BMSIT & M, Bengaluru -560064

15CSL77 – WEB TECHNOLOGY Lab Manual

Mr. Shankar R

SYLLABUS

PART A

1. Write a JavaScript to design a simple calculator to perform the following operations:
   sum, product, difference and quotient.

2. Write a JavaScript that calculates the squares and cubes of the numbers from 0 to 10 and outputs HTML text that displays the resulting values in an HTML table format.

3. Write a JavaScript code that displays text "TEXT-GROWING" with increasing font size in the interval of 100ms in RED COLOR, when the font size reaches 50pt it displays "TEXT-SHRINKING" in BLUE color. then the font size decreases to 5pt.

4. Develop and demonstrate a HTML5 file that includes JavaScript script that uses functions for the following problems:
   a. Parameter: A string
   b. Output: The position in the string of the left-most vowel
   c. Parameter: A number
   d. Output: The number with its digits in the reverse order

5. Design an XML document to store information about a student in an engineering college affiliated to VTU. The information must include USN, Name, and Name of the College, Branch, Year of Joining, and email id. Make up sample data for 3 students. Create a CSS style sheet and use it to display the document.

6. Write a PHP program to keep track of the number of visitors visiting the web page and to display this count of visitors, with proper headings.

7. Write a PHP program to display a digital clock which displays the current time of the server.

8. Write the PHP programs to do the following:
   a. Implement simple calculator operations.
   b. Find the transpose of a matrix.
   c. Multiplication of two matrices.
   d. Addition of two matrices.

9. Write a PHP program named states.py that declares a variable states with value "Mississippi Alabama Texas Massachusetts Kansas". write a PHP program that does the following:
   a. Search for a word in variable states that ends in xas. Store this word in element 0 of a list named statesList.
   b. Search for a word in states that begins with k and ends in s. Perform a case-insensitive comparison. [Note: Passing re.Ias a second parameter to method compile performs a case-insensitive comparison.] Store this word in element1 of statesList.
   c. Search for a word in states that begins with M and ends in s. Store this word in element 2 of the list.
   d. Search for a word in states that ends in a. Store this word in element 3 of the list.

10. Write a PHP program to sort the student records which are stored in the database using selection sort.

PART –B (MINI-PROJECT)

Develop a web application project using the languages and concepts learnt in the theory and exercises listed in part A with a good look and feel effects. You can use any web technologies and frameworks and databases.

Note:

1. In the examination each student picks one question from part A.
2. A team of two or three students must develop the mini project. However during the examination, each student must demonstrate the project individually.
3. The team must submit a brief project report (15-20 pages) that must include the following
    a. Introduction
    b. Requirement Analysis
    c. Software Requirement Specification
    d. Analysis and Design
    e. Implementation
    f. Testing

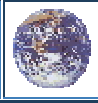
Conduction of Practical Examination:

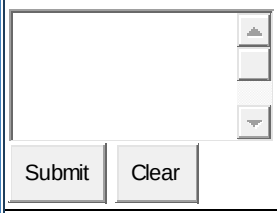
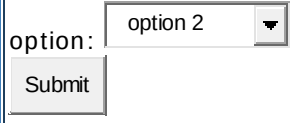
1. All laboratory experiments from part A are to be included for practical examination.
2. Mini project has to be evaluated for 30 Marks as per 6(b).
3. Report should be prepared in a standard format prescribed for project work.
4. Students are allowed to pick one experiment from the lot.
5. Strictly follow the instructions as printed on the cover page of answer script.
6. Marks distribution: a) Part A: Procedure + Conduction + Viva:10 + 35 +5 =50 Marks b) Part B: Demonstration + Report + Viva voce = 15+10+05 = 30 Marks
7. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

HTML Tags Chart

To use any of the following HTML tags, simply select the HTML code you'd like and copy and paste it into your web page.

Tag	Name	Code Example	Browser View
<!--	**comment**	**<!--**This can be viewed in the **HTML** part of a document**-->**	Nothing will show
<a -	**anchor**	** Visit Our Site****	Visit Our Site
	bold	****Example****	**Example**
<big>	**big (text)**	**<big>**Example**</big>**	Example
<body>	**body of HTML document**	**<body>**The content of your **HTML** page**</body>**	Contents of your web page
 	line break	The contents of your page** **The contents of your page	The contents of your web page The contents of your web page
	center	** **This will center your contents** **	This will center your contents
<dd>	**definition description**	<dl> <dt>Definition Term</dt> **<dd>Definition of the term</dd>** <dt>Definition Term</dt> **<dd>Definition of the term</dd>** </dl>	Definition Term **Definition of the term** Definition Term **Definition of the term**
<dl>	**definition list**	**<dl>** <dt>Definition Term</dt> <dd>Definition of the term</dd> <dt>Definition Term</dt> <dd>Definition of the term</dd> **</dl>**	Definition Term Definition of the term Definition Term Definition of the term
<dt>	**definition term**	<dl> **<dt>Definition Term</dt>** <dd>Definition of the term</dd> **<dt>Definition Term</dt>** <dd>Definition of the term</dd> </dl>	**Definition Term** Definition of the term **Definition Term** Definition of the term
	emphasis	This is an ****Example**** of using the emphasis tag	This is an *Example* of using the emphasis tag
<embed>	**embed object**	**<embed** src="yourfile.mid" width="100%" height="60" align="center">	
<embed>	**embed object**	<**embed** src="yourfile.mid" autostart="true" hidden="false" loop="false"> <noembed><bgsound src="yourfile.mid" loop="1"></noembed>	Music will begin playing when your page is loaded and will only play one time. A control panel will be displayed to enable your visitors to stop the music.
	font	**<font** face="Times New Roman">Example****	Example

	font	**<font** face="Times New Roman" size="4">Example****	Example
	font	**<font** face="Times New Roman" size="+3" color="#ff0000">Example****	Example
<form>	**form**	**<form** action="mailto:you@yourdomain.com "> Name: <input name="Name" value="" size="10"> Email: <input name="Email" value="" size="10"> <input type="submit"> **</form>**	Name: Email: Submit
<h1> <h2> <h3> <h4> <h5> <h6>	**heading 1** **heading 2** **heading 3** **heading 4** **heading 5** **heading 6**	** **Heading 1 Example** ** ** **Heading 2 Example** ** ** **Heading 3 Example** ** ** **Heading 4 Example** ** ** **Heading 5 Example** ** ** **Heading 6 Example** **	**Heading 1** **Heading 2** **Heading 3** **Heading 4** **Heading 5** **Heading 6**
<head>	**heading of HTML document**	**<head>**Contains elements describing the document**</head>**	Nothing will show
<hr>	**horizontal rule**	**<hr />**	Contents of your web page Contents of your web page
<hr>	**horizontal rule**	**<hr** width="50%" size="3" />	Contents of your web page Contents of your web page
<hr>	**horizontal rule**	**<hr** width="50%" size="3" noshade />	Contents of your web page Contents of your web page
<hr> (Internet Explorer)	**horizontal rule**	**<hr** width="75%" color="#ff0000" size="4" />	Contents of your web page Contents of your web page
<hr> (Internet Explorer)	**horizontal rule**	**<hr** width="25%" color="#6699ff" size="6" />	Contents of your web page Contents of your web page
<html>	**hypertext markup language**	**<html>** <head> <meta> <title>Title of your web page</title> </head> <body>**HTML web page** contents </body> **</html>**	Contents of your web page
	italic	****Example****	*Example*
<img>	**image**	**<img** src="Earth.gif" width="41" height="41" border="0" alt="text describing the image" />	

<input>	**input field**	Example 1: <form method=post action="/cgi-bin/example.cgi"> **<input** type="text" size="10" maxlength="30"> **<input** type="Submit" value="Submit"> </form>	Example 1: 
<input> (Internet Explorer)	**input field**	Example 2: <form method=post action="/cgi-bin/example.cgi"> **<input** type="text" style="color: #ffffff; font-family: Verdana; font-weight: bold; font-size: 12px; background-color: #72a4d2;" size="10" maxlength="30"> **<input** type="Submit" value="Submit"> </form>	Example 2: 
<input>	**input field**	Example 3: <form method=post action="/cgi-bin/example.cgi"> <table border="0" cellspacing="0" cellpadding="2"><tr><td bgcolor="#8463ff"><input type="text" size="10" maxlength="30"></td><td bgcolor="#8463ff" valign="Middle"> <input type="image" name="submit" src="yourimage.gif"></td></tr> </table> </form>	Example 3: 
<input>	**input field**	Example 4: <form method=post action="/cgi-bin/example.cgi"> Enter Your Comments: <textarea wrap="virtual" name="Comments" rows=3 cols=20 maxlength=100></textarea> **<input** type="Submit" value="Submit"> **<input** type="Reset" value="Clear"> </form>	Example 4: 
<input>	**input field**	Example 5: <form method=post action="/cgi-bin/example.cgi"> Select an option: <select> <option >option 1</option> <option selected>option 2</option> <option>option 3</option> <option>option 4</option> <option>option 5</option> <option>option 6</option> </select> **<input** type="Submit" value="Submit"> </form>	Example 5: Select an option: 
<input>	**input field**	Example 6:	Example 6:

