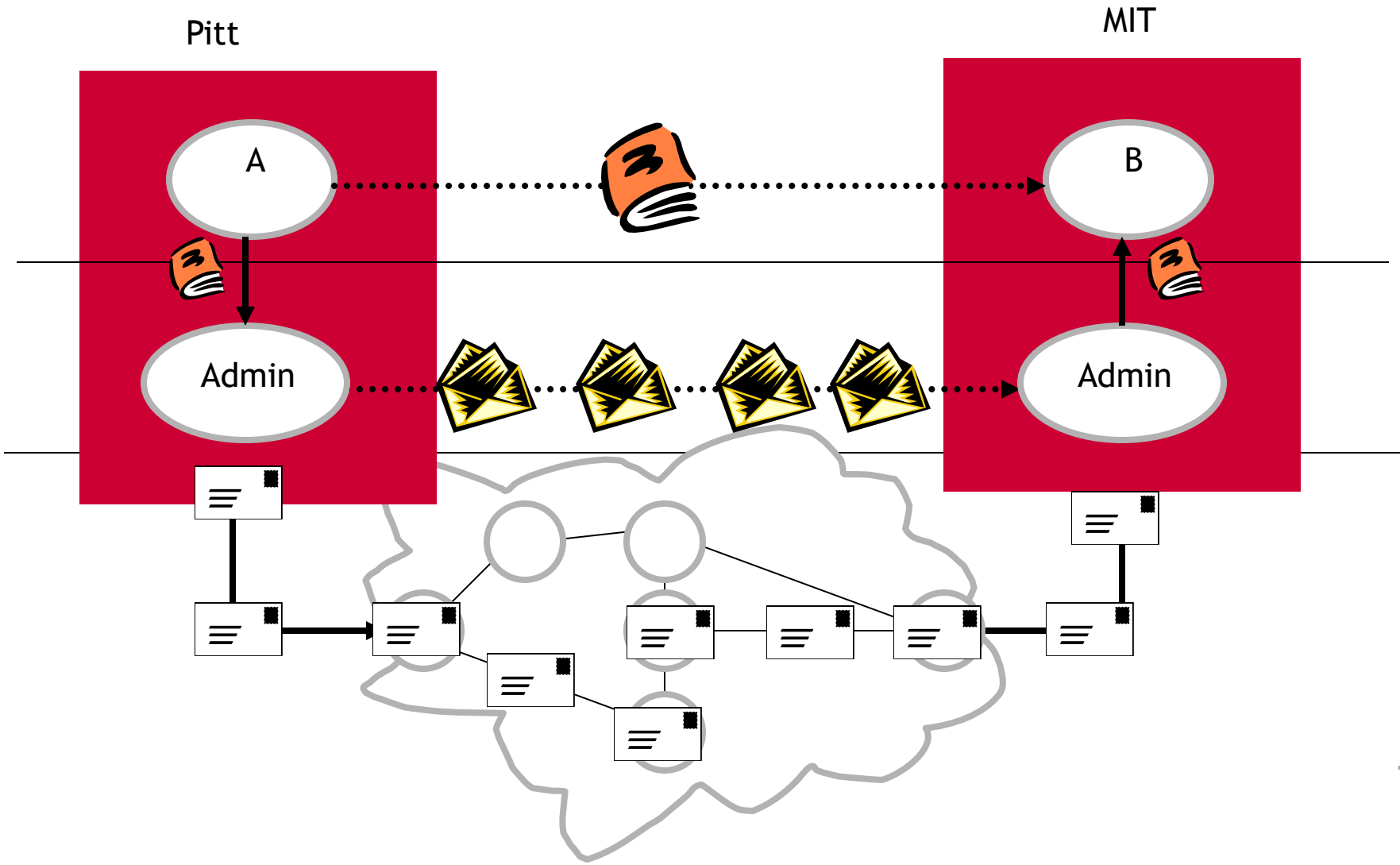
Connection termination

Drawbacks

Dedicated channel -> for voice is suitable but computer data not suitable because some times traffic is high.

Initial delay - > connection establishment

Mailing System



Layered network architecture

Open systems interconnection (OSI) model

Seven layer model

- **Purpose – communication functions are partitioned into set of layers – subtasks**
- **Change in one layer should not require changes in other layers**

Layered network architecture

-Physical layer

- Transmit raw bit stream over a physical medium

-Datalink layer

- Reliable transfer of frames over a point to point link (error control)

-Network layer

- Establishing, maintaining and terminating connections
- Point to point

-Transport layer

- Concerned with end to end reliable data transfer with error recovery

-Session layer

- Manages sessions

-Presentation layer

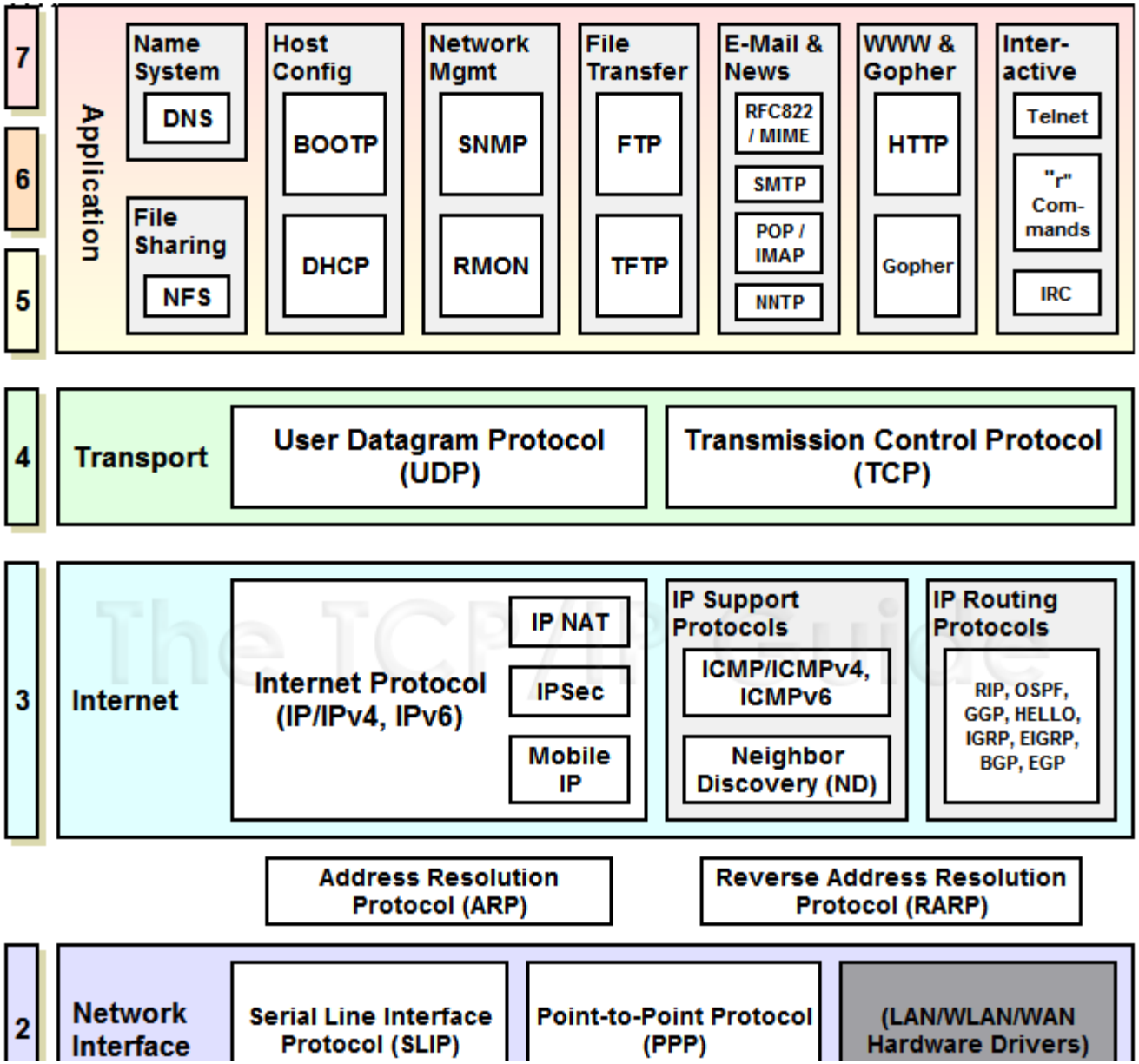
- Provides data independence – encryption

-Application

Layered network architecture

OSI Model				
Layer		Protocol data unit (PDU)	Function[3]	Examples
Host layers	7. Application	Data	High-level APIs, including resource sharing, remote file access, directory services and virtual terminals	TLS, FTP, HTTP, HTTPS, SMTP, SSH, Telnet
	6. Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption	CSS, GIF, HTML, XML, JSON
	5. Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes	RPC, SCP, NFS,PAP
	4. Transport	Segment (TCP) / Datagram (UDP)	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing	NETBEUI, TCP, UDP
Media layers	3. Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control	AppleTalk, ICMP, IPsec, IPv4, IPv6
	2. Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer	IEEE 802.2, L2TP, LLDP, MAC, PPP
	1. Physical	Bit	Transmission and reception of raw bit streams over a physical medium	DOCSIS, DSL, Ethernet physical layer, ISDN, USB

Layered network architecture



TCP/IP and Domain Names

Basic task of IP – moving packets as quickly as possible from one router to another

Yet, it doesn't check whether packets are delivered successfully, thus need **TCP**

TCP (**Transmission Control Protocol**) – disassemble/reassemble packets, error checking, ACK packets

TCP/IP and Domain Names

We need some sort of address in order to identify different nodes, as if every house has a mailing address in order to receive mail from others

The one used by Internet Protocol is called **IP address**

Every host on the Internet has a unique IP address, made up of four numbers. E.g.. 192.56.215.131, each number is between 0 and 255

The Internet Assigned Numbers Authority (IANA) manages the IP address space allocations globally and delegates five regional Internet registries (RIRs) to allocate IP address blocks to local Internet registries (Internet service providers) and other entities.

TCP/IP and Domain Names

The numbers in an IP address is hard to remember, while names are easier

Domain Name System – a mapping between the human-readable name (domain name) of a host and its IP address

A **domain name** consists of two or more parts, e.g. *cs.pitt.edu*

The rightmost label conveys the top-level domain, e.g. *edu*

TCP/IP and Domain Names

Each label to the left specifies a subdomain, in our example, subdomain is *pitt* (University of Pittsburgh), and sub-subdomain is *cs* (computer science).

A top-level domain contains of multiple subdomains, each subdomain can contain multiple sub-subdomain, so on.

The database contains the mapping between a domain name and an IP address is stored on a **DNS server.**

World Wide Web

The **World Wide Web** (commonly shortened to the **Web**) is a system of interlinked, hypertext documents accessed via the Internet.

It is created to share files/documents and overcome the barrier of different file formats

Hypertext refers to text on a computer that will lead the user to other, related information on demand.

World Wide Web

hypertext documents are created using a special kind of document formatting or “markup” language called **HyperText Markup Language** (**HTML**).

HTML is sent or received over the network using **HyperText Transfer Protocol** (**HTTP**).

A **browser** is a software program which interprets the HT documents and displays it on the user’s screen.

URLs and Client-Server Model

Each document/resource on the WWW needs to have an identifier in order to be accessed by others.

A **Uniform Resource Identifier** (**URI**), is a compact string of characters used to identify or name a resource.

A **Uniform Resource Locator** (**URL**) is a URI which provides means of obtaining the resource by describing its network “location”.

URLs and Client-Server Model

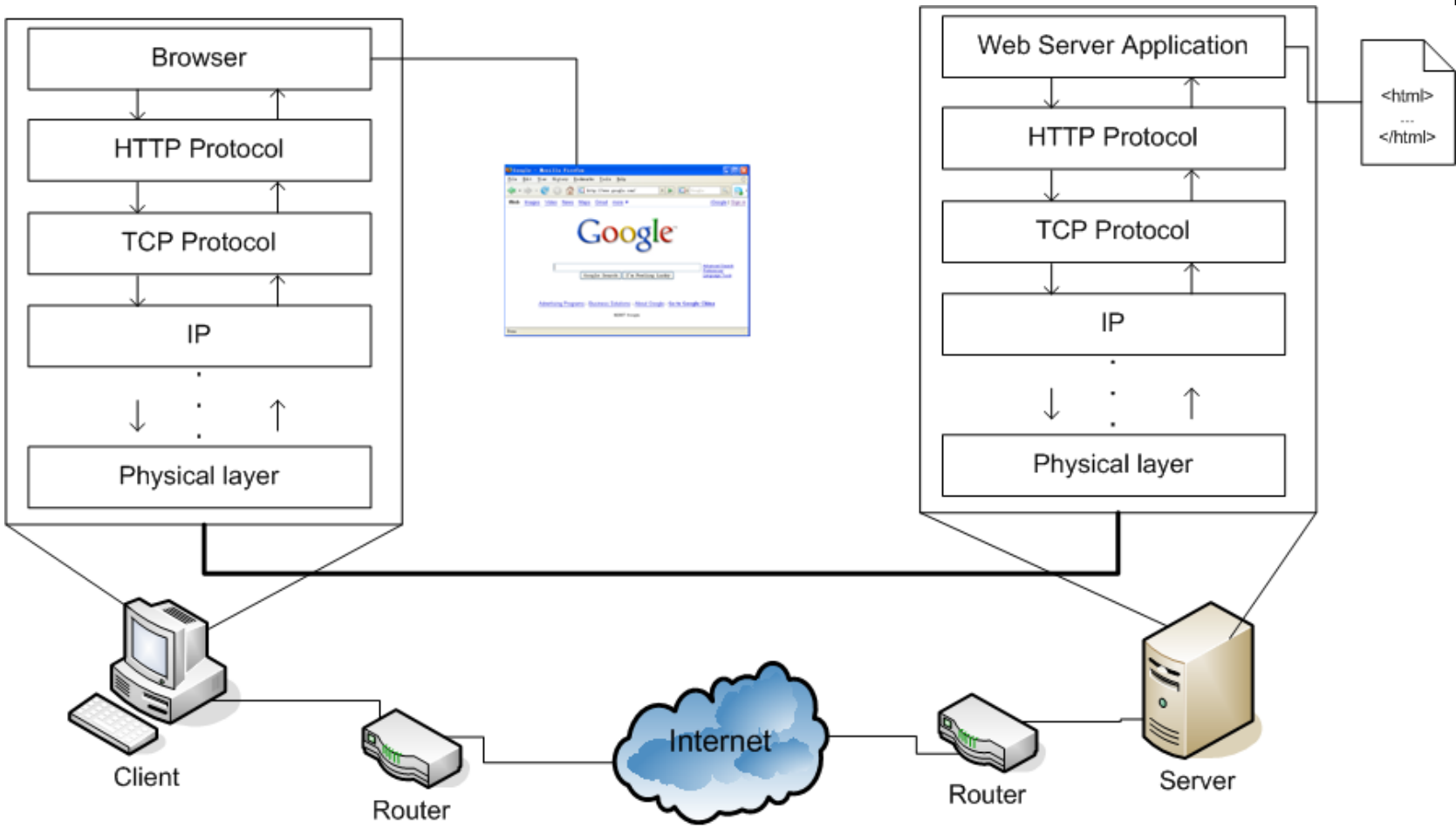
Two things are given by the URL

- Exact location of the document
- The method or protocol by which to retrieve and display the document

Example,
http://www.cs.pitt.edu/~mehmud/cs134/index.html

- http:// – specifies the protocol
- www.cs.pitt.edu – specifies the host name / domain name
- /~mehmud/cs134/index.html – specifies the path of the document on the host

Putting it All Together



Mini Project

CMS

Joomla - > well known CMS system being used in market

Joomla is an open source Content Management System (CMS), which is used to build websites and online applications. It is free and extendable which is separated into front-end templates and back-end templates (administrator). Joomla is developed using PHP, Object Oriented Programming, software design patterns and MySQL (used for storing the data). This tutorial will teach you the basics of Joomla using which you can create websites with ease.