		<form method=post action="/cgi-bin/example.cgi"> Select an option: **<input** type="radio" name="option"> Option 1 **<input** type="radio" name="option" checked> Option 2 **<input** type="radio" name="option"> Option 3 Select an option: **<input** type="checkbox" name="selection"> Selection 1 **<input** type="checkbox" name="selection" checked> Selection 2 **<input** type="checkbox" name="selection"> Selection 3 **<input** type="Submit" value="Submit"> </form>	Select an option: Option 1 Option 2 Option 3 Select an option: Selection 1 Selection 2 Selection 3 Submit
	list item	Example 1: <menu> **<li** type="disc">List item 1** ** **<li** type="circle">List item 2** ** **<li** type="square">List item 3** ** </MENU> Example 2: <ol type="i"> ** **List item 1** ** ** **List item 2** ** ** **List item 3** ** ** **List item 4** ** 	Example 1: • **List item 1** o **List item 2** ▪ **List item 3** Example 2: i. **List item 1** ii. **List item 2** iii. **List item 3** iv. **List item 4**
<link>	**link**	<head> <link rel="stylesheet" type="text/css" href="style.css" /> </head>	
<marquee> (Internet Explorer)	**scrolling text**	**<marquee** bgcolor="#cccccc" loop="-1" scrollamount="2" width="100%">Example Marquee**</marquee>**	
<menu>	**menu**	**<menu>** <li type="disc">List item 1 <li type="circle">List item 2 <li type="square">List item 3 **</menu>**	• List item 1 o List item 2 ▪ List item 3
<meta>	**meta**	**<meta** name="Description" content="Description of your site"> **<meta** name="keywords" content="keywords describing your site">	Nothing will show
<meta>	**meta**	**<meta** HTTP-EQUIV="Refresh" CONTENT="4;URL=http://www.yourdomain.com/">	Nothing will show

<meta>	**meta**	**<meta** http-equiv="Pragma" content="no-cache">	Nothing will show
<meta>	**meta**	**<meta** name="rating" content="General">	Nothing will show
<meta>	**meta**	**<meta** name="robots" content="all">	Nothing will show
<meta>	**meta**	**<meta** name="robots" content="noindex,follow">	Nothing will show
	ordered list	**Numbered** ** ** - List item 1 - List item 2 - List item 3 - List item 4 ** ** **Numbered Special Start** **<ol start="5">** - List item 1 - List item 2 - List item 3 - List item 4 ** ** **Lowercase Letters** **<ol type="a">** - List item 1 - List item 2 - List item 3 - List item 4 ** ** **Capital Letters** **<ol type="A">** - List item 1 - List item 2 - List item 3 - List item 4 ** ** **Capital Letters Special Start** **<ol type="A" start="3">** - List item 1 - List item 2 - List item 3 - List item 4 ** ** **Lowercase Roman Numerals** **<ol type="i">** - List item 1 - List item 2 - List item 3 - List item 4 ** **	**Numbered** 1. List item 1 2. List item 2 3. List item 3 4. List item 4 **Numbered Special Start** 5. List item 1 6. List item 2 7. List item 3 8. List item 4 **Lowercase Letters** a. List item 1 b. List item 2 c. List item 3 d. List item 4 **Capital Letters** A. List item 1 B. List item 2 C. List item 3 D. List item 4 **Capital Letters Special Start** C. List item 1 D. List item 2 E. List item 3 F. List item 4 **Lowercase Roman Numerals** i. List item 1 ii. List item 2 iii. List item 3 iv. List item 4 **Capital Roman Numerals** I. List item 1 II. List item 2 III. List item 3 IV. List item 4

		Capital Roman Numerals **<ol type="I">** - List item 1 - List item 2 - List item 3 - List item 4 ** ** **Capital Roman Numerals Special Start** **<ol type="I" start="7">** - List item 1 - List item 2 - List item 3 - List item 4 ** **	**Capital Roman Numerals Special Start** VII. List item 1 VIII. List item 2 IX. List item 3 X. List item 4
<option>	**listbox option**	<form method=post action="/cgi-bin/example.cgi"> Select an option: <select> **<option>**option 1**</option>** **<option selected>**option 2**</option>** **<option>**option 3**</option>** **<option>**option 4**</option>** **<option>**option 5**</option>** **<option>**option 6**</option>** </select> </form>	Select an option: option 2
<p>	**paragraph**	This is an example displaying the use of the paragraph tag. <p> This will create a line break and a space between lines. Attributes: Example 1: **<p align="left">** This is an example displaying the use of the paragraph tag. Example 2: **<p align="right">** This is an example displaying the use of the paragraph tag. Example 3: **<p align="center">** This is an example displaying the use of the paragraph tag.	This is an example displaying the use of the paragraph tag. This will create a line break and a space between lines. Attributes: Example 1: This is an example displaying the use of the paragraph tag. Example 2: This is an example displaying the use of the paragraph tag. Example 3: This is an example displaying the use of the paragraph tag.
<small>	**small (text)**	****Example****	Example

<table>
<tr><td><strike></td><td>deleted text</td><td><strike>Example</strike></td><td><s>Example</s></td></tr>
<tr><td></td><td>strong emphasis</td><td>Example</td><td>Example</td></tr>
<tr><td><table></td><td>table</td><td>Example 1:

<table border="4" cellpadding="2" cellspacing="2" width="100%">
<tr>
<td>Column 1</td>
<td>Column 2</td>
</tr>
</table>

Example 2: (Internet Explorer)

<table border="2" bordercolor="#336699" cellpadding="2" cellspacing="2" width="100%">
<tr>
<td>Column 1</td>
<td>Column 2</td>
</tr>
</table>

Example 3:

<table cellpadding="2" cellspacing="2" width="100%">
<tr>
<td bgcolor="#cccccc">Column 1</td>
<td bgcolor="#cccccc">Column 2</td>
</tr>
<tr>
<td>Row 2</td>
<td>Row 2</td>
</tr>
</table></td><td>Example 1:
<table><tr><td>Column 1</td><td>Column 2</td></tr></table>
Example 2:
<table><tr><td>Column 1</td><td>Column 2</td></tr></table>
Example 3:
<table><tr><td>Column 1</td><td>Column 2</td></tr><tr><td>Row 2</td><td>Row 2</td></tr></table></td></tr>
<tr><td><td></td><td><table border="2" cellpadding="2" cellspacing="2" width="100%">
<tr>
<td>Column 1</td>
<td>Column 2</td>
</tr>
</table></td><td><table><tr><td>Column 1</td><td>Column 2</td></tr></table></td></tr>
<tr><td><th></td><td>table header</td><td>
<table>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr>
<td>Row 2</td>
<td>Row 2</td>
<td>Row 2</td>
</tr>
<tr>
<td>Row 3</td>
<td>Row 3</td>
<td>Row 3</td>
</tr></td><td><table><tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr><tr><td>Row 2</td><td>Row 2</td><td>Row 2</td></tr><tr><td>Row 3</td><td>Row 3</td><td>Row 3</td></tr><tr><td>Row 4</td><td>Row 4</td><td>Row 4</td></tr></table></td></tr>
</table>

		</tr> <tr> <td>Row 4</td> <td>Row 4</td> <td>Row 4</td> </tr> </table> 	
<title>	**document title**	**<title>**Title of your **HTML page</title>**	Title of your web page will be viewable in the title bar.
<tr>	**table row**	<table border="2" cellpadding="2" cellspacing="2" width="100%"> **<tr>** <td>Column 1</td> <td>Column 2</td> **</tr>** </table>	**Column 1** **Column 2**
<tt>	**teletype**	**<tt>**Example**</tt>**	`Example`
<u>	**underline**	<u>Example</u>	Example
	unordered list	Example 1: ** ** - List item 1 - List item 2 ** ** Example 2: **<ul type="disc">** - List item 1 - List item 2 **<ul type="circle">** - List item 3 - List item 4 ** ** ** **	Example 1: • List item 1 • List item 2 Example 2: • List item 1 • List item 2 o List item 3 o List item 4

HTML 5 NEW TAG

TAG NOT SUPPORTED IN HTML 5

<!--...-->	Define a comment
<!DOCTYPE>	Defines the document type
<a>	Defines a hyperlink href, hreflang, media, ping , rel, target, type
<abbr>	Defines an abbreviation
<acronym>	Used to define an embedded acronyms
<address>	Defines an address element
<applet>	Used to define an embedded applet
<area>	Defines an area inside an image map alt, coords, href, hreflang, media, ping, rel, shape, target, type
<article>	Defines an article cite, pubdate
<aside>	Defines content aside from the page content
<audio>	Defines sound content autobuffer, autoplay, controls, src
<b>	Defines bold text
<base>	Defines a base URL for all the links in a page href, target
<basefont>	Used to define a default font-color, font-size, or font-family for all the document
<bdo>	Defines the direction of text display dir
<big>	Used to make text bigger
<blockquote>	Defines a long quotation cite
<body>	Defines the body element
 	Inserts a single line break
<button>	Defines a push button autofocus, disabled, form, formaction, formenctype, formmethod, formnovalidate, formtarget, name, type, value
<canvas>	Defines graphics height, width
<caption>	Defines a table caption
	Used to center align text and content
<cite>	Defines a citation
<code>	Defines computer code text autobuffer, autoplay, controls, src
<col>	Defines attributes for table columns
<colgroup>	Defines groups of table columns span
<command>	Defines a command button checked, disabled, icon, label, radiogroup, type
<datalist>	Defines a dropdown list
<dd>	Defines a definition description
<del>	Defines deleted text cite, datetime
<details>	Defines details of an element open
<dialog>	Defines a dialog (conversation)
<dfn>	Defines a definition term
<dir>	Used to define a directory list
	Defines a section in a document
<dl>	Defines a definition list
<dt>	Defines a definition term
	Defines emphasized text
<embed>	Defines external interactive content or plugin height, src, type, width
<fieldset>	Defines a fieldset disabled, form, name
<figure>	Defines a group of media content, and their caption
<font>	Used to define font face, font size, and font color of text
<footer>	Defines a footer for a section or page
<form>	Defines a form accept-charset, action, autocomplete, enctype, method, name, novalidate, target
<frame>	Used to define one particular window (frame) within a frameset
<frameset>	Used to define a frameset, which organized multiple windows (frames)
<h1> to <h6>	Defines header 1 to header 6
<head>	Defines information about the document
<header>	Defines a header for a section or page
<hgroup>	Defines information about a section in a document
<hr>	Defines a horizontal rule
<html>	Defines an html document manifest, xmlns
<i>	Defines italic text
<iframe>	Defines an inline sub window height, name, sandbox, seamless, src, width
<img>	Defines an image alt, src, height, ismap, usemap, width
<input>	Defines an input field accept, alt, autocomplete, autofocus, checked, disabled, form, formaction, formenctype, formmethod, formnovalidate, formtarget, height, list, max, maxlength, min, multiple, name, pattern, placeholder, readonly, required, size, src, step, type, value, width
<ins>	Defines inserted text cite, datetime
<keygen>	Defines a generated key in a form autofocus, challenge, disabled, form, keytype, name
<kbd>	Defines keyboard text
<label>	Defines an inline sub window for, form
<legend>	Defines a title in a fieldset
	Defines a list item value
<link>	Defines a resource reference href, hreflang, media, rel, sizes, type
<map>	Defines an image map name
<mark>	Defines marked text
<menu>	Defines a menu list label, type
<meta>	Defines meta information charset, content, http-equiv, name
<meter>	Defines measurement within a predefined range high, low, max, min, optimum, value
<nav>	Defines navigation links
<noframes>	Used to display text for browsers that do not handle frames
<noscript>	Defines a noscript section
<object>	Defines an embedded object data, form, height, name, type, usemap, width
	Defines an ordered list reversed, start
<optgroup>	Defines an option group label, disabled
<option>	Defines an option in a drop-down list disabled, label, selected, value
<output>	Defines some types of output for, form, name
<p>	Defines a paragraph
<param>	Defines a parameter for an object name, value
<pre>	Defines preformatted text
<progress>	Defines progress of a task of any kind max, value
<q>	Defines a short quotation cite
<rp>	Used in ruby annotations to define what to show browsers that to not support the ruby element
<rt>	Defines explanation to ruby annotations
<ruby>	Defines ruby annotations
<s>, <strike>	Used to define strikethrough text.
<samp>	Defines sample computer code
<script>	Defines a definition list async, type charset defer, src
<section>	Defines a section cite
<select>	Defines a selectable list autofocus, disabled, form, multiple, name, size
<small>	Defines small text
<source>	Defines media resources media, src, type
	Defines a section in a document
	Defines strong text
<style>	Defines a style definition type, media, scoped
<sub>, <sup>	Defines sub/super-scripted text
<table>	Defines a table summary
<tbody>	Defines a table body summary
<td>	Defines a table cell colspan, headers, rowspan
<textarea>	Defines a text area autofocus, cols, disabled, form, maxlength, name, placeholder, readonly, readonly, required, rows, wrap
<tfoot>, <thead>	Defines a table footer / head
<th>	Defines a table header colspan, headers, rowspan, scope
<time>	Defines a date/tim datetime
<title>	Defines the document title
<tr>	Defines a table row datetime
<tt>	Used to define teletype text
<u>	Used to define underlined text
	Defines an unordered list
<var>	Defines a variable
<video>	Defines a video autobuffer, autoplay, controls, height, loop, src, width

HTML5 TAG CHEAT SHEET

HTML

Blockquote	<blockquote> </blockquote>	Offsets long quotations
Body	<body> </body>	Identifies the body of an HTML document
Bold		Displays text in boldface
Break (line break)	 	Forces a line break in the document
Button	<button> </button>	Creates a push button
Caption	<caption> </caption>	Adds a table caption
Cite	<cite> </cite>	Identifies a citation or source reference
Code	<code> </code>	Identifies a computer code fragment
Column (table)	<col />	Creates a group of table column attributes
Column group	<colgroup> </colgroup>	Creates a specific column group
Comment	<!-- -->	Adds comments not displayed as part of the page
Definition description	<dd> </dd>	Identifies definition in a glossary or definition list
Defining instance	<dfn> </dfn>	Indicates a defining instance of a given term
Definition list	<dl> </dl>	Creates definition or glossary list
Definition term	<dt> </dt>	Identifies a term to be defined
Deleted text	<del> </del>	Displays strikethrough text
Division		Sets apart element's contents on separate line
Emphasis		Indicates emphasis (usually, displays in italics)
Fieldset	<fieldset></fieldset>	Groups together related controls
Font	 	Modifies font appearance
Form	<form> </form>	Creates an interactive form
Frame	<frame />	Defines a frame
Frameset	<frameset> </frameset>	Defines a group of frames
Head	<head> </head>	Identifies HTML document header
Heading 1–6	, ...	Identifies heading levels 1–6
Horizontal rule	<hr />	Adds a horizontal line
HTML	<html> </html>	Identifies HTML document
Image	<img />	Embeds an image
Inline frame	<iframe> </iframe>	Creates an inline subwindow (floating frame)
Input	<input />	Specifies a form control (user input)
Inserted text	<ins> </ins>	Marks up a document, identifying text inserted in a new version
Italic text		Displays text in italics
Keyboard text	<kbd> </kbd>	Identifies text to be entered by the user
Label	<label> </label>	Assigns labels to form fields
Legend	<legend> </legend>	Sets apart groups of form elements in boxes
Link	<link />	Used in document head to create links to other documents
List item		Identifies an item in an ordered or unordered list
Map	<map> </map>	Identifies a client-side image map
Meta information	<meta />	Contains descriptive information
Noframes	<noframes> </noframes>	Contains alternate content for browsers not supporting frames
Noscript	<noscript> </noscript>	Contains alternate content for browsers not supporting client-side scripts
Object	<object> </object>	Identifies a generic embedded object
Option	<option> </option>	Identifies a selectable choice in a form
Option group	<optgroup> </optgroup>	Creates a logical group of options in a form
Ordered list		Identifies a numbered list
Paragraph		Identifies a paragraph
Parameter	<param />	Contains a set of values required by an object at run-time
Preformatted text	<pre> </pre>	Overrides HTML's text formatting
Quotation	<q> </q>	Identifies a short inline quotation
Sample text	<samp> </samp>	Sample output
Script	<script> </script>	Encloses script statements
Select	<select> </select>	Creates a drop-down menu in a form
Small		Renders text in a "small" font
Span		Contains a customizable inline element
Strong emphasis		Displays text as strongly emphasized (usually, as boldface)
Style	<style> </style>	Includes style information in the document header
Subscript		Renders text as subscript
Superscript		Renders text as superscript
Table	<table> </table>	Identifies a table
Table body	<tbody> </tbody>	Creates a grouping of table body elements
Table data	<td> </td>	Creates a table data cell
Table footer	<tfoot> </tfoot>	Identifies a table footer
Table header	<thead> </thead>	Creates a grouping of rows in a table header
Table header cell	<th> </th>	Identifies a table header cell
Table row	<tr> </tr>	Identifies a row of cells in a table
Textarea	<textarea> </textarea>	Creates a multiple line text field in a form
Title (document)	<title> </title>	Used in document header to assign a title
Teletype text	<tt> </tt>	Renders text in a monospaced style
Unordered list		Identifies an unordered (bulleted) list
Variable	<var> </var>	Indicates a variable or program argument

HTML ATTRIBUTES

ATTRIBUTE NAME	ELEMENTS IT WORKS WITH	VALUE TYPES	FUNCTION
abbr	<td>, <th>	Text	Includes an abbreviation for header cell name
accept	<form>, <input>	Comma-separated list of media types	Provides a list of MIME types for file upload
accept-charset	<form>	Space-separated list of character encodings	Provides a list of supported character sets
accesskey	<a>, <area>, <button>, <input>, <label>, <legend>, <textarea>	Charset (A single character from ISO-10646)	Identifies an accessibility key character
action	<form>	URI (*Uniform Resource Identifier*)	Identifies a server-side form handler
alt	<area>, <img>	Text	Displays a short text description of an image
alt	<input>	CDATA (A sequence of characters from the document's character set. May include entities.)	Displays a short text description of form buttons for non-graphical browsers
archive	<object>	CDATA	Contains a URL where an archive file is located
axis	<td>, <th>	CDATA	Includes a comma-separated list of related headers
bgcolor	<body>, <table>, <td>, <th>, <tr>	Color	Specifies background color.
border	<img>, <table>	Pixels	Controls border width around table
cellpadding	<table>	Length (nn for pixels or nn% for percentage length)	Adds extra space within cells
cellspacing	<table>	Length	Adds extra space between cells
char	<col>, <colgroup>, <tbody>, <td>, <tfoot>, <th>, <thead>, <tr>	Character (A single character from ISO-10646)	Identifies alignment character for table elements. For instance, char=":"
charoff	<col>, <colgroup>, <tbody>, <td>, <tfoot>, <th>, <thead>, <tr>	Length	Specifies offset for alignment character
charset	<a>, <link>, <script>	Charset (A character encoding)	Identifies character encoding of linked resource
checked	<input>	(checked)	Identifies a pre-selected option in checkboxes and radio buttons
cite	<blockquote>, <q>	URI	Identifies URI for source document
cite	<del>, <ins>	URI	Includes info on reason for change
class	All except: <base>, <head>, <html>, <meta>, <param>, <script>, <style>, <title>	CDATA	Identifies a space-separated list of classes in an element
classid	<object>	URI	Contains a URI for an object's implementation
codebase	<object>	URI	Identifies a base URI for classid, data, archive
codetype	<object>	Media type	Identifies content type for code
cols	<frameset>	Multiple lengths	Define columns in a frames page (default 100%)
cols	<textarea>	Number	Defines the width (in characters) of a text area
colspan	<td>, <th>	Number	Specifies the number of table columns spanned by a single cell
content	<meta>	CDATA	Identifies information types included in <meta> element
coords	<a>	Coords	Sets coordinates for clickable area in client-side image maps
coords	<area>	Coords (A comma-separated list of lengths)	Sets the points that define a shape in an image map
data	<object>	URI	Contains a reference to an object's data
datetime	<del>, <ins>	Date and time in ISO date format	Identifies date and time of change
declare	<object>	(declare)	Declares an object without instantiating it
defer	<script>	(defer)	Enables user agent to defer execution of script
dir	All except: <base>, <bdo>, , <frame>, <frameset>, <iframe>, <param>, <script>	(ltr | rtl)	Specifies direction for weak/neutral text