Design a website of your choice

Prepare a presentation about

1) steps of development

2) complete website in run

Evaluation

Ease to use (how easily user can interact -> friendly environment)

Major grading -> Look of the website (colors, presentation, design)

*not a lab time work

Final Projects

- HR Model – payroll system, leaves, etc
- Student, Faculty Portal
- Research Journal website – I will provide example
- Please give suggestion – mine will be final approval

softwares

- Install package XAMP
- Which contains all the required applications
  - Apache server, PHP, Mysql database, Javascript

HTML Source Document

HTML source document

- A text-only document
- Consists of (1) actual text, and (2) tags

A **tag** is an html code that is enclosed in angel brackets <>; used to lay out the web page.

XHTML is a simple, more standardized version of HTML

XHTML/HTML can be created using a simple text editor like notepad

File extension must be **.html** or **.htm**

XHTML Tags/Elements

Tags are also called **elements**

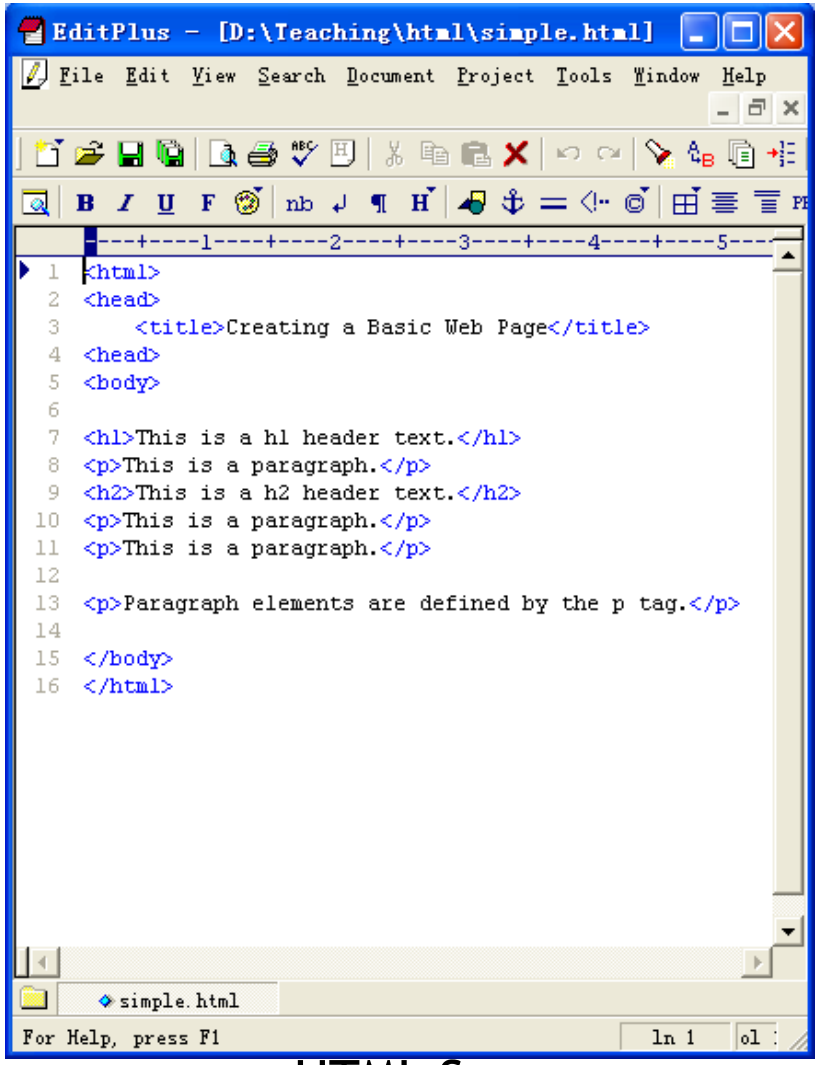
An **attribute** is a special code that can enhance or modify a tag. They are generally located in the starting tag after the tag name.

Basic syntax for xhtml tags and attributes

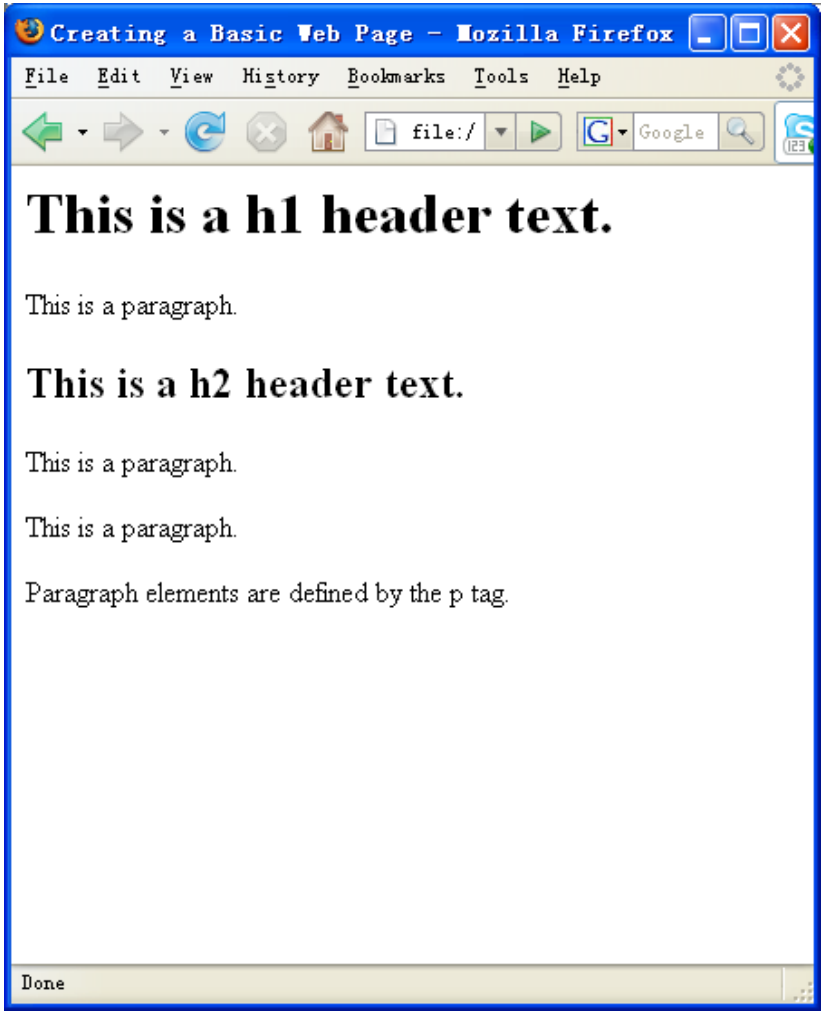
- **<tag attribute="value"> </tag>**

- All tags must be lower case
- all values of attributes need to surrounded by quotes

Sample HTML



HTML Source



Firefox display of the html source

Sample HTML

C:\xampp\htdocs\second.html - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

change.log | first.html | first1.html | first1.txt | CompoundInterest.php | hello.php | quadrat.htm | quadrat.php | first.html

```
1  <html>
2      <head>
3          <title>Hello, world!</title>
4      </head>
5          <body>
6              <p><em>Content</em></p>
7              <h1 align="center">First JavaScript</h1>
8              <font size='6'color='green'><center>DUET</center></font>
9              <h1 align="center" size='6'color='green'>web engg!!!</h1>
10
11         <script type="text/javascript">
12         <!--
13         document.write("Hello Javascript!");
14         //-->
15         document.write("<font size='5'color='red'><center>Hello, world!</center></font>");
16         document.write("<br><font size='7' color='blue'><center>It's a beautiful day!</center></font>");
17
18         document.write("<font size='5'color='green'><center>Hello, world!</font>");
19         document.write("<br><font size='7' color='gray'>It's a beautiful day!</center></font>");
20         document.write("<br><br><br><font size='7' color='black'><center>Hello, world!</font>");
21
22         </script>
23
24         </body>
25 </html>
```

Content

First JavaScript

DUET

web engg!!!

Hello Javascript!

Hello, world!

It's a beautiful day!

Hello, world!

It's a beautiful day!

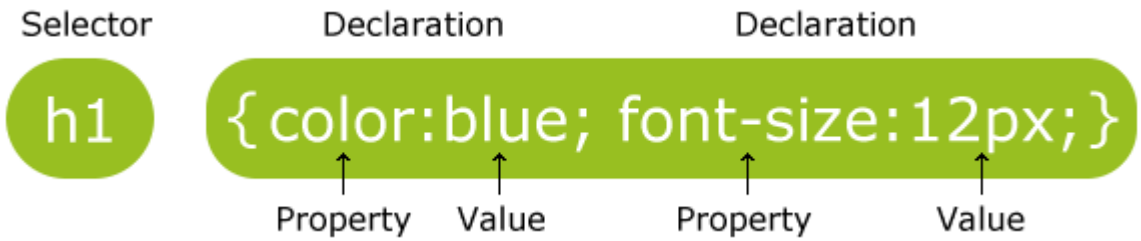
Hello, world!

CSS

CSS is a stylesheet language that describes the presentation of an HTML document. CSS **saves a lot of work**. It can control the layout of multiple web pages all at once

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style



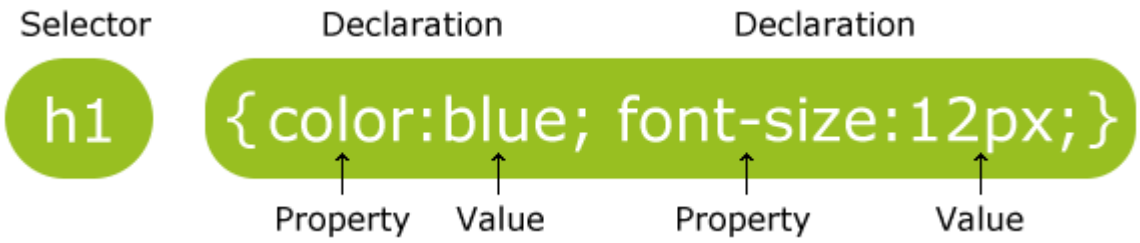
CSS

The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.



CSS , external style sheet

change.log | first.html | first1.html | first1.txt | CompoundInterest.php | hello.php | qua

```html
<html>
        <head>
        <title>design file!</title>
        <link rel="stylesheet" type="text/css" href="mystyle.css">
        </head>
<body>
<p>This is a paragraph...</p>
<p id="para1">Hello World!</p>

<h1>heading1</h1>
<h2>heading2</h2>

</body>
</html>
```

```css
body {
    background-color: lightblue;
}

h1 {
    color: navy;
    margin-left: 20px;
}

#para1 {
    text-align: center;
    color: red;
}

p {
    text-align: center;
    color: blue;
}
```

This is a paragraph...

Hello World!

heading1

heading2

CSS, internal style sheet

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

```
<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
```

CSS, inline style

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

```
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
```

CSS...

```
p {
    border: 1px solid black;
}
```

The id Attribute

```
<p id="p01">I am different</p>
```

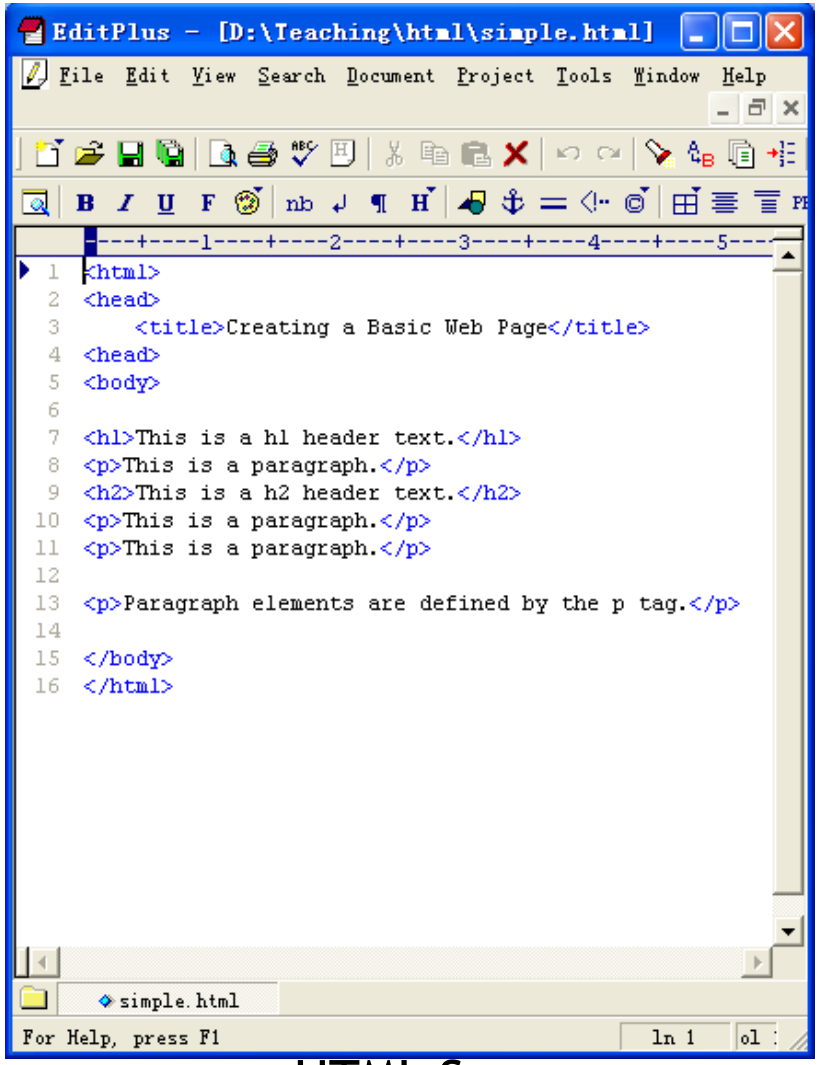
```
#p01 {
    color: blue;
}
```

The class Attribute

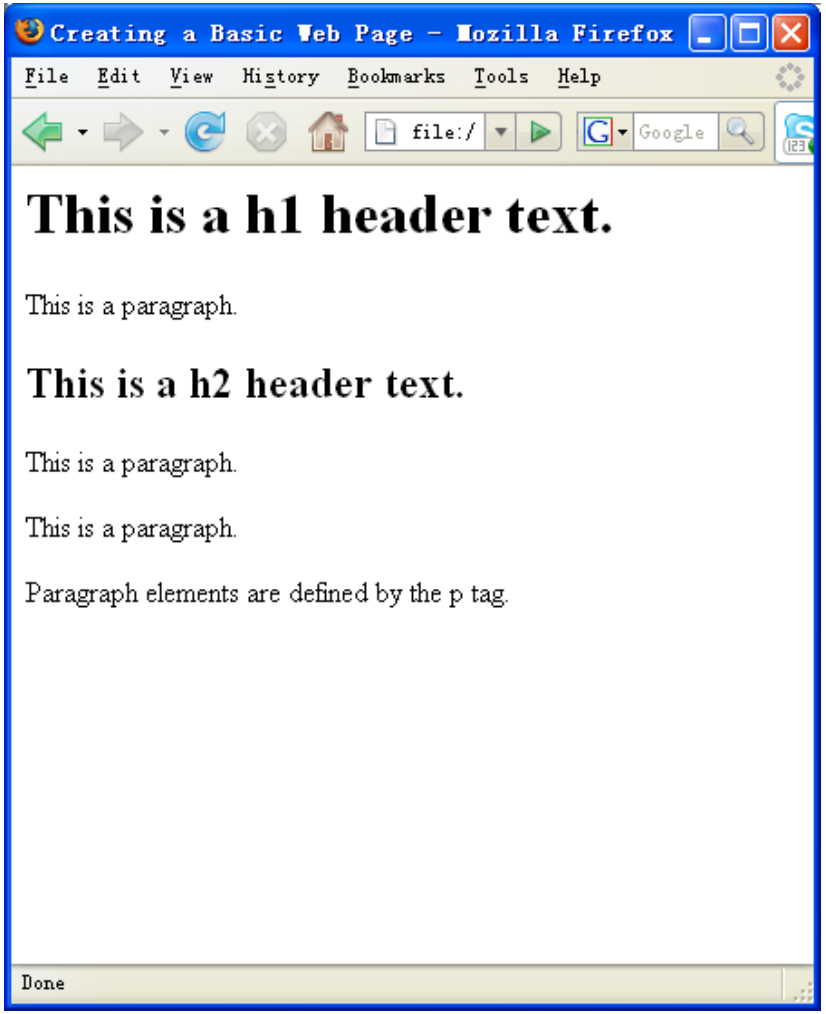
```
<p class="error">I am different</p>
```

```
p.error {
    color: red;
}
```

Sample HTML



HTML Source



Firefox display of the html source

JOOMLA

For windows

IIS 7

PHP 5.3.1

MySQL 5.1

FTP->filezilla-project.org – client

Demo.joomla.org

Fill form leftside – 30 days

Second box is your web address…road-runner it will become road-runner.cloudaccess.net

Activate account from email

Write URL of your side in browser

JOOMLA

www.joomla.org -> download joomla 3.-

Wampserver or XAMPP - download

Check wampserver for visual studio version

1 install server wampserver or xampp

2 extract joomla in folder - unzip - peazip

3- Skype - tools - connections..uncheck the use port - use port 80 and 443....