dir	<bdo>	(ltr | rtl)	Specifies directionality of text
disabled	<button>, <input>, <optgroup>, <option>, <select>, <textarea>	(disabled)	Disables form elements
enctype	<form>	Media type	Sets the encoding method for form data
for	<label>	ID reference defined by other attributes	Matches field ID value
frame	<table>	void | above | below| hsides | lhs | rhs | visdes | box | border	Identifies which parts of a table structure should display
frameborder	<frame>, <iframe>	(1 | 0)	Turns frame borders on or off
headers	<td>, <th>	ID reference defined by other attributes	Contains a list of IDs for header cells
height	<iframe>	Length	Specifies frame height
height	<img>, <object>	Length	Specifies an image's or object's height in pixels
href	<a>, <area>, <link>	URI	Identifies a URI for a linked resource
href	<base>	URI	Identifies a URI that acts as a base URI
hreflang	<a>, <link>	A language code	Specifies a language code
http-equiv	<meta>	Name	Identifies an HTTP response header name
id	All except: <base>, <head>, <html>, <meta>, <script>, <style>, <title>	ID	Specifies a document-wide unique ID
ismap	<img>, <input>	(ismap)	Identifies a server-side image map
label	<optgroup>	Text	Identifies a group of options in a form
label	<option>	Text	Specifies an option name in a form
lang	All except: <base>, , <frame>, <frame-set>, <iframe>, <param>, <script>	Language code	Indicates the language being used in an element's contents
longdesc	<frame>, <iframe>	URI	Provides a link to a long description of a frame's contents
longdesc	<img>	URI	Provides a link to a long description for non-graphical browsers
marginheight	<frame>, <iframe>	Pixels	Specifies frame margin height
marginwidth	<frame>, <iframe>	Pixels	Specifies frame margin width
maxlength	<input>	Number	Specifies maximum number of characters for text fields
media	<link>	Single or comma-separated list of media descriptors	Identifies the media a linked style sheet should apply to
media	<style>	Single or comma-separated list of media descriptors	Identifies the media an embedded style sheet should apply to
method	<form>	(get | post)	Specifies the HTTP method used to submit a form
multiple	<select>	(multiple)	Permits multiple selections in a form
name	<a>	CDATA	Creates a named marker in a document
name	<button>, <textarea>	CDATA	Allows a form control to be named
name	<select>	CDATA	Assigns a name to a form field
name	<form>	CDATA	Assigns a name to a form for scripting purposes
name	<frame>, <iframe>	CDATA	Assigns a name to a frame for targeting purposes
name	<img>	CDATA	Assigns a name to an image for scripting purposes
name	<input>, <object>	CDATA	Assigns a name for scripting purposes
name	<map>	CDATA	Assigns a name for reference by usemap
name	<meta>	Name	Assigns a name to meta information
name	<param>	CDATA	Specifies the name of the object property being set
nohref	<area>	(nohref)	Identifies a region that has no action in an image map
noresize	<frame>	(noresize)	Prevents resizing of frames
onblur	<a>, <area>, <button>, <input>, <label>, <select>, <textarea>	Script (A script expression or code segment)	Causes an action when the element loses the focus
onchange	<input>, <select>, <textarea>	Script	Causes an action when the element value is changed
onclick	All except: <base>, <bdo>, , <font>, <frame>, <frameset>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style>, <title>	Script	Causes an action when pointer (mouse) button is clicked
ondblclick	(same as above)	Script	Causes an action when pointer (mouse) button is double clicked
onfocus	<a>, <area>, <button>, <input>, <label>, <select>, <textarea>	Script	Causes an action when an element receives the focus
onkeydown	All except: <base>, <bdo>, , <font>, <frame>, <frameset>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style>, <title>	Script	Causes an action when a key is pressed down
onkeypress	(same as above)	Script	Causes an action when a key is pressed and released
onkeyup	(same as above)	Script	Causes an action when a key is released
onload	<body>	Script	Causes an action when the document has been loaded
onload	<frameset>	Script	Causes an action when all the frames have been loaded
onmouse-down	All except: <base>, <bdo>, , <font>, <frame>, <frameset>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style>, <title>	Script	Causes an action when a pointer or mouse button is pressed down
onmouse-move	(same as above)	Script	Causes an action when a pointer or mouse is moved within the element
onmouseout	All except: <base>, <bdo>, , <font>, <frame>, <frameset>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style>, <title>	Script	Causes an action when a pointer is moved away from the element
onmouseover	(same as above)	Script	Causes an action when a pointer or cursor is moved onto the element
onmouseup	(same as above)	Script	Causes an action when a pointer or mouse button is released
onreset	<form>	Script	Causes an action when a form is reset
onselect	<input>, <textarea>	Script	Causes an action when some text is selected
onsubmit	<form>	Script	Causes an action when a form is submitted
onunload	<body>	Script	Causes an action when the document has been removed
onunload	<frameset>	Script	Causes an action when all the frames have been removed
profile	<head>	URI	Links to a named dictionary of meta information
readonly	<input>, <textarea>	(readonly)	Prevents editing of text fields in a form
rel	<a>, <link>	Link types	Identifies forward link types
rev	<a>, <link>	Link types	Identifies reverse link types
rows	<frameset>	Multiple lengths	Specifies a list of lengths
rows	<textarea>	Number	Sets the number of rows in a text area
rowspan	<td>, <th>	Number	Specifies the number of rows spanned by a single table cell
rules	<table>	none | groups | rows | cols | all	Controls the display of rules within a table
scheme	<meta>	CDATA	Identifies the expected format of the content attribute
scope	<td>, <th>	row | col | rowgroup | colgroup |	Indicates the scope covered by header cells
scrolling	<frame>, <iframe>	(yes | no | auto)	Turns scroll bars on or off in a frame
selected	<option>	(selected)	Defines preselected option in a form
shape	<a>	rect | circle | poly | default	Defines a selectable region in a client-side image map
shape	<area>	rect | circle | poly | default	Controls interpretation of coordinates in an image map
size	<input>	CDATA	Specifies the size (in characters) of a text form control
size	<select>	Number	Specifies the number of rows visible in a drop-down menu
span	<col>	Number	Specifies the number of table columns affected by COL attributes
span	<colgroup>	Number	Specifies the default number of table columns in a column group
src	<frame>, <iframe>	URI	Identifies the source of frame content
src	<input>	URI	Identifies the URI of an image for form fields with images

CONTINUED ON OTHER SIDE

SPARKCHARTS™

HTML

KEY HTML TERMS

- **Hypertext:** A means of connecting documents by text links.
- **HTML:** *Hypertext Markup Language.* A language that uses elements, attributes, and values (*markup*) to construct and link (*hypertext*) documents for easy access and display.
- **XML:** *Extensible Markup Language.* A customizable markup language.
- **XHTML:** *Extensible Hypertext Markup Language.* A reformulation of HTML 4 as an XML 1.0 application.
- **DHTML:** *Dynamic Hypertext Markup Language.* HTML, CSS, and JavaScript combined to create dynamic (as opposed to static) pages.
- **SGML:** *Standard Generalized Markup Language.* The mother language from which HTML, XHTML, and XML were created.
- **Element:** A container that specifies the nature, formatting, or function of a portion of a document. E.g., <form>, <table>, <p>, <img />.
- **Tag:** Signifies the opening and closing of an element. E.g., <table> </table>.
- **Empty element:** An element that has no content and is written as a single tag. E.g., <img />.
- **Block element:** Causes a line break after the element. E.g., <p>, <h1>, <table>,
.
- **Inline element:** Displays *inline* and does not generate a line break. E.g., <img />, <b>, <i>.
- **Proprietary element:** An element not part of the official HTML recommendation. Often, it is supported only by the browser for which it was developed. E.g., Netscape's <blink> element.
- **Deprecated element or attribute:** An element or attribute that is being phased out of HTML and therefore eventually will not be supported by browsers.
- **Attribute:** Lists a characteristic of a particular element. E.g, <table border="3">.
- **Value:** Modifies the attribute in which it occurs. E.g., <table border="3">.
- **Nesting:** The practice of placing elements inside one another (as opposed to overlapping them). Nesting is the correct syntax for HTML. Overlapping is incorrect and can cause problems with your Web page.
  - Correctly nested elements: <tr><td></td></tr>
  - Incorrectly nested (overlapped) elements: <tr><td></tr></td>
- **User agent (UA):** The means (e.g., a Web browser) by which one accesses an HTML document.
- **Client-side:** Describes anything that is done on the user's computer.
- **Server-side:** Describes anything that is done on the Web server.
- **CGI:** *Common Gateway Interface.* A protocol that is used for managing the exchange of information between the user, the server, and the Web site owner.
- **Script:** A portion of programming code that can function in a Web page or on the server, but not as a stand-alone program.
- **Applet:** A small program that can be embedded in a Web page.
- **Well-formed document:** A document that uses correct HTML or XHTML syntax.
- **Valid document:** A document that conforms to a particular DTD (*Document Type Definition*).
- **DTD:** *Document Type Definition.* A standard that identifies the elements, attributes, and values that comprise HTML, XHTML, XML, or any other language created with SGML.
- **W3C:** *The World Wide Web Consortium.* The group responsible for establishing standards for the World Wide Web (www.w3.org).