4- copy past joomla folder in c-wamp-www or xampp-htdocs

5- open localhost/phpmyadmin

Create a database

Create database duet

Check previledges

- **add user -> username - duet**
- **Host - localhost**
- **Password - duet**
- **Check all previledges**

- **On webbrowser write**
- **http://localhost/duet -> installation steps**
- **Error? Fatal error**: Cannot use Joomla\String\String as String because 'String' is a special class name in **C:\xampp\htdocs\duet\libraries\vendor\joomla\registry\src\Format\Json.php** on line **12**

I solved the problem by deleting Xampp 7,0 and installing xampp 5,6. Now it is working.

HTML

HTML links are defined with the **<a>** tag:

Click here to go to duet.

HTML images are defined with the **<img>** tag.

<img src="cbnu.jpg" alt="jbnu.ac.kr" width="104" height="142">

Or

<img src="pic_mountain.jpg" alt="Mountain View" style="width:304px;height:228px;">

The source file (**src**), alternative text (**alt**), and size (**width** and **height**) are provided as **attributes**:

HTML images as link.

```
<a href="http://www.duet.edu.pk">
  <img src="cbnu.jpg" alt="university" style="width:42px;height:42px;
border:0;">
</a>
```

.....

```
<img src="cbnu.jpg" alt="Smiley face" style="float:right;width:42px;
height:42px;">
```

The image will float to the right of the text.

HTML tables.

Tables are defined with the **<table>** tag.

Tables are divided into **table rows** with the **<tr>** tag.

Table rows are divided into **table data** with the **<td>** tag.

A table row can also be divided into **table headings** with the **<th>** tag.

HTML tables.

```
<table border="1" style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Jill	Smith	50
Eve	Jackson	94

HTML tables (CSS).

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
th, td {
    padding: 10px;
}
```

Jill	Smith	50
Eve	Jackson	94

HTML tables (heading).

```
<table style="width:50%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
     </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
     </tr>
</table>
```

Firstname	Lastname
Eve	Jackson

HTML lists.

An unordered list starts with the **<ul>** tag. Each list item starts with the **<li>** tag.

```
<ul style="list-style-type:disc">
  <li>Coffee</li>
  <li>Tea</li>
  </ul>
```

Style	Description
list-style-type:disc	The list items will be marked with bullets (default)
list-style-type:circle	The list items will be marked with circles
list-style-type:square	The list items will be marked with squares
list-style-type:none	The list items will not be marked

HTML lists.

An ordered list starts with the **<ol>** tag. Each list item starts with the **<li>** tag.

```
<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  </ol>
```

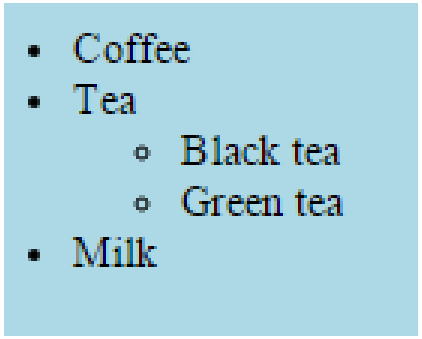
1. Coffee
2. Tea
3. Milk

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

HTML lists.

Nested HTML Lists

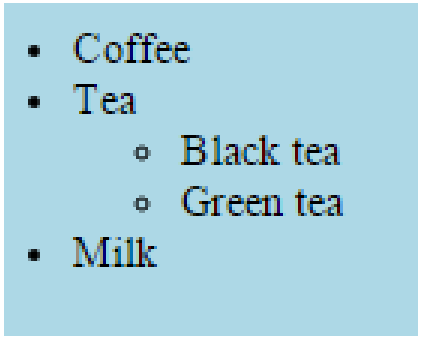
```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```



HTML lists.

Nested HTML Lists

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```



Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Examples of block-level elements:

<h1> - <h6>

<p>

<form>

Block-level Elements

The Element

The element is a **block-level element** that is often used as a container for other HTML elements.

When used together with CSS, the element can be used to style blocks of content:

```
<div style="background-color:black; color:white; padding:20px;">

<h2>London</h2>
<p>London is the capital city of England.</p>

</div>
```

Block-level Elements

The Element

The element is an **inline element** that is often used as a container for some text.

When used together with CSS, the element can be used to style parts of the text:

```
<h1>My <span style="color:red">Important</span> Heading</h1>
```

My Important Heading

HTML Forms

HTML forms are used to collect user input.

```
<form>
.
form elements
.
</form>
```

HTML forms contain **form elements**.

Form elements are different types of input elements, checkboxes, radio buttons, submit buttons, and more.

The <input> Element

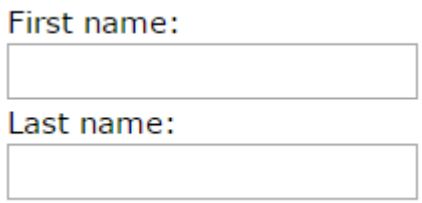
HTML forms are used to collect user input.

Type	Description
text	Defines normal text input
radio	Defines radio button input (for selecting one of many choices)
submit	Defines a submit button (for submitting the form)

Text Input

<input type="text"> defines a one-line input field for **text input:**

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

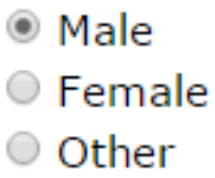


Radio Button Input

<input type="radio"> defines a **radio button.**

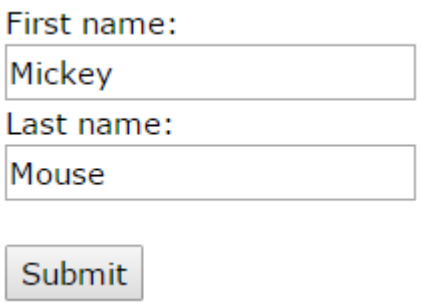
Radio buttons let a user select ONE of a limited number of choices:

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```



The Submit Button

<input type="submit"> defines a button for **submitting** a form to a **form-handler**.The form-handler is typically a server page with a script for processing input data. The form-handler is specified in the form's **action** attribute:




```
<form action="action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

The **action attribute** defines the action to be performed when the form is submitted.The common way to submit a form to a server, is by using a submit button.Normally, the form is submitted to a web page on a web server.

The Method Attribute

The **method attribute** specifies the HTTP method (**GET** or **POST**) to be used when submitting the forms:

<form action="action_page.php" **method="get"**>

<form action="action_page.php" **method="post"**>

GET: If the form submission is passive (like a search engine query), and without sensitive information.

action_page.php?firstname=Mickey&lastname=Mouse

POST: If the form is updating data, or includes sensitive information (password).

POST offers better security because the submitted data is not visible in the page address.

Grouping Form Data with <fieldset>

The **<fieldset>** element groups related data in a form.

The **<legend>** element defines a caption for the <fieldset> element.

```
<form action="action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey"><br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

The <select> Element (Drop-Down List)

The **<select>** element defines a **drop-down** list:

```
<select name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

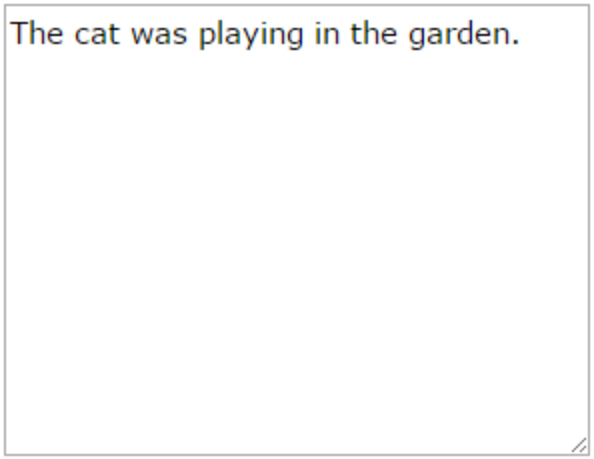
The **<option>** elements defines the options to select.

The list will normally show the first item as selected.

The <textarea> Element

The **<textarea>** element defines a multi-line input field (**a text area**):

```
<textarea name="message" rows="10" cols="30">
The cat was playing in the garden.
</textarea>
```



JAVASCRIPT

Scripting language – write small codes, executed as bigger part of HTML document

Object oriented scripting language

Designed for building web pages using HTML

Script is Interpreted by browser

- Plateform independent

JAVASCRIPT

Javascript code added to HTML can perform functions such as:

Decision making – based on user action (what he clicks)

Submitting forms

Performing mathematical operations

Data entry etc.

JAVASCRIPT

It allows interaction with the properties of objects

Internal built-in objects (eg *window* object) –

- Can access properties of windows to modify windows

Browser objects(eg *document* object)

- Display material on browser
- Can modify properties of document(font , color etc)

Javascript can run on both client side or server side

Client side

- By the browser

Server side

- Need some functionality (tool)

JAVASCRIPT

OBJECTS and METHODS

An object is collection of properties and methods which can be viewed, modified and interacted with.

Simple example of property is color

Document.bgcolor=“red”.

JAVASCRIPT

METHODS

In object oriented paradigm, methods refer to functions that can be used to manipulate objects and their properties.

Example: write(), which when invoked on the document object, causes a specific string of characters to be outputted

Document.write(“hello everyone”)

JAVASCRIPT

EXAMPLES

```
<HTML>

<TITLE> display text</TITLE>

<BODY>

<SCRIPT>

Document.writeln(“<H1> hello</H1>”);

</SCRIPT>


</BODY>


</HTML>
```

JAVASCRIPT

EXAMPLES

Simple event handler with confim box

Property – onclick (event handler)

```
<SCRIPT>
Function response()
{
Confirm(“hello there”);
}
</HTML>
<FORM>
<INPUT TYPE=“button” VALUE=“Click and see” onClick=“response()”>
</FORM>
```

JAVASCRIPT

EXAMPLES

Simple event handler with confim box

Property – onclick (event handler)

CONFIRM() method return a boolean value

“true” if ok is pressed

“false” if cancel is pressed

The return value can be used in decision logic

JAVASCRIPT

```
<html>
<head>
    <script>
    function newDoc() {
    var c=confirm("visit duet");
    if (c==true)
        window.location.assign("http://www.duet.edu.pk")
        else
            window.location.assign("http://www.google.com")
    }
    </script>
</head>
    <body>
    <input type="button" value="Load new document" onclick="newDoc()">
    </body>
</html>
```

JAVASCRIPT

alert

Javascript alert box, when clicked, displays a text and waits until visitor presses ok button

The basic command is

Alert(“any message to be displayed”)

Background bolor

Modify the bgColor attribute of the document class

example

On mouse click, the background color togges between blue, green and red

JAVASCRIPT

Background bolor

```
<html>
<head>
    <script>
    function newDoc() {
    if (document.bgColor=="red")
    document.bgColor="blue";
    else if (document.bgColor=="blue")
        document.bgColor="green";
        else
        document.bgColor="red";

    }
    </script>
</head>
    <body>
    <input type="button" value="Load new document" onclick="newDoc()">
    </body>
</html>
```

JAVASCRIPT

Read user input

```
<html>
<head>
    <script>
    document.writeln("<H2> guess weather</H2>");

    function newDoc(f) {
    a = confirm("will it rain today");

    if (a)
    f.option.value="rain";
    else
        f.option.value="no rain";

    }
    </script>
</head>
    <body>
    <form>
    <input type="button" value="click" onclick="newDoc(form)">
    <p>forecast:    <input type="text" name="option" size="9">

    </form>
    </body>
</html>
```

JAVASCRIPT

Simple mathematical calculation

Example

Extract data from form field into a variable

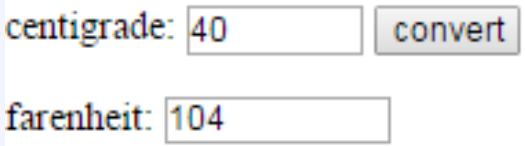
Perform mathematical calculation on the variable

Store the result back into the form field

JAVASCRIPT

Simple mathematical calculation

```
<html>
<head>
    <script>

    function newDoc(myform) {
    var cent = parseFloat(myform.C.value);
    farh=(cent*1.8)+32;
    myform.F.value=farh;


    }
    </script>
</head>
    <body>
    <form>
    centigrade: <input type="text" name=C size ="6" onclick="newDoc(form)">
    <input type="button" value="convert" onclick="newDoc(form)">
    <p>farenheit:   <input type="text" name=F size="9">

    </form>
    </body>
</html>
```



Access form data

Example

Extract the individual selected fields from a drop down menu

Use the selected value for further processing

Here the value simply echoed back through alert box

JAVASCRIPT

Access form data

```
<html>
<head>
    <script>

    function newDoc(f) {
    if(f.favcolor[0].selected)
    alert("favourite color is red");
    if(f.favcolor[1].selected)
    alert("favourite color is yellow");
    if(f.favcolor[2].selected)
    alert("favourite color is green");
    }
    </script>
</head>
    <body>
    <form>
    my favourite color:
    <select name="favcolor">
    <option> red
    <option> yellow
    <option> green
    </select>

    <input type="button" value="enter" onclick="newDoc(form)">

    </form>
    </body>
</html>
```


my favourite color: green enter
localhost says:
favourite color is green
Prevent this page from creating additional dialogs.
OK