XHTML DISTINCTIONS

XHTML 1.0 is the current recommendation by the W3C for authoring Web documents. Although it uses the same elements as HTML, there are a number of important distinctions between the two. In particular XHTML is much stricter regarding syntax. The following points are important to remember when writing valid XHTML:

- Documents must be **well-formed** (syntactically correct).
- Element and attribute names must be in **lowercase**.
- **End tags** are required for all **non-empty elements**.
- **Empty elements** should be written in a **minimized form**. E.g.,
.
- Attribute values must always be in **quotation marks**.
- All attributes must have **values**.

SYNTAX

Element with content: <elementName attribute="value">Content</elementName>
Ex: LinkName
Empty element: <elementName attribute="value" />
Ex:

NUMBERED (ORDERED) LISTS

- <ol type="I"> displays capital Roman numerals.
- <ol type="i"> displays small Roman numerals.
- <ol type="A"> displays capital letters.
- <ol type="a"> displays small letters.
- <ol type="1"> displays Arabic numerals (default).

BULLETED (UNORDERED) LISTS

- <ul type="square"> displays a solid, square bullet.
- <ul type="disc"> displays a solid disc (default).
- <ul type="circle"> displays a circle.

CREATING HYPERLINKS

• Text link (internal)
```
<a href="mypage2.htm">Link to a page on my own site</a>
```

• Text link (external)
```
<a href="http://www.othersite.com">Link to someone else's site</a>
```

• Image link (internal)
```
<a href="mypage2.htm"><img src="button.gif" width="50" height="20" alt="Click here" /></a>
```

• Image link (external, with blue border turned off)
```
<a href="http://www.othersite.com"><img src="button2.gif" border="0" width="50" height="20" alt="Other guy's site" /></a>
```

• Email (mailto) link
```
<a href="mailto:name@emailaddress.com">Send me a greeting</a>
```

CREATING TABLES

```
<table width="##" cellpadding="##" cellspacing="##" border="##">
<tr><td>Row 1 Cell 1</td><td>Row 1 Cell 2</td></tr>
<tr><td>Row 2 Cell 1</td><td>Row 2 Cell 2</td></tr>
</table>
```

Use the <table></table> element to create a table, the <tr></tr> element to define rows, and the <td></td> element to create cells. To create a dynamic table that resizes according to the browser window size, use % values in the width and/or height attributes.

CONSTRUCTING FORMS

• Identify where the form information will be sent.
```
<form action="mailto:name@emailaddress.com" method="post">
```
• Create a text box.
```
<p>Name <input type="text" name="name" size="20" maxlength="40" /></p>
```
• Create a text input area.
```
<p>Enter text: <textarea name="response" rows="5" cols="25"> </textarea></p>
```

• Offer a choice of one item with radio buttons.
```
Choice 1: <input type="radio" name="choices" value="choice 1"
  checked="checked" /><br />
Choice 2: <input type="radio" name="choices" value="choice 2" /><br />
Choice 3: <input type="radio" name="choices" value="choice 3" /><br />
```

• Offer multiple choices with check boxes.
```
<input type="checkbox" name="selections" value="selection1" />Selection 1<br />
<input type="checkbox" name="selections" value="selection2" />Selection 2<br />
<input type="checkbox" name="selections" value="selection3" />Selection 3<br />
```

• Use the select and option elements to create a scrolling list box (size attribute value greater than 1) **or a pull-down menu** (size attribute value of 1).
```
<select name="pulldownmenu" size="3">
<option value="firstitem">First Item</option>
<option value="seconditem">Second Item</option>
<option value="thirditem">Third Item</option>
<option value="fourthitem">Fourth Item</option>
</select>
```

• Use the "password" attribute to create a password field.
```
<input type="password" size="15" />
```
• Use the "submit" and "reset" attributes to create submit/reset buttons
```
<input type="submit" /><br />
<input type="reset" />
```

CREATING A FRAMESET PAGE

```
<html>
<head><title>Frameset Page</title></head>
<frameset cols="25%,75%">
     <frame src="document_a.htm" />
     <frame src="document_b.htm" />
</frameset>
</html>
```

Note: For the frameset page to display properly, the corresponding frame documents must also be created (in this case, document_a.htm and document_b.htm).

APPLYING STYLE SHEETS

• Link to an external style sheet:
```
<head>
<link rel="stylesheet" type="text/css" href="stylesheetdoc.css" />
</head>
```
• Embed a style sheet:
```
<head>
<style type="text/css">
  selector {property: value;}
  selector {property: value;}
</style>
</head>
```
• Apply style inline:
```
<p style="property: value; property: value">Formatted text</p>
```

INCLUDING SCRIPTS

```
<script type="text/javascript"> <!--
document.write ("This is fun");
--> </script>
```

The script can be placed in the <head> of the document or in the body. It can also exist as a separate document on the server. To link to an external script, you must add the attribute src="document.js" to the opening script tag.

HTML ELEMENTS

ELEMENT NAME	SYNTAX	FUNCTION
Abbreviation	<abbr> </abbr>	Identifies abbreviated text
Acronym	<acronym> </acronym>	Identifies acronyms
Anchor	<a> 	Creates links
Area	<area />	Defines hot spots in client-side image maps
Attribution information	<address> </address>	Identifies author and contact information
Base URL	<base />	Identifies a base URL for a document
Bidirectional algorithm	<bdo> </bdo>	Overrides the directionality of language characters
Big text	<big> </big>	Makes text one size larger than default

Notes: Deprecated elements and attributes are listed in red. Information was excerpted from the XHTML 1.0, HTML 4.01 (www.w3.org/markup), and CSS 1 (www.w3.org/style/css) recommendations.

HTML ATTRIBUTES (continued)

ATTRIBUTE NAME	ELEMENTS IT WORKS WITH	VALUE TYPES	FUNCTION
src	<img>	URI	Identifies the URI of image to embed
src	<script>	URI	Identifies the URI for an external script
standby	<object>	Text	Contains a message to display while loading an object
style	All except: <base>, <head>, <html>, <meta>, <param>, <script>, <style>, <title>	Style rules	Contains associated style information
summary	<table>	Text	Identifies the purpose/structure of a table for speech output
tabindex	<a>, <area>, <button>, <input>, <object>, <select>, <textarea>	Number	Specifies tabbing order for form fields
target	<a>, <area>, <base>, <form>, <link>	Frame name or _blank, _parent, _self, or _top	Identifies a frame or browser window in which content is to be rendered
title	All except: <base>, <head>, <html>, <meta>, <param>, <script>, <title>	Text	Assigns an advisory title to an element
type	<a>, <link>	Media type (RFC2045)	Identifies advisory content type
type	<button>	button | submit | reset	Defines the function of a form button
type	<input>	text | password | checkbox | radio | submit | reset | file | hidden | image | button	Defines the kind of input that is needed for a form field
type		disc | square | circle or 1 | a | A | i | I	Specifies a list item style
type	<object>	Media type (RFC2045)	Identifies content type for object data
type		1 | a | A | i | I	Specifies numbering style in an ordered list
type	<param>	Media type (RFC2045)	Identifies content type for value when valuetype="ref"
type	<script>	Media type (RFC2045)	Identifies content type of script language
type	<style>	Media type (RFC2045)	Identifies content type of style language
type		disc | square | circle	Specifies bullet style in an unordered list
usemap	<img>, <input>, <object>	URI	Instructs user agent to use a client-side image map
valign	<col>, <colgroup>, <tbody>, <td>, <tfoot>, <th>, <thead>, <tr>	top | middle | bottom | baseline	Specifies vertical alignment in cells
value	<button>	CDATA	Specifies a value sent to the server when clicked
value	<input>	CDATA	Specifies values for radio buttons and checkboxes in forms
value	<option>	CDATA	Specifies a value in a form item
value	<param>	CDATA	Specifies the value of a parameter
valuetype	<param>	data | ref | object	Specifies how a value should be interpreted
width	<col>, <colgroup>	Multiple length	Specifies width
width	<iframe>, <img>, <object>, <table>	Length	Specifies width

CASCADING STYLE SHEETS (CSS) I

KEY CSS TERMS

Selector: Identifies *which portions* of a document will be affected by the style. Frequently a selector is an HTML element name.

Declaration: Identifies *which* style is to be applied to a selector (contains both a property and value).

Property: Identifies which characteristic of an element is to be modified (similar to an HTML attribute).

Value: Identifies the specific style to be applied.

Inheritance: The passing of style from a parent element to its children and grandchildren (descendants).

Descendant: An element that is *nested* inside another element.

Rule: A complete style instruction, including a selector and one or more properties and values.

Text properties: Modify how *text* is displayed.

Font properties: Modify how *fonts* are displayed.

Box properties: Modify how *margins, borders,* and *padding* are displayed.

Background properties: Set *background images* and *background colors.*

Classification properties: Set the *display* and *function* characteristics of elements.

Linked style sheet: A style sheet that exists on the server as a separate text document and is linked to a Web page or pages through the <link> element.

Embedded style sheet: A style sheet that is contained in the <head> portion of an HTML document, inside the <style></style> element.

Inline style sheet: A style sheet that is inserted directly into an HTML element through the use of the style attribute.

Cascade: The order of priority a Web browser gives to style sheets. Generally (though not always), this order is as follows: inline styles take priority over embedded styles, and embedded styles take priority over linked styles.

CSS SYNTAX

- **Style rule:**

```
selector {property: value; property:
value; property: value;}
```

or

```
selector {
        property: value;
        property: value;
        property: value
        }
```

- **Specifying a URL:**

```
url(picture.gif)
```

- **Adding an `!important` declaration:**

```
selector {property: value !important;}
```

(The important declaration will usually prevent a user's styles from overriding your own.)

RELATIVE LENGTH UNITS

- **Ems:** em
- **Exes:** ex
- **Percentage:** %
- **Pixels:** px

ABSOLUTE LENGTH UNITS

- **Centimeters:** cm
- **Inches:** in
- **Millimeters:** mm
- **Picas:** pc
- **Points:** pt

GENERIC FONTS

- **Cursive**
- **Fantasy**
- **Monospace**
- **Sans-serif**
- **Serif**

USEFUL PSEUDO CLASSES

- **a:link**
  Determine link appearance.
- **a:visited**
  Determine appearance of visited links.
- **a:active**
  Determine appearance of active links.
- **a:hover**
  Change appearance of links when a cursor moves over them.

SELECTORS

Selector	Syntax
Type (element)	`body {property: value;}` (multiple selectors may be separated by commas) `h1, h2, h3, h4, h5, h6 {property: value;}`
Descendant (contextual)	Selects an element <h3> that is a descendant of another element . `span h3 {property: value;}`
Class	Selects an element that contains a corresponding class attribute. `.boldface {property: value;}` selects: <p class="boldface">
ID	Selects an element that contains a unique ID attribute. `#243 {property: value;}` selects: <p id="243">
Pseudo-class	Selects an element based on a potential condition that may occur when a page is viewed. `a:visited {color: red;}` sets the color of visited links to red.
Pseudo-element	Selects an element based on a potential situation in which that element occurs. `p:first-line {font-size: 2em}` increases the font size of the first line of every paragraph.

TEXT PROPERTIES

Property	Possible Values
color	[color] | inherit
text-align	left | right | center | justify
text-decoration	none | underline | overline | line-through | blink | inherit
text-indent	[length] | [percentage] | inherit
text-transform	capitalize | uppercase | lowercase | none | inherit
text-shadow	none | [color] | [length] | inherit
letter-spacing	normal | [length] | inherit
word-spacing	normal | [length] | [percentage] | inherit
vertical-align	baseline | sub | super | top | text-top | middle | bottom | text-bottom | [percentage] | [length] | inherit

FONT PROPERTIES

Property	Possible Values
font-family	[font name] | [generic name] | inherit
font-size	xx-small | x-small | small | medium | large | x-large | xx-large | [length] | [percentage] | larger | smaller | [relative measurement] | inherit
font-style	normal | italic | oblique | inherit
font-variant	normal | small-caps | inherit
font-weight	normal | bold | lighter | bolder | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit
font-stretch	ultra-condensed | extra-condensed | condensed | semi-condensed | normal | semi-expanded | expanded | extra-expanded | ultra-expanded | inherit
font-size-adjust	[number] | none | inherit
line-height	normal | [number] | [length] | [percentage] | inherit

BACKGROUND PROPERTIES

Property	Possible Values
background-color	transparent | [color value] | inherit
background-image	none | [url] | inherit
background-repeat	repeat | repeat-x | repeat-y | no-repeat | inherit
background-attachment	scroll | fixed | inherit
background-position	[percentage] | [length{1,2}] | [top | center | bottom] | [left | center | right] | inherit

BOX PROPERTIES

Property	Possible Values
width	[length] | [percentage] | auto | inherit
height	[length] | [percentage] | auto | inherit
clear	none | left | right | both | inherit
float	left | right | none | inherit
border-top-width	[length] | thin | medium | thick | inherit
border-bottom-width	[length] | thin | medium | thick | inherit
border-left-width	[length] | thin | medium | thick | inherit
border-right-width	[length] | thin | medium | thick | inherit
border-width	[length] | thin | medium | thick | inherit
border-color	[color value] | [color name] | transparent | inherit
border-style	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit
margin-top	[length] | [percentage] | auto | inherit
margin-right	[length] | [percentage] | auto | inherit
margin-bottom	[length] | [percentage] | auto | inherit
margin-left	[length] | [percentage] | auto | inherit
padding-top	[length] | [percentage] | inherit
padding-right	[length] | [percentage] | inherit
padding-bottom	[length] | [percentage] | inherit
padding-left	[length] | [percentage] | inherit

CLASSIFICATION PROPERTIES

Property	Possible Values
display	block | inline | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit
list-style-type	disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none | inherit
list-style-image	url | none | inherit
list-style-position	inside | outside | inherit
white-space	normal | pre | nowrap | inherit

SHORTHAND PROPERTIES

Property	Values
background	[background-color || background-image || background-repeat || background-attachment || background-position] | inherit
font	[font-weight] || [font-style] || [font-variant] || [font-size] || [line-height] || [font-family] || caption | icon | menu | message-box | small-caption | status-bar | inherit
border	[border-width] || [border-style] || color | inherit
border-[top, right, bottom, left]	[border-width] || [border-style] || color | inherit
margin	margin-width {1,4} | inherit
padding	padding-width {1,4} | inherit

- When adding values with the border, margin, and padding shorthand properties, the syntax is as follows:
  1. If a single value is supplied, it applies to all four sides.
  2. If two values are supplied, they apply to the top and bottom, respectively.
  3. If three values are supplied, the first and third apply to the top and bottom; the second applies to both the right and left sides.
  4. If four values are supplied, they apply to the top, right, bottom, and left sides respectively.

ADDING COLOR TO WEB PAGES

METHODS FOR CODING COLOR

- **Name (sixteen basic colors only):** white
- **Netscape Named (not all browsers support):** lemonchiffon
- **Hexadecimal:** #ffffff
- **Short hex:** #fff [CSS only]
- **rgb(decimal):** rgb(255, 255, 255) [CSS only]
- **rgb(%):** rgb(100%, 100%, 100%) [CSS only]

APPLYING COLOR WITH HTML (DEPRECATED)

1. **The *bgcolor* attribute controls background color:**
- Accepts name, Netscape name, or hexadecimal code.
- May be used with <body>, <table>, <td>, <th>, and <tr>.
- Syntax:
  <body bgcolor="red"> <table bgcolor="#ff0000">
2. **The *color* attribute controls text color:**
- Accepts name, Netscape name, or hexadecimal code.
- May be used with the <font> and <basefont> elements.
- Syntax:
  <font color="blue"> <basefont color="#0000ff">

APPLYING COLOR WITH CSS

1. **The *background-color* property:**
- Accepts colors by name, hexadecimal code, short hex, rgb(decimal), and rgb(%).
- May be used with any element (selector).
- Syntax:

```
body {background-color: green;}
table {background-color: #00ff00;}
p {background-color: #0f0;}
div {background-color: rgb(0,255,0);}
span {background-color: rgb(0%,100%,0%);}
```

2. **The *color* property:**
- Controls *foreground (usually text) color.*
- Accepts colors by name, hexadecimal code, short hex, rgb(decimal), and rgb(%).
- May be used with any element (selector).
- Syntax:

```
p {color: black;}
td {color: #000000;}
body {color: #000;}
div {color: rgb(0,0,0,);}
span {color: rgb(0%,0%,0%);}
```

THE SIXTEEN BASIC COLORS

See chart in next column.

THE WEB SAFE COLORS

Non-dithering or *Web safe* colors are constructed using combinations of red, green, and blue in amounts of 0%, 20%, 40%, 60%, 80%, 100%. These values must be expressed in hexadecimal if you are using the HTML *bgcolor* attribute, but may be written in short hex, rgb(decimal), and rgb(%), in CSS.

THE WEB SAFE COLOR PERCENTAGES

Percent	Hexadecimal	Short Hex	Numerical Value
0%	00	0	0
20%	33	3	51
40%	66	6	102
60%	99	9	153
80%	CC	C	204
100%	FF	F	255

SPECIAL CHARACTERS (ENTITIES)

DEFINITION

Character entities are case sensitive combinations of letters and numbers that represent a specific character or letter. For example, *®* and *®* both represent the registered trademark symbol, ®. Every character in the ISO 8859 Latin 1 Character Set has a numeric reference like *®*, but only certain characters have character references like *®*.

USE

1. **Inserting special characters:** Entities may be used to insert special characters into your Web page. For example, if you want to represent an amount of money using the Pound (£), you could use either a character entity or numeric entity to represent the pound symbol:
   Item price: £2.5
   or
   Item price: £2.5 .
   In either case, the text will display: **Item price: £2.5.**
2. **Escaping out characters:** Some characters, such as the "less than" and "greater than" signs, can cause problems with your pages if you simply type them in from the keyboard. This is because a browser might confuse those characters with HTML, CSS, or JavaScript code. To avoid this problem, you need to escape out those characters by using entities. For example, if you are designing an HTML tutorial page and wish to represent the <table> tag, you would need to write it either as
   <table>
   or as
   <table>

SYNTAX

- All entities are case-sensitive.
- All entities must begin with an & (ampersand).
- All entities must end with a ; (semicolon).
- Numeric entities must have a # (crosshatch) before the entity number.