JAVASCRIPT

array

```
<html>
<head>
    <script>
    var cricket = [];
cricket[0] = "wasim";
cricket[1] = "imran";
cricket[2] = "lara";

    </script>
</head>
    <body>
    <script>
    var number= cricket.length;
document.write("<H2> best cricketers of world </H2>" + "<p>");
for (i=0;i<number;i++)
    document.write("number:" + (i+1)+ ", " + cricket[i] + "<p>");

    </script>

    </body>
</html>
```

best cricketers of world

number:1, wasim

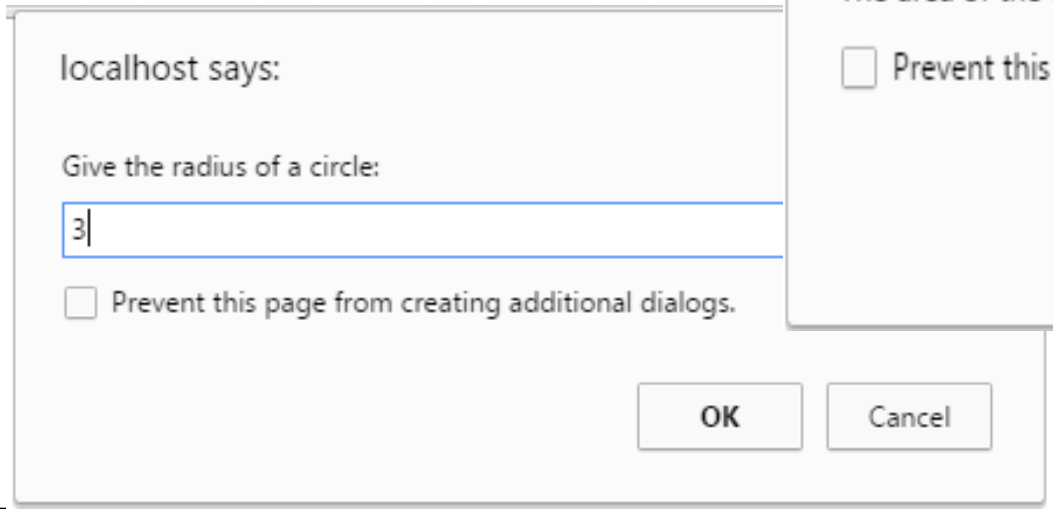
number:2, imran

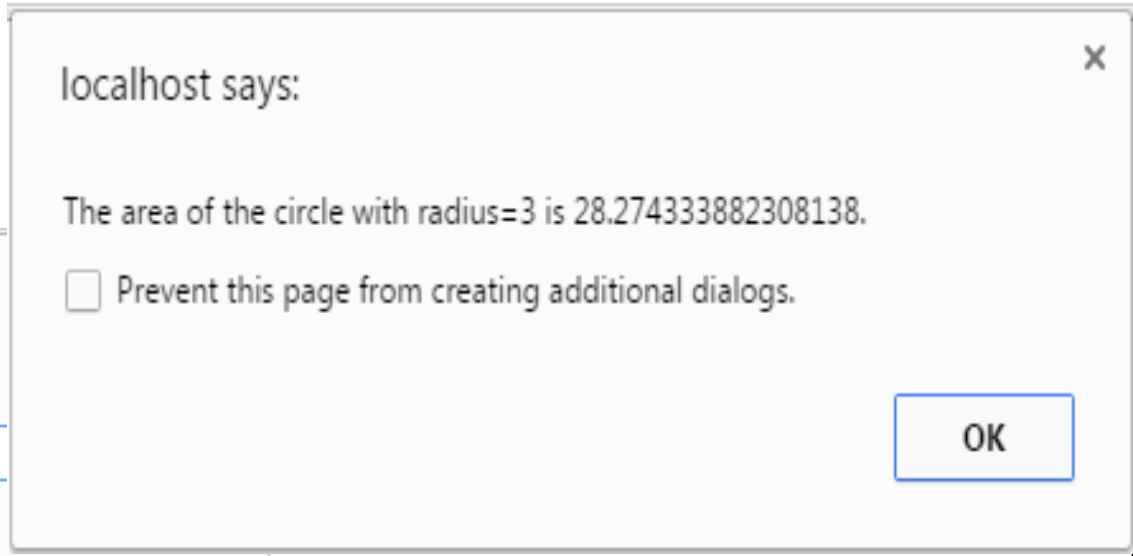
number:3, lara

JAVASCRIPT

prompt

```
<html>
<head>
<title>Calculate area of a circle.</title>
<script>
var radius=prompt("Give the radius of a circle: ");
radius=parseFloat(radius);
var area=Math.PI*radius*radius;
alert("The area of the circle with radius="+radius+" is "+area+".");
</script>
</head>
<body>
</body>
</html>
```





Airthmatic operators

Operator	Symbol	Examples	Precedence
Addition	+	`3 + 4`	2
Subtraction	–	`Z - 10`	2
Multiplication	*	`A*b`	1
Division	/	`z/3.333`	1
Modulus (remainder)	%	`17%3` (=2), `16.6%2.7` (=0.4)	1

Shorthand arithmetic/assignment operators

Operator	Implementation	Interpretation
+=	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;
++	x++; or ++x;	x=x+1;
--	y--; or --y;	x=x-1;

Some properties and methods of the JavaScript Math object

Property	Description
`Math.E`	Base of the natural logarithm, *e*, 2.71828
`Math.LN2`	Natural logarithm of 2, 0.693147
`Math.LN10`	Natural logarithm of 10, 2.302585
`Math.LOG2E`	Log to the base 2 of *e*, 1.442695
`Math.LOG10E`	Log to the base 10 of *e*, 0.434294
`Math.PI`	π, 3.1415927
`Math.SQRT1_2`	Square root of ½, 0.7071067
`Math.SQRT2`	Square root of 2, 1.4142136
Method	**Returns**
`Math.abs(x)`	Absolute value of *x*
`Math.acos(x)`	Arc cosine of *x*, ±π, for $-1 \le x \le 1$
`Math.asin(x)`	Arc sine of *x*, ±π/2, for $-1 \le x \le 1$
`Math.atan(x)`	Arc tangent of *x*, ±π/2, for $-\infty < x < \infty$ (compare with `Math.atan2(y,x)`)
`Math.atan2(y,x)`	Arc tangent of angle between *x*-axis and the point (*x*,*y*), measured counterclockwise (compare with `Math.atan(x)`)
`Math.ceil(x)`	Smallest integer greater than or equal to *x*
`Math.cos(x)`	Cosine of *x*, ±1

Math object

Within a with statement block, references to an object's properties and methods do not have to be prefixed with the object name and dot operator.

For uniformly distributed no: in range [1:n] -> Math.floor(n*(Math.random()%1) + 1);

```
<html>
<head>
<title>Demonstration of the Math object.</title>
<script language="javascript"type="text/javascript">
for (var i=1;i<=10;i++)
with (Math) {
var x=floor(100*(random()%1))+1;
document.write(x+"   :  "+sqrt(x)+"   :  "+pow(x,3)+"<br />");
}
</script>
</head>
<body>
</body>
</html>
```

18 : 4.242640687119285 : 5832
89 : 9.433981132056603 : 704969
19 : 4.358898943540674 : 6859
60 : 7.745966692414834 : 216000
25 : 5 : 15625
6 : 2.449489742783178 : 216
3 : 1.7320508075688772 : 27
85 : 9.219544457292887 : 614125
8 : 2.8284271247461903 : 512
25 : 5 : 15625

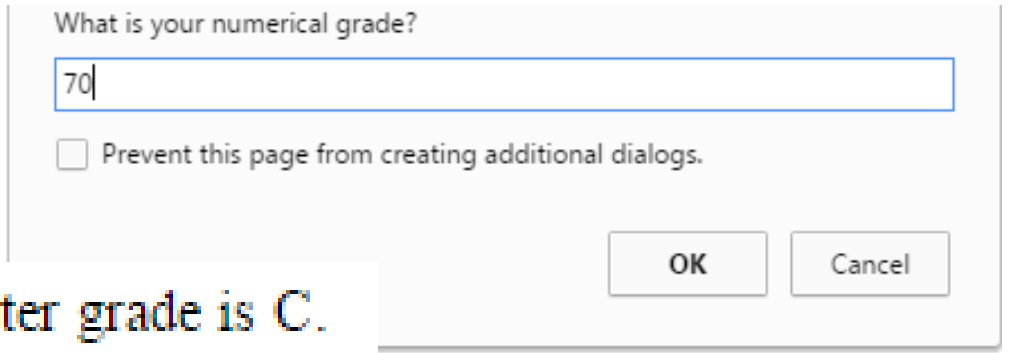
Relational and logical operators

Operator	Interpretation	Math Symbol	Precedence	Example	Value
Relational					
<	Less than	<	2	`-3.3<0`	`true`
>	Greater than	>	2	`17.7>17.5`	`true`
>=	Greater than or equal to	≥	2	`7.7>=7.7`	`true`
<=	Less than or equal to	≤	2	`7.6<=7.7`	`true`
==	Equal to, allowing for type conversion	=	3	`9=="9"`	`true`
===	Equal to, no type conversion	=	3	`9==="9"`	`false`
				`"a"==="a"`	`true`
!=	Not equal to, allowing for type conversion	≠	3	`9!="8"`	`true`
				`9!="9"`	`false`
!==	Not equal to, no type conversion	≠	3	`9!=="9"`	`true`
Logical					
&&	AND		4	`(x==3)&&(y<0)`	
||	OR		5	`(x==3)||(z==4)`	
!	NOT		1[a]	`!(x==3)`	

JAVASCRIPT

If else statement (Grading example)

```
<html>
<head>
<title>Get letter grade</title>
<script language="javascript"type="text/javascript">
var grade= parseFloat(prompt("What is your numerical grade?"));
document.write("For a numerical grade of "+grade+ ", your letter grade is ");
if (grade >= 90)document.write("A");
else if (grade >= 80)document.write("B");
else if (grade >= 70)document.write("C");
else if (grade >= 60)document.write("D");
else document.write("F");
document.write(".");
</script>
</head>
<body>
</body>
</html>
```



For a numerical grade of 70, your letter grade is C.

The switch Construct

```
<html>
<head>
<title>Days in Month</title>
<script language="javascript" type="text/javascript">
var month=prompt("Give month (1-12): ");
switch (month) {
  case "1":
  case "3":
  case "5":
  case "7":
  case "8":
  case "10":
  case "12":
    alert("There are 31 days in this month."); break;
  case "4":
  case "6":
  case "9":
  case "11":
    alert("There are 30 days in this month."); break;
```

The switch Construct

```
    case "2":
      alert("There are either 28 or 29 days in this
            month."); break;
default:
     alert("I do not understand your month entry.");
}
</script>
</head>
<body>
</body>
</html>
```

Loop Structures

Count-Controlled Loops

```
for (counter= {expression giving on initial value of counter};
  {expression giving high (or low) value of counter};
  {expression controlling incrementing (or decrementing) of counter})
```

Conditional Loops

conditions that control the execution or termination of a loop structure must be determined by values calculated inside the loop, while the script is running. Such circumstances require conditional loops.

There are two kinds of conditional loops: **pre-test** and **post-test** loops. The statements in pre-test loops may or may not be executed at all, depending on the original values of loop-related variables. Post-test loops are always executed at least once, and the values of loop-related variables are tested at the end of the loop.

Conditional Loops

pre-test loop:

```
while ({logical expression}) {
  {statements that result in changing the value of the pre-test logical
   expression}
}
```

post-test loop:

```
do {
  {statements that result in changing the value of the post-test logical
   expression}
} while ({logical expression});
```

JAVASCRIPT

array

```
<html>
<head>
<title>Site Names</title>
<script>
  var siteID = ["Drexel",3,"home",101];
  var i;
  for (i=0; i<siteID.length; i++)
   document.write(i+", "+siteID[i]+"<br />");
</script>
</head>
<body>
</body>
</html>
```

0, Drexel
1, 3
2, home
3, 101

Two dimension array (HW: solve the problem)

```
<html>
<head>
<title>Two-D arrays</title>
<script language="javascript" type="text/javascript">
var siteID = new Array();
siteID[0]=new Array("Drexel",39.955,-75.188,10.);
siteID[1]=new Array("home",40.178,-75.333,140.);
siteID[2]=new Array("NC",35.452,-81.022,246);
siteID[3]=new Array("Argentina",-34.617,-58.37,30.);
siteID[4]=new Array("Netherlands",52.382,4.933,-1);
var r,c,n_rows=siteID.length,n_cols=siteID[0].length;
for (r=0; r<n_rows; r++) {
  document.write(siteID[r][0]);
  for (c=1; c<n_cols; c++) {
      document.write(", "+siteID[r][c]);
  }
  document.write("<br />");
}
</script>
</head>
<body>
```

```
Drexel, 39.955, -75.188, 10
home, 40.178, -75.333, 140
NC, 35.452, -81.022, 246
Argentina, -34.617, -58.37, 30
```

Using Arrays to Access the Contents of Forms

Accessing Values of type="text"Fields

```
<html>
<head>
<title>Using the elements[] array to access values in forms.
</title>
</head>
<body>
<form name="myform">
A[0]<input type="text"value="3" /><br />
A[1]<input type="text"value="2" /><br />
</form>
<script language="javascript"type="text/javascript">
for(var i=0; i<document.myform.elements.length; i++) {
document.write("A["+i+"] = "+document.myform.elements[i].value+"<br />");
}
</script>
</body>
</html>
```

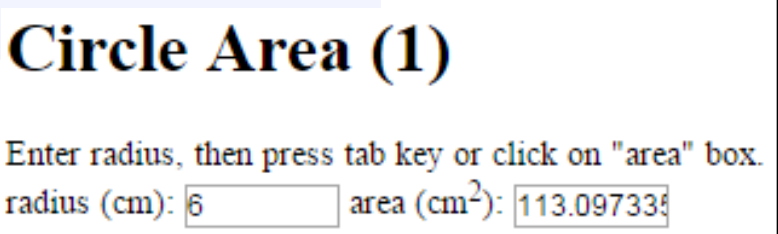
A[0] = 3
A[1] = 2

JAVASCRIPT

Onblur event handler (when object loses focus)

Passing Numerical Values to a Function

```
<html>
<head>
<title>Circle Area (1)</title>
<script language="javascript"type ="text/javascript">
function getArea(r) {
return Math.PI*r*r;
}
</script>
</head>
<body>
<h1>Circle Area (1)</h1>
<p>
<form>
Enter radius, then press tab key or click on "area"
box.<br />
radius (cm):
<input type="text" name="radius" size="6" maxlength="7"
value="-99", onblur = "area.value=getArea(parseFloat(radius.value));">

area (cm<sup>2</sup>):
<input type="text" name="area" size="6" maxlength="7"
value="-99">
</form>
</body>
</html>
```



Event handlers

Event	Trigger
`onAbort`	Abort selected in browser (stop loading of image or document), usually by clicking the Stop button
`onBlur`	When the object loses focus
`onChange`	When the object is changed (generally a form element)
`onClick`	When the object is clicked
`onDblClick`	When the object is double-clicked
`onDragDrop`	When an object is dropped into the user agent window (generally a file)
`onError`	When a JavaScript error occurs (not a browser error — only JavaScript code errors will trigger this event)
`onFocus`	When an object receives focus
`onKeyDown`	When the user presses a key
`onKeyPress`	When the user presses and/or holds down a key

Event	Trigger
`onKeyUp`	When the user releases a key
`onload`	When the object is loaded into the user agent (typically used with the `<body>` element to run a script when the document has completed loading)
`onMouseDown`	When the mouse button is depressed
`onMouseMove`	When the mouse is moved
`onMouseOut`	When the mouse pointer moves outside the boundary of an object
`onMouseOver`	When the mouse pointer moves within the boundary of an object
`onMouseUp`	When the mouse button is released
`onMove`	When an object (generally a window or frame) is moved
`onReset`	When the user selects a reset button
`onResize`	When an object (generally a window or frame) is resized
`onSelect`	When the user selects text within the object (generally a form element)
`onSubmit`	When the user selects a submit button
`onUnload`	When the object is unloaded from the user agent (generally used with the `<body>` element to run a script when the user navigates away from a document—a favorite tool of pop-up window coders)

JavaScript - Form Validation

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.