LETTERS AND NUMERALS

A	A	a	a	0	0
B	B	b	b	1	1
C	C	c	c	2	2
D	D	d	d	3	3
E	E	e	e	4	4
F	F	f	f	5	5
G	G	g	g	6	6
H	H	h	h	7	7
I	I	i	i	8	8
J	J	j	j	9	9
K	K	k	k		
L	L	l	l		
M	M	m	m		
N	N	n	n		
O	O	o	o		
P	P	p	p		
Q	Q	q	q		
R	R	r	r		
S	S	s	s		
T	T	t	t		
U	U	u	u		
V	V	v	v		
W	W	w	w		
X	X	x	x		
Y	Y	y	y		
Z	Z	z	z		

PUNCTUATION AND SYMBOLS

Normal space	(none)		
Exclamation point	!	!	
Quotation mark	"	"	"
Hash mark	#	#	
Dollar sign	$	$	
Percent sign	%	%	
Ampersand	&	&	&
Apostrophe	'	'	
Parenthesis (left)	(	(	
Parenthesis (right)	)	)	
Asterisk	*	*	
Plus sign	+	+	
Comma	,	,	
Hyphen/minus sign	-	-	
Period	.	.	
Slash	/	/	
Colon	:	:	
Semicolon	;	;	
Less than	<	<	<
Equal sign	=	=	
Greater than	>	>	>
Question mark	?	?	
Commercial "at" sign	@	@	
Square bracket (left)	[	[	
Backslash	\	\	
Square bracket (right)	]	]	
Caret	^	^	
Underscore	_	_	
Grave accent	`	`	
Curly brace (left)	{	{	
Vertical bar	|			
Curly brace (right)	}	}	
Tilde	~	~	
Non-breaking space	(none)		
Inverted exclamation point	¡	¡	¡
Euro sign	€	€	€

Cent sign	¢	¢	¢
Pound sign	£	£	£
Currency sign	¤	¤	¤
Yen sign	¥	¥	¥
Broken vertical bar	¦	¦	¦
Section sign	§	§	§
Umlaut	¨	¨	¨
Copyright symbol	©	©	©
Feminine ordinal	ª	ª	ª
Left angle quote	«	«	«
Logical not sign	¬	¬	¬
Soft hyphen	-		
Registered trademark	®	®	®
Macron accent	¯	¯	¯
Degree sign	°	°	°
Plus or minus	±	±	±
Superscript two	²	²	²
Superscript three	³	³	³
Acute accent	´	´	´
Micro sign	µ	µ	µ
Paragraph sign	¶	¶	¶
Middle dot	·	·	·
Cedilla	¸	¸	¸
Superscript one	¹	¹	¹
Masculine ordinal	º	º	º
Right angle quote	»	»	»
Fraction one-fourth	¼	¼	¼
Fraction one-half	½	½	½
Fraction three-fourths	¾	¾	¾
Inverted question mark	¿	¿	¿
Multiplication sign	×	×	×
Division sign	÷	÷	÷

MISCELLANEOUS SPECIAL CHARACTERS

Capital A (grave accent)	À	À	À
Capital A (acute accent)	Á	Á	Á
Capital A (circumflex accent)	Â	Â	Â
Capital A (tilde)	Ã	Ã	Ã
Capital A (umlaut)	Ä	Ä	Ä
Capital A (ring)	Å	Å	Å
Capital AE (ligature)	Æ	Æ	Æ
Capital C (cedilla)	Ç	Ç	Ç
Capital E (grave accent)	È	È	È
Capital E (acute accent)	É	É	É
Capital E (circumflex accent)	Ê	Ê	Ê
Capital E (umlaut)	Ë	Ë	Ë
Capital I (grave accent)	Ì	Ì	Ì
Capital I (acute accent)	Í	Í	Í
Capital I (circumflex accent)	Î	Î	Î
Capital I (umlaut)	Ï	Ï	Ï
Capital ETH (Icelandic)	Ð	Ð	Ð
Capital N (tilde)	Ñ	Ñ	Ñ
Capital O (grave accent)	Ò	Ò	Ò
Capital O (acute accent)	Ó	Ó	Ó
Capital O (circumflex accent)	Ô	Ô	Ô
Capital O (tilde)	Õ	Õ	Õ
Capital O (umlaut)	Ö	Ö	Ö
Capital O (slash)	Ø	Ø	Ø
Capital U (grave)	Ù	Ù	Ù
Capital U (acute)	Ú	Ú	Ú
Capital U (circumflex)	Û	Û	Û
Capital U (umlaut)	Ü	Ü	Ü
Capital Y (acute)	Ý	Ý	Ý
Capital Thorn (Icelandic)	Þ	Þ	Þ
Small sz (ligature, German)	ß	ß	ß
Small a (grave accent)	à	à	à
Small a (acute accent)	á	á	á
Small a (circumflex accent)	â	â	â
Small a (tilde)	ã	ã	ã
Small a (umlaut)	ä	ä	ä
Small a (ring)	å	å	å
Small ae (ligature)	æ	æ	æ
Small c (cedilla)	ç	ç	ç
Small e (grave accent)	è	è	è
Small e (acute accent)	é	é	é
Small e (circumflex accent)	ê	ê	ê
Small e (umlaut)	ë	ë	ë
Small i (grave accent)	ì	ì	ì
Small i (acute accent)	í	í	í
Small i (circumflex accent)	î	î	î
Small i (umlaut)	ï	ï	ï
Small eth (Icelandic)	ð	ð	ð
Small n (tilde)	ñ	ñ	ñ
Small o (grave accent)	ò	ò	ò
Small o (acute accent)	ó	ó	ó
Small o (circumflex accent)	ô	ô	ô
Small o (tilde)	õ	õ	õ
Small o (umlaut)	ö	ö	ö
Small o (slash)	ø	ø	ø
Small u (grave accent)	ù	ù	ù
Small u (acute accent)	ú	ú	ú
Small u (circumflex accent)	û	û	û
Small u (umlaut)	ü	ü	ü
Small y (acute accent)	ý	ý	ý
Small thorn (Icelandic)	þ	þ	þ
Small y (umlaut)	ÿ	ÿ	ÿ
Left arrow	←	←	←
Up arrow	↑	↑	↑
Right arrow	→	→	→
Down arrow	↓	↓	↓
Horizontal arrow	↔	↔	↔
Spade suit (black)	♠	♠	♠
Club suit (black)	♣	♣	♣
Heart suit (black)	♥	♥	♥
Diamond suit (black)	♦	♦	♦

SPARKCHARTS™
Writer: James Pence
Designer: Dan O. Williams
Series Editors: Sarah Friedberg, Matt Blanchard
Report errors at:
www.sparknotes.com/errors

Essential Javascript -- A Javascript Tutorial

By Patrick Hunlock

Javascript is an interpreted language with a C like syntax. While many people brush the language off as nothing more than a browser scripting language, it actually supports many advanced concepts such as object-oriented-programing, recursion, lambda, and closures. It's a very approachable language for the beginner that quickly scales to be as powerful a tool as your skills allow.

This reference will cover the basic language constructs. This is not a beginner's guide to programming. This article focuses on bringing people who already know another programming language up to speed on Javascript methodology. Additionally, this is not an exhaustive language definition, it is a broad overview that will occasionally focus in on some more advanced concepts. It's here to get you started, other articles will focus on making you an expert.

Getting Started

To dive into Javascript all you need is a simple text-editor and a browser. In windows, you can use notepad under your accessories and Linux and mac users have a similar editor. Simply create a blank HTML page as such…

```
<html>
  <head>
     <title>Learning Javascript</title>
  </head>
  <body>
     <p>Hello World!
  </body>
</html>
```

Save the file then in your browser type in the file name you just created to see the results. Javascript is interpreted so any changes you make to this file will show up instantly in the browser the moment you hit the reload button.

In-Line Javascript

To define a Javascript block in your web page, simply use the following block of HTML.

```
<script type='text/javascript'>
// Your script goes here.
</script>
```

You can place these script blocks anywhere on the page that you wish, there are some rules and conventions however. If you are generating dynamic content as the page loads you will want the script blocks to appear where you want their output to be. For instance, if I wanted to say "Hello World!" I would want my script block to appear in the <body> area of my web page and not in the <head> section.

Unless your scripts are generating output as the page loads, good practice says that you should place your scripts at the very bottom of your HTML. The reason for this is that each time the browser encounters a <script> tag it has to pause, compile the script, execute the script, then continue on generating the page. This takes time so if you can get away with it, make sure the browser hits your scripts at the end of the page instead of the start.

External Javascript

External Javascript is where things get interesting. Any time you have a block of code which you will want to use on several different web pages you should place that block in an external Javascript file. The clock on the upper right-hand corner of this page is a good example. The clock appears on almost every page on this site and so it is included in my "common.js" file. Every web-page on the site will load this file and so the clock is available to all of my web-pages.

There's nothing fancy about an external Javascript file. All it is, is a text file where you've put all your Javascript. Basically everything that would ordinarily go ***between*** the <script> tags can go in your external file. Note that between was stressed, you can not have the <script> </script> tags themselves in your external file or you will get errors.

The biggest advantage to having an external Javascript file is that once the file has been loaded, the script will hang around the browser's cache which means if the Javascript is loaded on one page then it's almost a sure thing that the next page on the site the user visits will be able to load the file from the browser's cache instead of having to reload it over the Internet (This is an incredibly fast and speedy process).