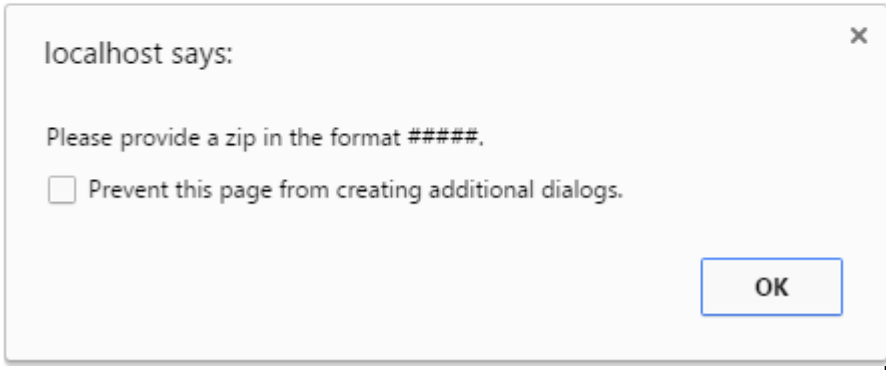
JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

Basic Validation – First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.

Data Format Validation – Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

```
<title>Form Validation</title>
  <script type="text/javascript">
        // Form validation code will come here.
  </script>
  </head>
<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">PK</option>
            <option value="2">UK</option>
          </select>
        </td>
      </tr>
      <tr>
        <td align="right"></td>
        <td><input type="submit" value="Submit" /></td>
      </tr>
    </table>
```

Name	wqe
EMail	ssada
Zip Code	
Country	[choose yours]
	Submit



JAVASCRIPT

```
function validate()
{

    if( document.myForm.Name.value == "" )
    {
        alert( "Please provide your name!" );
        document.myForm.Name.focus() ;
        return false;
    }

    if( document.myForm.EMail.value == "" )
    {
        alert( "Please provide your Email!" );
        document.myForm.EMail.focus() ;
        return false;
    }

    if( document.myForm.Zip.value == "" ||
    isNaN( document.myForm.Zip.value ) ||
    document.myForm.Zip.value.length != 5 )
    {
        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 );
}
//-->
```

p Language file

Data Format Validation

example shows how to validate an entered email address. An email address must contain at least a '@' sign and a dot (.). Also, the '@' must not be the first character of the email address, and the last dot must at least be one character after the '@' sign.

```
<script type="text/javascript">
   <!--
      function validateEmail()
      {
         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;
         }
         return( true );
      }
   //-->
</script>
```

JavaScript Errors - Throw and Try to Catch

The **try** statement lets you test a block of code for errors.

The **catch** statement lets you handle the error.

The **throw** statement lets you create custom errors.

The **finally** statement lets you execute code, after try and catch, regardless of the result.

Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things:

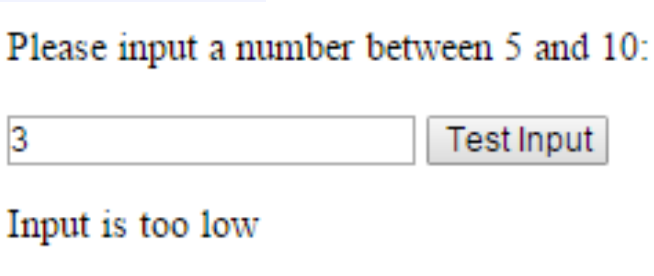
Input Validation Example

This example examines input. If the value is wrong, an exception (err) is thrown.

The exception (err) is caught by the catch statement and a custom error message is displayed:

JAVASCRIPT

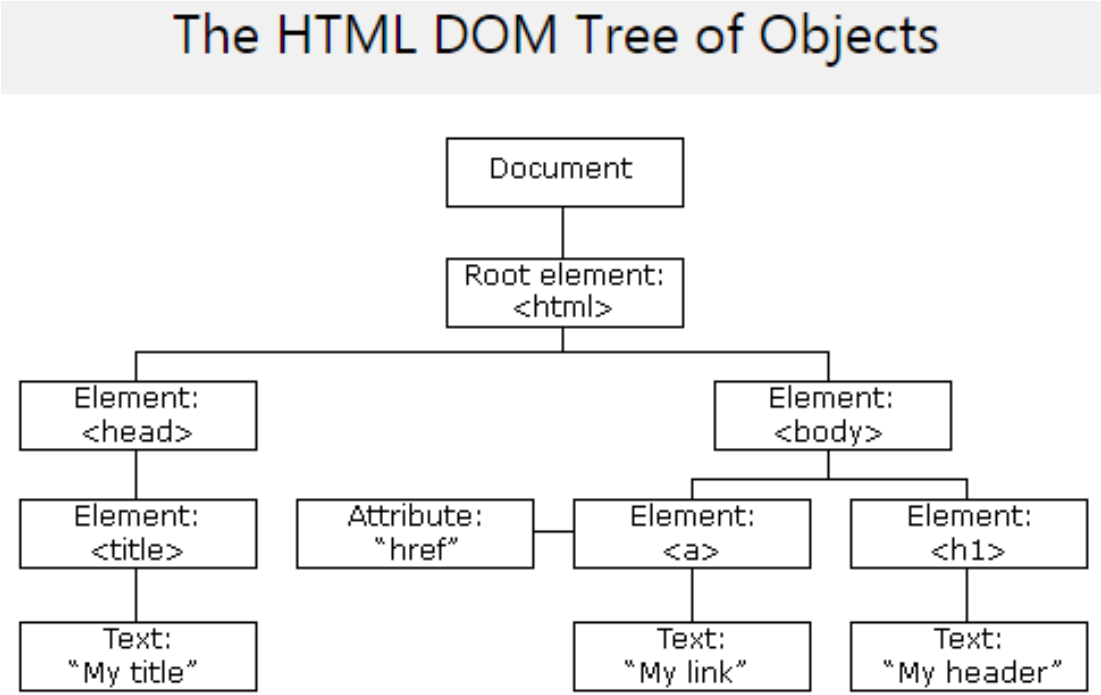
```
<html>
<body>

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

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

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

</body>
</html>
```



The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects:**



JAVASCRIPT

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

JavaScript can change all the HTML elements in the page

JavaScript can change all the HTML attributes in the page

JavaScript can change all the CSS styles in the page

JavaScript can remove existing HTML elements and attributes

JavaScript can add new HTML elements and attributes

JavaScript can react to all existing HTML events in the page

JavaScript can create new HTML events in the page

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

The HTML elements as **objects**

The **properties** of all HTML elements

The **methods** to access all HTML elements

The **events** for all HTML elements

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

The DOM gives you

-scripting access to all the elements on a web page. Inside the browser, the whole web page—paragraphs, forms, tables, etc.—is represented in an object hierarchy. Using JavaScript, you can dynamically create,modify and remove elements in the page.

-dynamically change style properties, which enables you to create effects, such as user-defined background colors and animations.

-document's **getElementById** method is the simplest way to access a specific element in a page

-The **getElementById** method returns objects called DOM nodes. Every piece of an HTML page (elements, attributes, text, etc.) is modeled in the web browser by a DOM node. All the nodes in a document make up the page's **DOM tree**, which describes the relationships among elements. Nodes are related to each other through child-parent relationships. An HTML5 element inside another element is said to be its **child**—the containing element is known as the **parent**. A node can have multiple children but only one parent. Nodes with the same parent node are referred to as **siblings**.

JAVASCRIPT

It's important to note that the DOM is a type of application program interface (API) allowing any programming language access to the structure of a Web document.

The main advantage of using the DOM is the ability to manipulate a document without another trip to the document's server.

As such, the DOM is typically accessed and used by client-side technologies, such as JavaScript.

API- is a set of routines, protocols, and tools for building software and applications.

getElementById

Finding a node on a page is the most fundamental task when working with the DOM from JavaScript. The most useful function is document.getElementById('elementId'). This function takes a string and returns the DOM element with that ID.

image of a star:

This is the DOM tree for the image:

```
<img id="star" src="star_off.gif">
```

JavaScript code below that finds the image element using getElementById, then sets the element's src property so that image is replaced.

```
document.getElementById('star').src = 'star_on.gif';
```

JAVASCRIPT

```
<html>
<head>
<script>
function getValue()
  {
  document.getElementById('star').src = 'html.gif';

  }
</script>
</head>
<body>

<img id="star" src="star_off.gif" onclick="getValue()">

</body>
</html>
```

Star_off.gif --→ html.gif

JAVASCRIPT -> childNodes

Sometimes the node you want to manipulate does not have an ID. This is particularly true of text nodes, which can't have IDs. But there may still be ways to find a node that doesn't have an ID. Each node has a childNodes property that contains an ordered array of all its children. One can index into this array.

Here is a span containing three stars:

This is the DOM tree for the span:

```
<span id="stars">
 ├ <img src="star_off.gif">
 ├ <img src="star_off.gif">
 └ <img src="star_off.gif">
</span>
```

finds the above span element using getElementById, finds the middle star using childNodes, then sets the element's src property . The other two stars should be 'off'. Remember that arrays in JavaScript start with index 0

```
document.getElementById('stars').childNodes[1].src = 'star_on.gif';
```

JAVASCRIPT

```
<html>
<head>
<script>
function getValue()
  {
document.getElementById('stars').childNodes[1].src = 'html.gif'

  }
</script>
</head>
<body>

<span id="stars" onclick="getValue()">
 <img src="star_off.gif">
 <img src="star_off.gif">
 <img src="star_off.gif">
</span>

</body>
</html>
```

Before click

after click

JavaScript - Animation

You can use JavaScript to create a complex animation having, but not limited to, the following elements −

Fireworks

Fade Effect

Roll-in or Roll-out

Page-in or Page-out

Object movements

JAVASCRIPT

JavaScript can be used to move a number of DOM elements (<img />, or any other HTML element) around the page according to some sort of pattern determined by a logical equation or function.

following two functions to be frequently used in animation programs.

setTimeout(function, duration) – This function calls **function** after **duration** milliseconds from now.

setInterval(function, duration) – This function calls **function** after every **duration** milliseconds.

clearTimeout(setTimeout_variable) – This function calls clears any timer set by the setTimeout() functions.

JavaScript can also set a number of attributes of a DOM object including its position on the screen. You can set *top* and left attribute of an object to position it anywhere on the screen. Here is its syntax

```
// Set distance from left edge of the screen.
object.style.left = distance in pixels or points;

or

// Set distance from top edge of the screen.
object.style.top = distance in pixels or points;
```

Manual Animation

So let's implement one simple animation using DOM object properties and JavaScript functions as follows. The following list contains different DOM methods.

We are using the JavaScript function **getElementById()** to get a DOM object and then assigning it to a global variable **imgObj**.

We have defined an initialization function **init()** to initialize **imgObj**where we have set its **position** and **left** attributes.

We are calling initialization function at the time of window load.

Finally, we are calling **moveRight()** function to increase the left distance by 10 pixels. You could also set it to a negative value to move it to the left side.

JAVASCRIPT

```
<head>
    <title>JavaScript Animation</title>

    <script type="text/javascript">
        <!--
            var imgObj = null;

            function init(){
                imgObj = document.getElementById('myImage');
                imgObj.style.position= 'relative';
                imgObj.style.left = '0px';
            }

            function moveRight(){
                imgObj.style.left = parseInt(imgObj.style.left) + 10 + 'px';
            }

            window.onload =init;
        //-->
    </script>

</head>

<body>

    <form>
        <img id="myImage" src="html.gif" />
        <p>Click button below to move the image to right</p>
        <input type="button" value="Click Me" onclick="moveRight();" />
    </form>

</body>
</html>
```

Click button below to move the image to right

Click Me

Important program
NB program
DOM…next

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 12.2: dom.html -->
<!-- Basic DOM functionality. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Basic DOM Functionality</title>
      <style type = "text/css">
         h1, h3       { text-align: center;
                        font-family: tahoma, geneva, sans-serif }
         p            { margin-left: 5%;
                        margin-right: 5%;
                        font-family: arial, helvetica, sans-serif }
         ul           { margin-left: 10% }
         a            { text-decoration: none }
         a:hover      { text-decoration: underline }
         .nav         { width: 100%;
                        border-top: 3px dashed blue;
                        padding-top: 10px }
         .highlighted { background-color: yellow }
         .submit      { width: 120px }
      </style>
      <script type = "text/javascript">
```

```
26          <!--
27          var currentNode; // stores the currently highlighted node
28          var idcount = 0; // used to assign a unique id to new elements
29
30          // get and highlight an element by its id attribute
31          function byId()
32          {
```

```
33             var id = document.getElementById( "gbi" ).value;
34             var target = document.getElementById( id );
35
36             if ( target )
37                switchTo( target );
38          } // end function byId
39
40          // insert a paragraph element before the current element
41          // using the insertBefore method
42          function insert()
43          {
44             var newNode = createNewNode(
45                document.getElementById( "ins" ).value );
46             currentNode.parentNode.insertBefore( newNode, currentNode );
47             switchTo( newNode );
48          } // end function insert
49
50          // append a paragraph node as the child of the current node
51          function appendNode()
52          {
53             var newNode = createNewNode(
54                document.getElementById( "append" ).value );
55             currentNode.appendChild( newNode );
56             switchTo( newNode );
57          } // end function appendNode
58
```

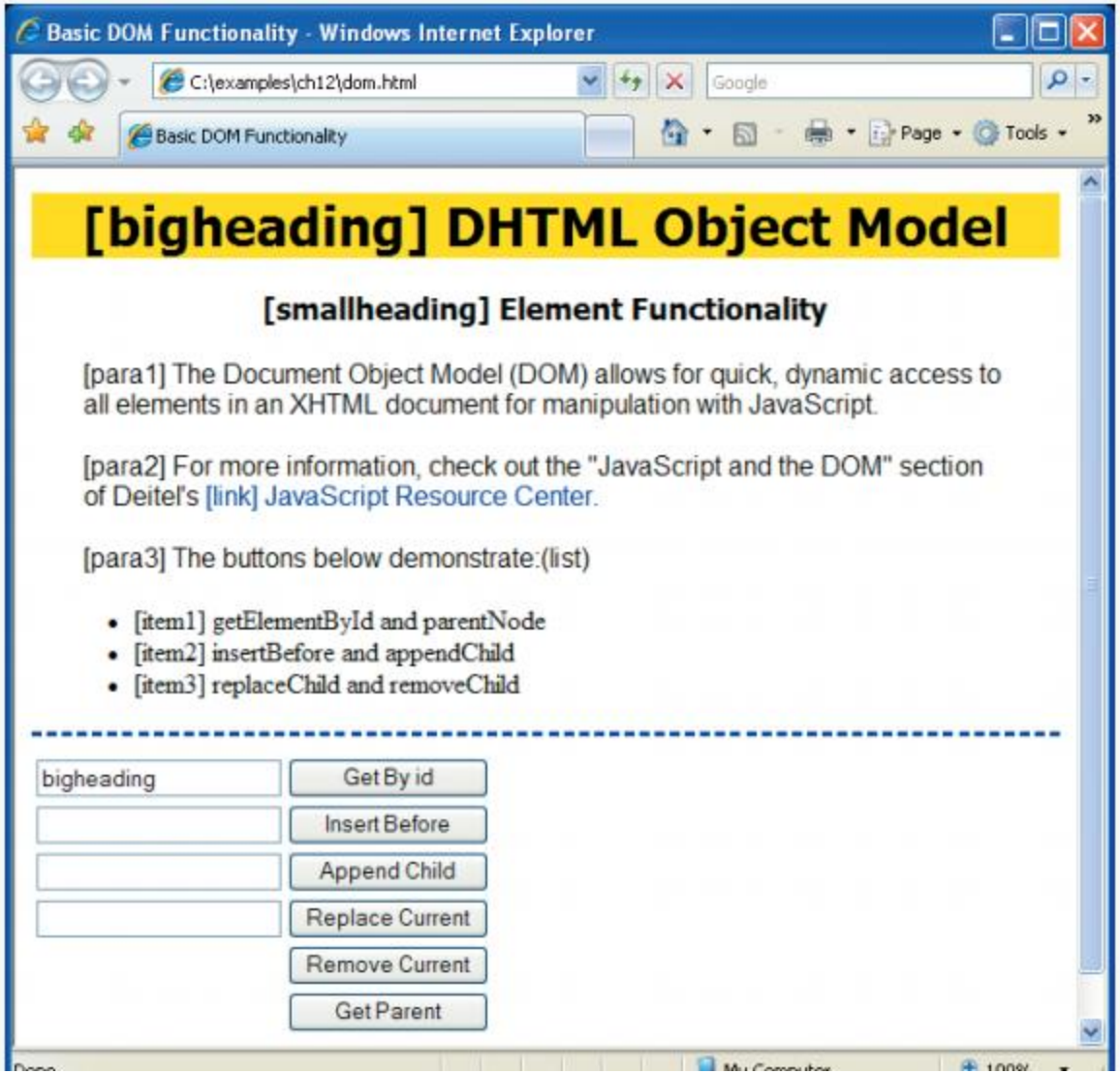
```
59          // replace the currently selected node with a paragraph node
60          function replaceCurrent()
61          {
62             var newNode = createNewNode(
63                document.getElementById( "replace" ).value );
64             currentNode.parentNode.replaceChild( newNode, currentNode );
65             switchTo( newNode );
66          } // end function replaceCurrent
67
68          // remove the current node
69          function remove()
70          {
71             if ( currentNode.parentNode == document.body )
72                alert( "Can't remove a top-level element." );
73             else
74             {
75                var oldNode = currentNode;
76                switchTo( oldNode.parentNode );
77                currentNode.removeChild( oldNode );
78             }
79          } // end function remove
80
81          // get and highlight the parent of the current node
82          function parent()
83          {
84             var target = currentNode.parentNode;
```

```
86              if ( target != document.body )
87                 switchTo( target );
88              else
89                 alert( "No parent." );
90           } // end function parent
91
92           // helper function that returns a new paragraph node containing
93           // a unique id and the given text
94           function createNewNode( text )
95           {
96              var newNode = document.createElement( "p" );
97              nodeId = "new" + idcount;
98              ++idcount;
99              newNode.id = nodeId;
100             text = "[" + nodeId + "] " + text;
101             newNode.appendChild(document.createTextNode( text ) );
102             return newNode;
103          } // end function createNewNode
104
105          // helper function that switches to a new currentNode
106          function switchTo( newNode )
107          {
108             currentNode.className = ""; // remove old highlighting
109             currentNode = newNode;
110             currentNode.className = "highlighted"; // highlight new node
111             document.getElementById( "gbi" ).value = currentNode.id;
```

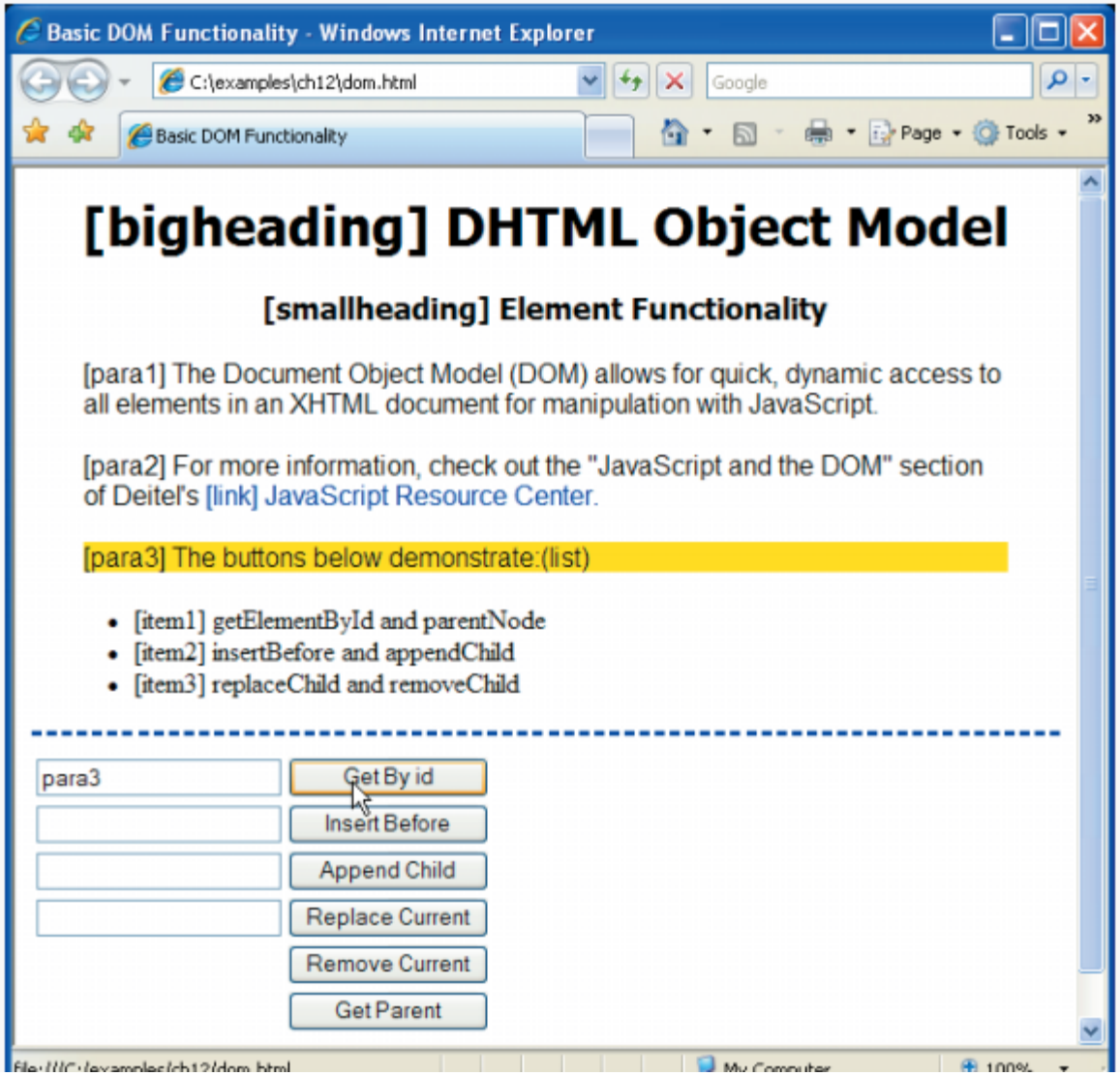
```
112            } // end function switchTo
113            // -->
114         </script>
115      </head>
116      <body onload = "currentNode = document.getElementById( 'bigheading' )">
117         <h1 id = "bigheading" class = "highlighted">
118            [bigheading] DHTML Object Model</h1>
119         <h3 id = "smallheading">[smallheading] Element Functionality</h3>
120         <p id = "para1">[para1] The Document Object Model (DOM) allows for
121            quick, dynamic access to all elements in an XHTML document for
122            manipulation with JavaScript.</p>
123         <p id = "para2">[para2] For more information, check out the
124            "JavaScript and the DOM" section of Deitel's
125            <a id = "link" href = "http://www.deitel.com/javascript">
126               [link] JavaScript Resource Center.</a></p>
127         <p id = "para3">[para3] The buttons below demonstrate:(list)</p>
128         <ul id = "list">
129            <li id = "item1">[item1] getElementById and parentNode</li>
130            <li id = "item2">[item2] insertBefore and appendChild</li>
131            <li id = "item3">[item3] replaceChild and removeChild </li>
132         </ul>
133         <div id = "nav" class = "nav">
134            <form onsubmit = "return false" action = "">
135               <table>
136                  <tr>
137                     <td><input type = "text" id = "gbi"
138                        value = "bigheading" /></td>
```

```
139                     <td><input type = "submit" value = "Get By id"
140                        onclick = "byId()" class = "submit" /></td>
141                  </tr><tr>
142                     <td><input type = "text" id = "ins" /></td>
143                     <td><input type = "submit" value = "Insert Before"
144                        onclick = "insert()" class = "submit" /></td>
145                  </tr><tr>
146                     <td><input type = "text" id = "append" /></td>
147                     <td><input type = "submit" value = "Append Child"
148                        onclick = "appendNode()" class = "submit" /></td>
149                  </tr><tr>
150                     <td><input type = "text" id = "replace" /></td>
151                     <td><input type = "submit" value = "Replace Current"
152                        onclick = "replaceCurrent()" class = "submit" /></td>
153                  </tr><tr><td />
154                     <td><input type = "submit" value = "Remove Current"
155                        onclick = "remove()" class = "submit" /></td>
156                  </tr><tr><td />
157                     <td><input type = "submit" value = "Get Parent"
158                        onclick = "parent()" class = "submit" /></td>
159                  </tr>
160               </table>
161            </form>
162         </div>
163      </body>
164   </html>
```

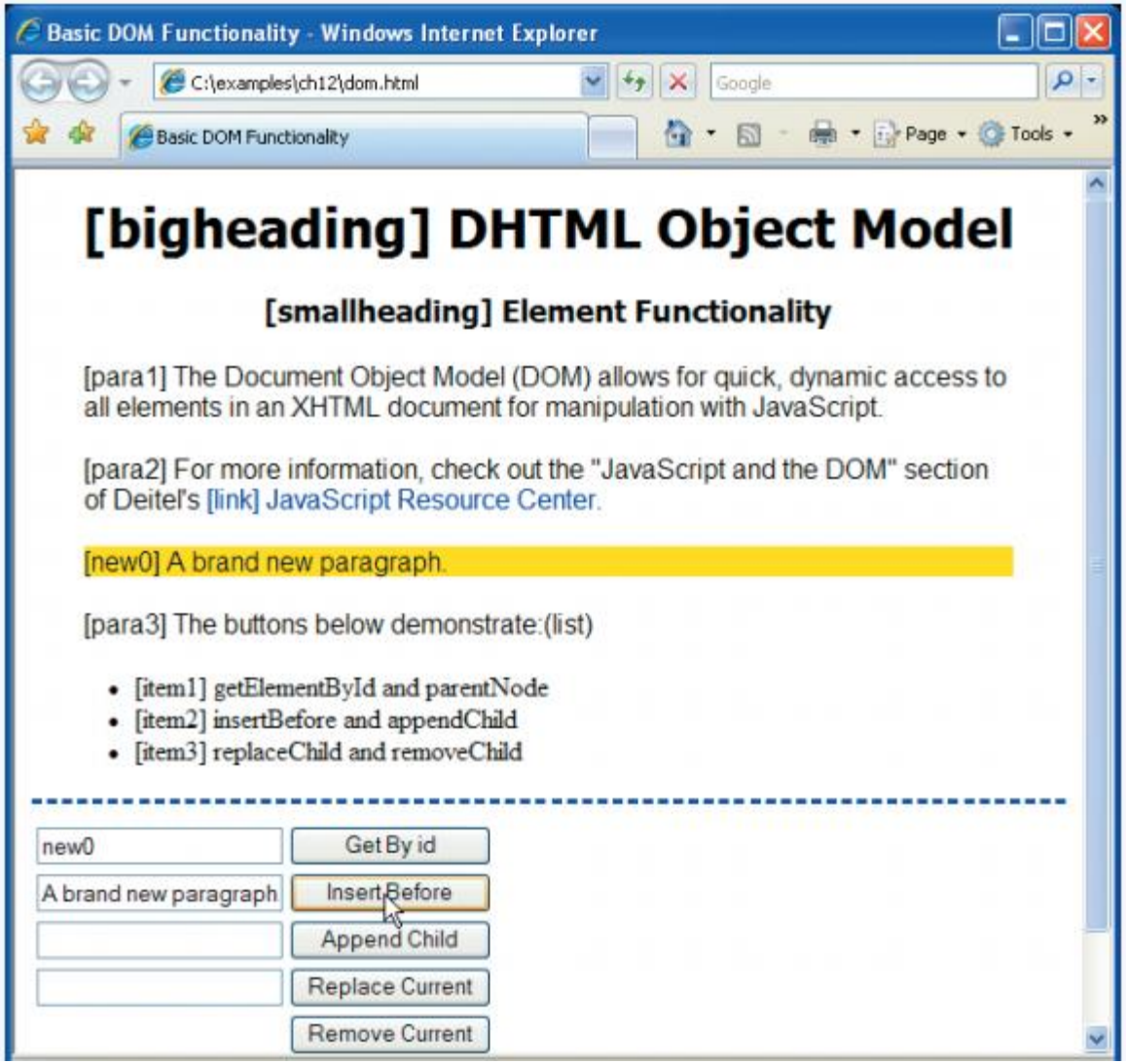
a) This is the page when it first loads. It begins with the large heading highlighted.



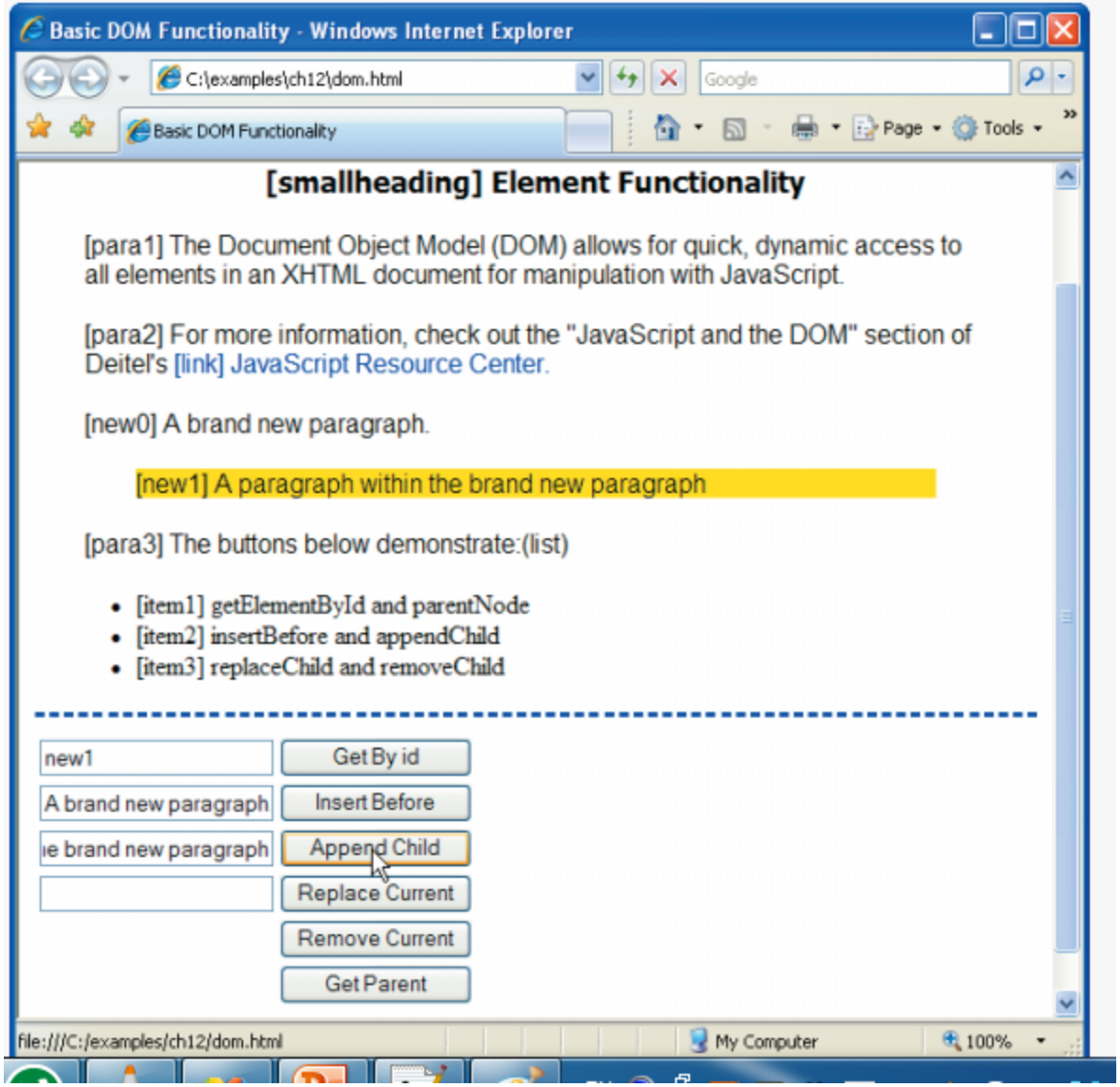
b) This is the document after using the Get By id button to select para3.



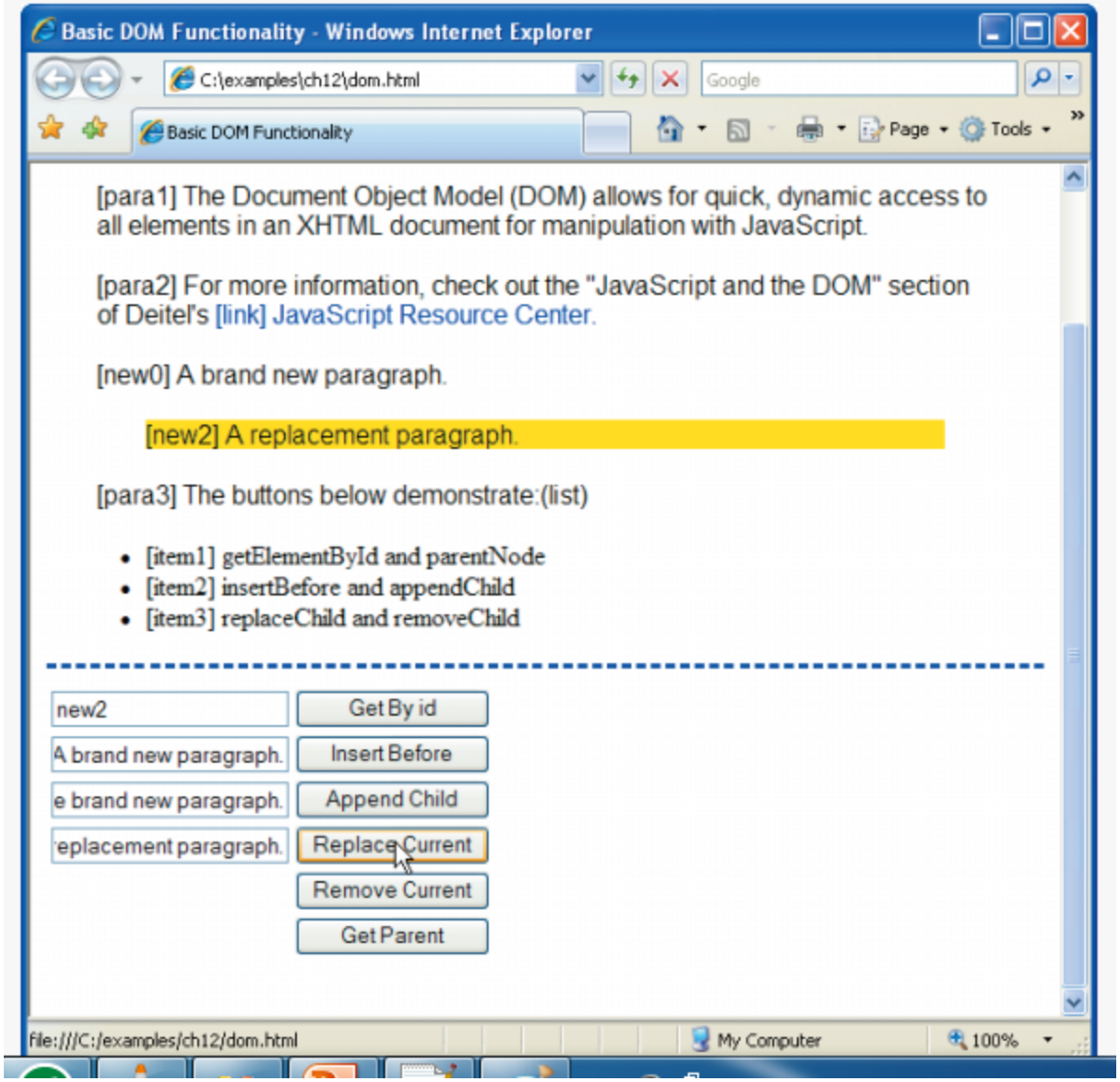
c) This is the document after inserting a new paragraph before the selected one.



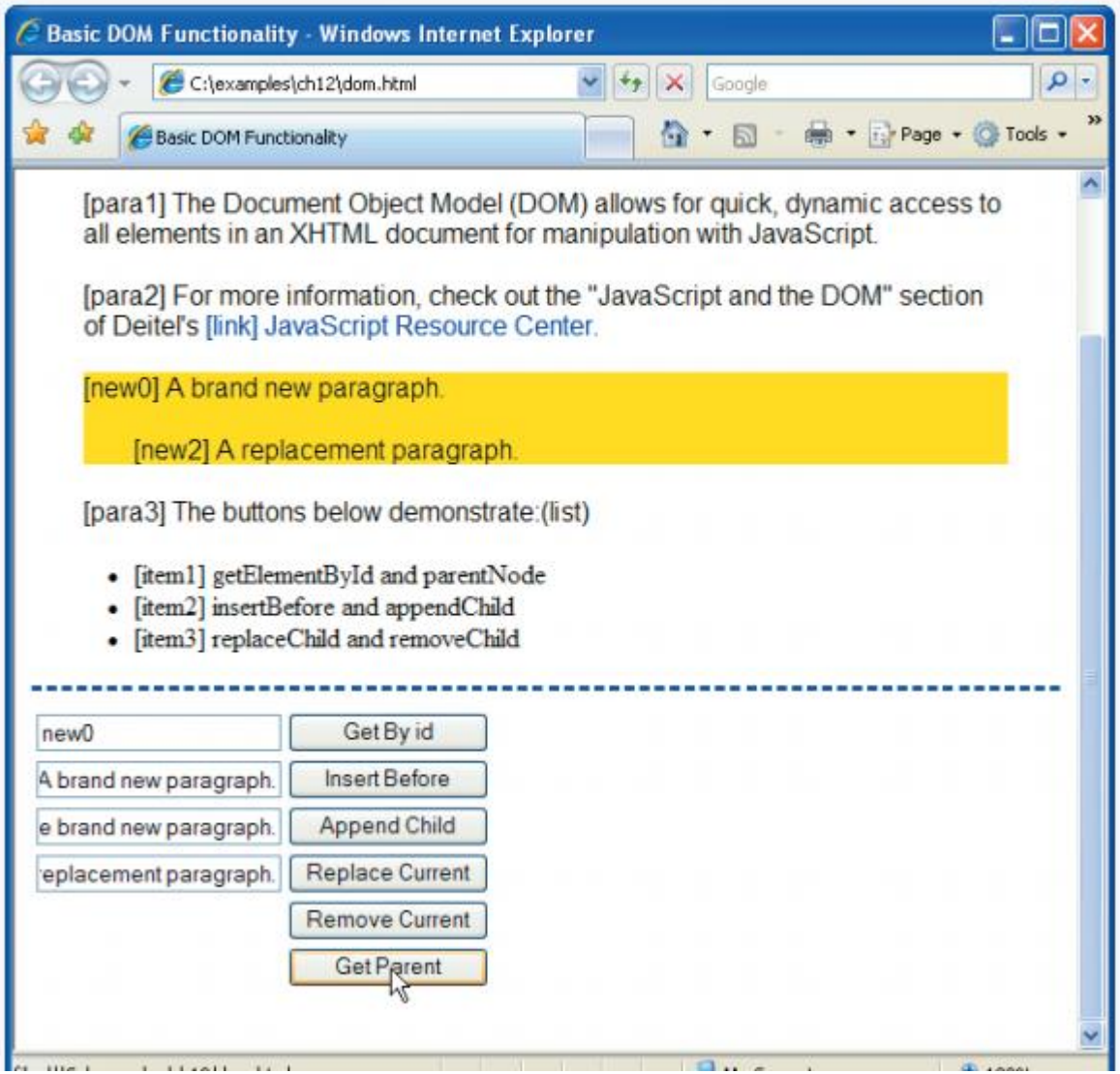
d) Using the Append Child button, a child paragraph is created.



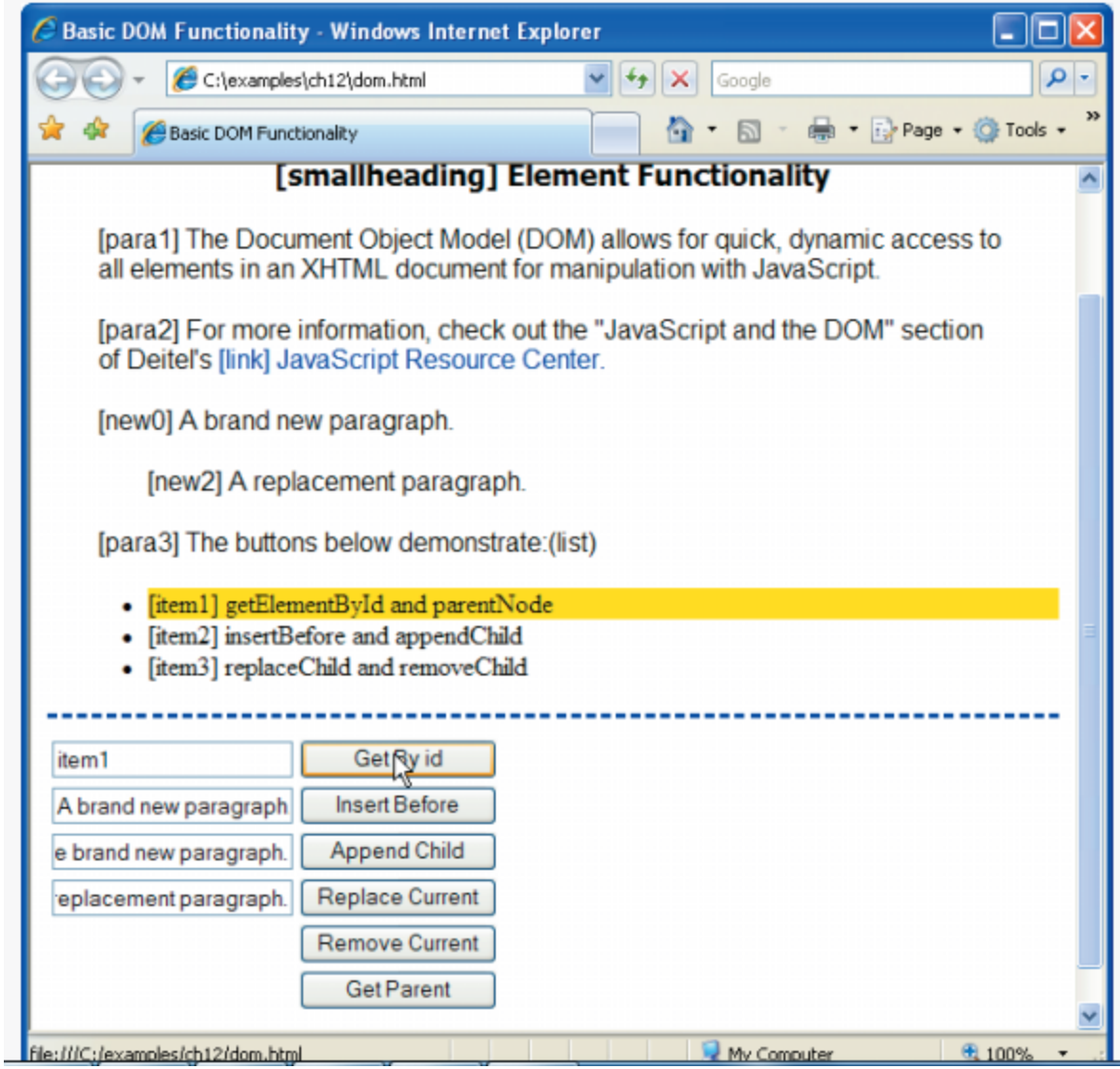
e) The selected paragraph is replaced with a new one.



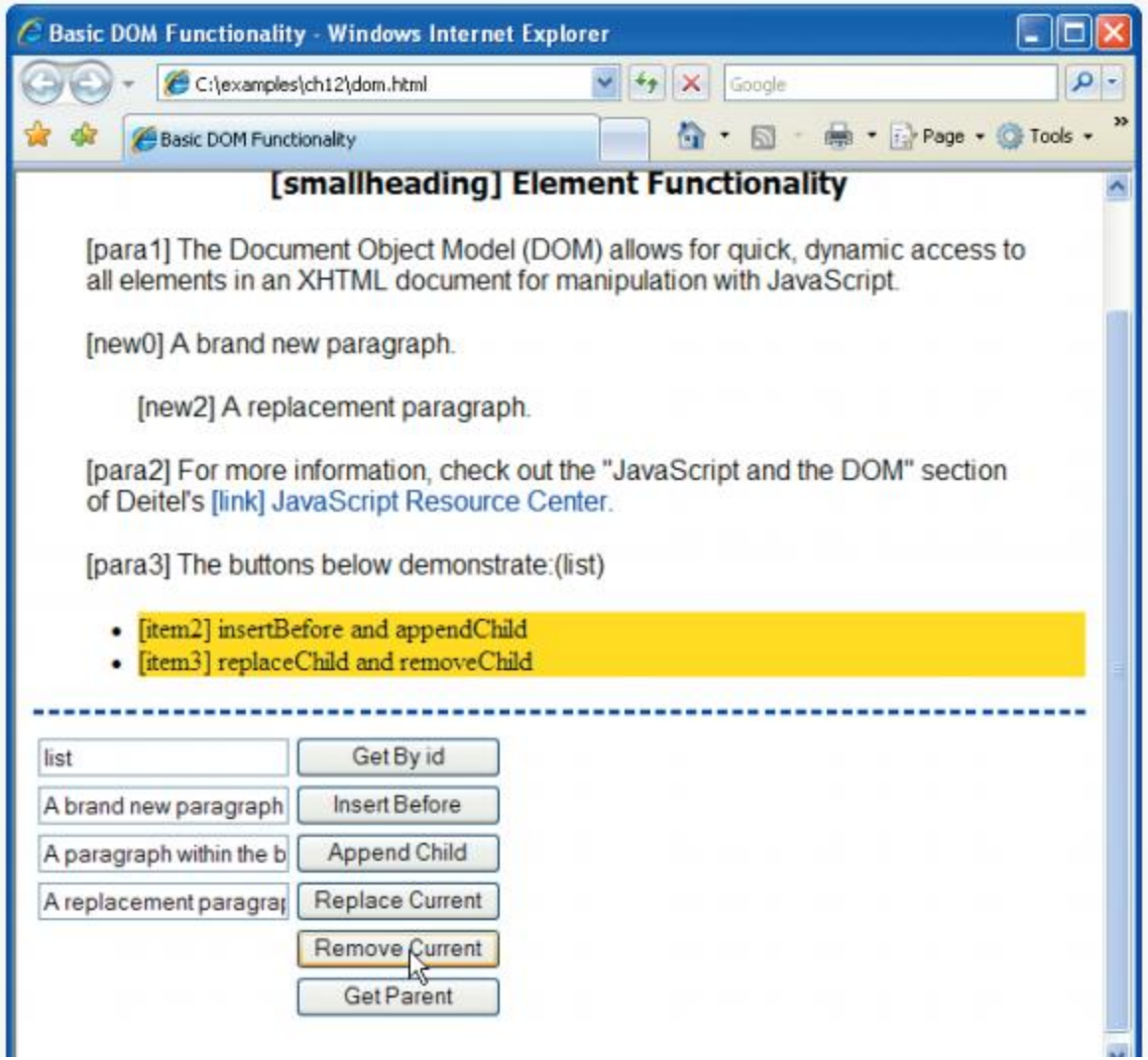
f) The Get Parent button gets the parent of the selected node.



g) Now we select the first list item.



h) The Remove Current button removes the current node and selects its parent.



PHP

PHP originally stood for *Personal Home Page*, but it now stands for the- *PHP: Hypertext Preprocessor*

When an HTML/JavaScript document is accessed online or locally with a browser, only the contents of that document are available. JavaScript code cannot access data stored elsewhere on a server. This restriction is inherent in the language syntax and operating environment and applies regardless of whether the server is actually at a different location, a remote server or whether external data exist on a local server

Creating (Declaring) PHP Variables

```php
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

Rules for PHP variables:

A variable starts with the $ sign, followed by the name of the variable

A variable name must start with a letter or the underscore character

A variable name cannot start with a number

A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

Variable names are case-sensitive ($age and $AGE are two different variables)

Output Variables

The PHP echo statement is often used to output data to the screen.

```
<?php
$txt = “php.com";
echo "I study on $txt!";
?>
```

The following example will produce the same output as the example above:

```
<?php
$txt = “php.com";
echo "I study on" . $txt . "!";
?>
```

PHP

PHP automatically converts the variable to the correct data type, depending on its value.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

local

global

static

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

Global and Local Scope

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

PHP The global Keyword

The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function):

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

012

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```

PHP Data Types

PHP supports the following data types:

String

Integer

Float (floating point numbers - also called double)

Boolean

Array

Object

NULL

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

PHP integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

An integer must have at least one digit

An integer must not have a decimal point

An integer can be either positive or negative

Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example $x is an integer. The PHP var_dump() function returns the data type and value

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing

PHP Array

An array stores multiple values in one single variable.

```
$cars = array("Volvo","BMW","Toyota");
```

PHP Object

An object is a data type which stores data and information on how to process that data.

In PHP, an object must be explicitly declared.

First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

```php
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

If a variable is created without a value, it is automatically assigned a value of NULL.

PHP String Functions

```
echo strlen("Hello world!"); // outputs 12

echo str_word_count("Hello world!"); // outputs 2

echo strpos("Hello world!", "world"); // outputs 6

echo str_replace("world", “14batch", "Hello world!"); // outputs Hello 14batch!
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing

A PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Increment/Decrement operators

Logical operators

String operators

Array operators

PHP Arithmetic Operators

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

PHP Assignment Operators

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Logical Operators

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

if statement - executes some code if one condition is true

if...else statement - executes some code if a condition is true and another code if that condition is false

if...elseif....else statement - executes different codes for more than two conditions

switch statement - selects one of many blocks of code to be executed

PHP

```php
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

```php
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

PHP

```php
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

SWITCH STATEMENT

Use the switch statement to **select one of many blocks of code to be executed.**

Use **break** to prevent the code from running into the next case automatically. The **default** statement is used if no match is found.

```php
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Loops

while - loops through a block of code as long as the specified condition is true

do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true

for - loops through a block of code a specified number of times

foreach - loops through a block of code for each element in an array

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

The example below first sets a variable $x to 1 ($x = 1). Then, the while loop will continue to run as long as $x is iess than, or equal to 5 ($x <= 5). $x will increase by 1 each time the loop runs ($x++):

Example

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

Parameters:

init counter: Initialize the loop counter value

test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment counter: Increases the loop counter value

The example below displays the numbers from 0 to 10:

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array ($colors):

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

PHP User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute immediately when a page loads.

A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user defined function declaration starts with the word "function":

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

```
<?php
function familyName($fname) {
    echo "$fname.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

```
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Functions - Returning values

To let a function return a value, use the return statement:

A float (floating point number) is a number with a decimal point or a number in exponential form.

```
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

PHP

Array

An array stores multiple values in one single variable:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Create an Array in PHP

In PHP, the array() function is used to create an array:

$cars = array("Volvo", "BMW", "Toyota");

Get The Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Loop Through an Indexed Array

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

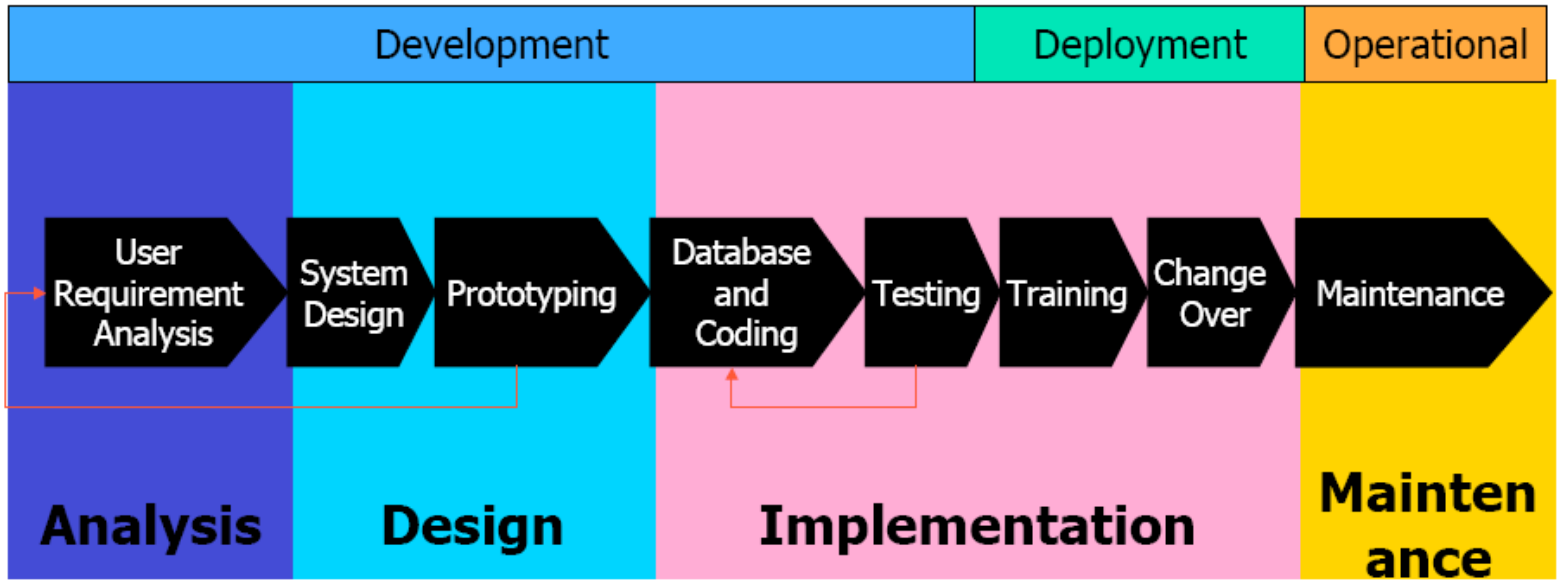
PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing

Typical life cycle



```
Welcome John
Your email address is john.doe@example.com
```

PHP Form Handling

The PHP superglobals $_GET and $_POST are used to collect form-data.

The form data is sent with the HTTP POST method

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

PHP multi dimension array

The PHP - Two-dimensional Arrays

A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

PHP

```php
<?php
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";

for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

Row number 0

- Volvo
- 22
- 18

Row number 1

- BMW
- 15
- 13

Row number 2

- Saab
- 5
- 2

Row number 3

- Land Rover
- 17
- 15

PHP

HOW???????????

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

PHP Include Files

The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement. It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

```
<?php
$color='red';
$car='BMW';
?>
```

```
<html>
<body>
<h1>Welcome to my home page!</h1>
<?php include 'vars.php';
echo "I have a $color $car.";
?>
</body>
</html>
```

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

PHP

What is a PHP Session?

Start a PHP Session

A session is started with the session_start() function.

Session variables are set v le: $_SESSION.

let's create a new page ca . In this page, we start a new PHP session and set

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

PHP

What is a PHP Session?

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session Variable values are stored in the global $_SESSION variable:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

PHP Exception Handling

Exceptions are used to change the normal flow of a script if a specified error occurs.

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

This is what normally happens when an exception is triggered:

-The current code state is saved

-The code execution will switch to a predefined (custom) exception handler function

-Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

PHP MySQL Database

MySQL is a database system used on the web

MySQL is a database system that runs on a server

MySQL is ideal for both small and large applications

MySQL is very fast, reliable, and easy to use

MySQL uses standard SQL

MySQL compiles on a number of platforms

MySQL is free to download and use

MySQL is developed, distributed, and supported by Oracle Corporation

PHP

MySQL works very well in combination of various programming languages like PERL, C, C++, JAVA and PHP.

Out of these languages, PHP is the most popular one because of its web application development capabilities.

MySQL Connection using PHP Script:

PHP provides **mysql_connect()** function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success or FALSE on failure.

connection mysql_connect(server,user,passwd,new_link,client_flag);

Parameter	Description
server	Optional - The host name running database server. If not specified, then default value is **localhost:3036**.
user	Optional - The username accessing the database. If not specified, then default is the name of the user that owns the server process.
passwd	Optional - The password of the user accessing the database. If not specified, then default is an empty password.
new_link	Optional - If a second call is made to mysql_connect() with the same arguments, no new connection will be established; instead, the identifier of the already opened connection will be returned.

client_flags	Optional - A combination of the following constants: ▫ MYSQL_CLIENT_SSL - Use SSL encryption ▫ MYSQL_CLIENT_COMPRESS - Use compression protocol ▫ MYSQL_CLIENT_IGNORE_SPACE - Allow space after function names ▫ MYSQL_CLIENT_INTERACTIVE - Allow interactive timeout seconds of inactivity before closing the connection

PHP

You can disconnect from MySQL database anytime using another PHP function**mysql_close()**. This function takes a single parameter, which is a connection returned by **mysql_connect()** function.

bool mysql_close (resource $link_identifier);

PHP

```
<html>
<head>
<title>Connecting MySQL Server</title>
</head>
<body>
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'guest';
   $dbpass = 'guest123';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);
   if(! $conn )
   {
     die('Could not connect: ' . mysql_error());
   }
   echo 'Connected successfully';
   mysql_close($conn);
?>
</body>
</html>
```

PHP

Create Database using PHP Script:

PHP uses **mysql_query** function to create or delete a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

bool mysql_query(sql, connection);

Parameter	Description
sql	Required - SQL query to create or delete a MySQL database
connection	Optional - if not specified, then last opened connection by mysql_connect will be used.

PHP

```
<html>
<head>
<title>Creating MySQL Database</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = 'CREATE DATABASE TUTORIALS';
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not create database: ' . mysql_error());
}
echo "Database TUTORIALS created successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

PHP

Selecting MySQL Database from Command Prompt:

PHP provides function **mysql_select_db** to select a database. It returns TRUE on success or FALSE on failure.

bool mysql_select_db(db_name, connection);

Parameter	Description
db_name	Required - MySQL Database name to be selected
connection	Optional - if not specified, then last opened connection by mysql_connect will be used.

PHP

```
<html>
<head>
<title>Selecting MySQL Database</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'guest';
$dbpass = 'guest123';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_select_db( 'TUTORIALS' );
mysql_close($conn);
?>
</body>
</html>
```

Create MySQL Tables

The table creation command requires:

Name of the table

Names of fields

Definitions for each field

To create new table in any existing database you would need to use PHP function **mysql_query()**. You will pass its second argument with proper SQL command to create a table.

PHP

Create MySQL Tables

```
<html>
<head>
<title>Creating MySQL Tables</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = "CREATE TABLE tutorials_tbl( ".
       "tutorial_id INT NOT NULL AUTO_INCREMENT, ".
       "tutorial_title VARCHAR(100) NOT NULL, ".
       "tutorial_author VARCHAR(40) NOT NULL, ".
       "submission_date DATE, ".
       "PRIMARY KEY ( tutorial_id )); ";
mysql_select_db( 'TUTORIALS' );
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not create table: ' . mysql_error());
}
echo "Table created successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

PHP

MySQL Insert Query

```
<html>
<head>
<title>Add New Record in MySQL Database</title>
</head>
<body>
<?php
if(isset($_POST['add']))
{
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}

if(! get_magic_quotes_gpc() )
{
   $tutorial_title = addslashes ($_POST['tutorial_title']);
   $tutorial_author = addslashes ($_POST['tutorial_author']);
}
else
{
   $tutorial_title = $_POST['tutorial_title'];
   $tutorial_author = $_POST['tutorial_author'];
}
$submission_date = $_POST['submission_date'];
```

PHP

MySQL Insert Query

```
$sql = "INSERT INTO tutorials_tbl ".
       "(tutorial_title,tutorial_author, submission_date) ".
       "VALUES ".
       "('$tutorial_title','$tutorial_author','$submission_date')";
mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not enter data: ' . mysql_error());
}
echo "Entered data successfully\n";
mysql_close($conn);
}
else
{
?>
<form method="post" action="<?php $_PHP_SELF ?>">
<table width="600" border="0" cellspacing="1" cellpadding="2">
<tr>
<td width="250">Tutorial Title</td>
<td>
<input name="tutorial_title" type="text" id="tutorial_title">
</td>
</tr>
<tr>
<td width="250">Tutorial Author</td>
<td>
<input name="tutorial_author" type="text" id="tutorial_author">
</td>
</tr>
<tr>
```

MySQL Insert Query

```
<tr>
<td width="250">Submission Date [ yyyy-mm-dd ]</td>
<td>
<input name="submission_date" type="text" id="submission_date">
</td>
</tr>
<tr>
<td width="250"> </td>
<td> </td>
</tr>
<tr>
<td width="250"> </td>
<td>
<input name="add" type="submit" id="add" value="Add Tutorial">
</td>
</tr>
</table>
</form>
<?php
}
?>
</body>
</html>
```

PHP

While doing data insert, it's best practice to use function**get_magic_quotes_gpc()** to check if current configuration for magic quote is set or not. If this function returns false, then use function **addslashes()** to add slashes before quotes.

MySQL Select Query

Following example will return all the records from **tutorials_tbl** table:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           1 | Learn PHP      | John Poul       | 2007-05-21      |
|           2 | Learn MySQL    | Abdul S         | 2007-05-21      |
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
3 rows in set (0.01 sec)

mysql>
```

PHP

MySQL Select Query

Try out the following example to Display all the records from tutorials_tbl table.

```php
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
                tutorial_author, submission_date
        FROM tutorials_tbl';

mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "Tutorial ID :{$row['tutorial_id']}  <br> ".
         "Title: {$row['tutorial_title']} <br> ".
         "Author: {$row['tutorial_author']} <br> ".
         "Submission Date : {$row['submission_date']} <br> ".
         "--------------------------------<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

MySQL Select Query

The content of the rows are assigned to the variable $row and the values in row are then printed.

NOTE: Always remember to put curly brackets when you want to insert an array value directly into a string.

In above example, the constant **MYSQL_ASSOC** is used as the second argument to PHP function **mysql_fetch_array()**, so that it returns the row as an associative array. With an associative array you can access the field by using their name instead of using the index.

PHP provides another function called **mysql_fetch_assoc()**, which also returns the row as an associative array.

PHP

MySQL works very well in combination of various programming languages like PERL, C, C++, JAVA and PHP.

Out of these languages, PHP is the most popular one because of its web application development capabilities.

PHP

MySQL works very well in combination of various programming languages like PERL, C, C++, JAVA and PHP.

Out of these languages, PHP is the most popular one because of its web application development capabilities.