Including an external file is basically the same as doing an in-line script, the only difference is that you specify a filename, and there's no actual code between <script> and </script>...

```
<script type='text/javascript' src='common.js'></script>
```

When the browser encounters this block it will load common.js, evaluate it, and execute it. Like in-line scripts above you can place this block anywhere you need the script to be and like in-line scripts you should place these as close to the bottom of the web-page as you can get away with.

The only difference between in-line Javascript blocks and external Javascript blocks is that an external Javascript block will pause to load the external file. If you discount that one thing, there's no procedural difference between the two!

Javascript is case sensitive.

It should also be noted, before we begin, that Javascript is extremely case sensitive so if you're trying to code along with any examples make sure lowercase is lowercase and uppercase is uppercase. For the most part Javascript is also a camel-cased language. That is, if you're trying to express more than one word you will eliminate the spaces, leave the first letter uncapitalized and capitalize the first letter of each word. Thus "get element by id" becomes "getElementByID".

By contrast, HTML itself is NOT case sensitive.

Output (writeln)

One of the most important things to do when learning a new language is to master basic input and output which is why hello world has become almost a cliché in programming textbooks. For Javascript you need three hello worlds because there are three ways to communicate with the user, each increasingly more useful than the last.

The first method is to use the `document.writeln(string)` command. This can be used while the page is being constructed. After the page has finished loading a new `document.writeln(string)` command will delete the page in most browsers, so use this only while the page is loading. Here's how a simple web-page will look...

```
<html>
  <head>
  </head>
  <body>
     <script type='text/javascript'>
        document.writeln('Hello World!');
     </script>
  </body>
</html>
```

As the page is loading, Javascript will encounter this script and it will output "Hello World!" exactly where the script block appears on the page.

The problem with *writeln* is that if you use this method after the page has loaded the browser will destroy the page and start constructing a new one.

For the most part, document.writeln is useful only when teaching yourself the language. Dynamic content during page load is better served by the server-side scripting languages. That said, *document.writeln* is very useful in pre-processing forms before they're sent to the server -- you can basically create a new web-page on the fly without the need to contact the server.

Output (alert)

The second method is to use a browser alert box. While these are incredibly useful for debugging (and learning the language), they are a horrible way to communicate with the user. Alert boxes will stop your scripts from running until the user clicks the OK button, and it has all the charm and grace of all those pop-up windows everyone spent so many years trying to get rid of!

```
<html>
  <head>
  </head>
  <body>
     <script type='text/javascript'>
        alert('Hello World!');
     </script>
  </body>
</html>
```

Output (getElementById)

The last method is the most powerful and the most complex (but don't worry, it's really easy!).

Everything on a web page resides in a box. A paragraph (<P>) is a box. When you mark something as bold you create a little box around that text that will contain bold text. You can give each and every box in HTML a unique identifier (an ID), and Javascript can find boxes you have labeled and let you manipulate them. Well enough verbiage, check out the code!

```
<html>
  <head>
  </head>
  <body>
     <div id='feedback'></div>
     <script type='text/javascript'>
        document.getElementById('feedback').innerHTML='Hello World!';
     </script>
  </body>
</html>
```

The page is a little bigger now but it's a lot more powerful and scalable than the other two. Here we defined a division and named it "*feedback*". That HTML has a name now, it is unique and that means we can use Javascript to find that block, and modify it. We do exactly this in the script below the division! The left part of the statement says on this web page (*document*) find a block we've named "*feedback*" (*getElementById('feedback')*), and change its HTML (*innerHTML*) to be *'Hello World!'*.

We can change the contents of '*feedback*' at any time, even after the page has finished loading (which document.writeln can't do), and without annoying the user with a bunch of pop-up alert boxes (which alert can't do!).

It should be mentioned that innerHTML is not a published standard. The standards provide ways to do exactly what we did in our example above. That mentioned, innerHTML is supported by every major Browser and in addition innerHTML works faster, and is easier to use and maintain. It's, therefore, not surprising that the vast majority of web pages use innerHTML over the official standards.

While we used "Hello World!" as our first example, its important to note that, with the exception of <script> and <style>, you can use full-blown HTML. Which means instead of just Hello World we could do something like this…

```
<html>
  <head>
  </head>
  <body>
     <div id='feedback'></div>
     <script type='text/javascript'>
        document.getElementById('feedback').innerHTML='<P><font color=red>Hello
World!</font>';
     </script>
  </body>
</html>
```

In this example, *innerHTML* will process your string and basically redraw the web page with the new content. This is a VERY powerful and easy to use concept. It means you can basically take an empty HTML element (which our feedback division is) and suddenly expand it out with as much HTML content as you'd like.

Input (One Click To Rule Them All)

Input, of course, is a little more complicated. For now we'll just reduce it to a bare click of the mouse.

If everything in HTML is a box and every box can be given a name, then every box can be given an event as well and one of those events we can look for is "*onClick*". Lets revisit our last example...

```
<html>
  <head>
  </head>
  <body>
     <div id='feedback' onClick='goodbye()'>Users without Javascript see
this.</div>
     <script type='text/javascript'>
        document.getElementById('feedback').innerHTML='Hello World!';
        function goodbye() {
           document.getElementById('feedback').innerHTML='Goodbye World!';
        }
     </script>
  </body>
</html>
```

Here we did two things to the example, first we added an "*onClick*" event to our feedback division which tells it to execute a function called *goodbye()* when the user clicks on the division. A function is nothing more than a named block of code. In this example goodbye does the exact same thing as our first hello world example, it's just named and inserts 'Goodbye World!' instead of 'Hello World!'.

Another new concept in this example is that we provided some text for people without Javascript to see. As the page loads it will place "Users without Javascript will see this." in the division. If the browser has Javascript, and it's enabled then that text will be immediately overwritten by the first line in the script which looks up the division and inserts "Hello World!", overwriting our initial message. This happens so fast that the process is invisible to the user, they see only the result, not the process. The *goodbye()* function is not executed until it's explicitly called and that only happens when the user clicks on the division.

While Javascript is nearly universal there are people who surf with it deliberately turned off and the search bots (googlebot, yahoo's slurp, etc) also don't process your Javascript, so you may want to make allowances for what people and machines are-not seeing.

Input (User Input)

Clicks are powerful and easy and you can add an onClick event to pretty much any HTML element, but sometimes you need to be able to ask for input from the user and process it. For that you'll need a basic form element and a button…

```
<input id='userInput' size=60> <button onClick='userSubmit()'>Submit</button><BR>
<P><div id='result'></div>
```

Here we create an input field and give it a name of *userInput*. Then we create a HTML button with an *onClick* event that will call the function *userSubmit()*. These are all standard HTML form elements but they're not bound by a <form> tag since we're not going to be submitting this information to a server. Instead, when the user clicks the submit button, the *onClick* event will call the *userSubmit()* function…

```
<script type='text/javascript'>
function userSubmit() {
   var UI=document.getElementById('userInput').value;
   document.getElementById('result').innerHTML='You typed: '+UI;
}
</script>
```

Here we create a variable called *UI* which looks up the input field *userInput*. This lookup is exactly the same as when we looked up our feedback division in the previous example. Since the input field has data, we ask for its *value* and place that value in our *UI* variable. The next line looks up the *result* division and puts our output there. In this case the output will be "You Typed: " followed by whatever the user had typed into the input field.

We don't actually need to have a submit button. If you'd like to process the user input as the user types then simply attach an *onKeyup* event to the input field as such…

```
<input id='userInput' onKeyUp="userSubmit()" size=60><BR>
<P><div id='result'></div>
```

There's no need to modify the *userSubmit()* function. Now whenever a user presses a key while the *userInput* box has the focus, for each keypress, *userSubmit()* will be called, the value of the input box retrieved, and the *result* division updated.

Javascript is an Event Driven Language

As you can tell from the input examples, Javascript is an event driven language which means your scripts react to events you set up. Your code isn't running all the time, it simply waits until an event starts something up! Going into all the Javascript events is beyond the scope of this document but here's a short-list of common events to get you started.

Event	Description
onAbort	An image failed to load.
onBeforeUnload	The user is navigating away from a page.
onBlur	A form field lost the focus (User moved to another field)
onChange	The contents of a field has changed.
onClick	User clicked on this item.
onDblClick	User double-clicked on this item.
onError	An error occurred while loading an image.
onFocus	User just moved into this form element.
onKeyDown	A key was pressed.
onKeyPress	A key was pressed OR released.
onKeyUp	A key was released.
onLoad	This object (iframe, image, script) finished loading.
onMouseDown	A mouse button was pressed.
onMouseMove	The mouse moved.
onMouseOut	A mouse moved off of this element.
onMouseOver	The mouse moved over this element.
onMouseUp	The mouse button was released.
onReset	A form reset button was pressed.
onResize	The window or frame was resized.
onSelect	Text has been selected.
onSubmit	A form's Submit button has been pressed.
onUnload	The user is navigating away from a page.

These events can be attached to most any HTML tag or form element. Of them all *onClick* will probably be what you end up using most often.

Comments

Javascript supports two types of comments. Double-slashes (//) tell javascript to ignore everything to the end of the line. You will see them used most often to describe what is happening on a particular line.

```
var x=5; // Everything from the // to end of line is ignored(*)
var thingamajig=123.45; // 2 times the price of a whatsit.
```

Block quotes begin a comment block with a slash-asterisk (/*) and Javascript will ignore everything from the start of the comment block until it encounters an asterisk-slash (*/). Block quotes are useful for temporally disabling large areas of code, or describing the purpose of a function, or detailing the purpose and providing credits for the script itself.

```
function whirlymajig(jabberwocky) {
   /* Here we take the jabberwocky and insert it in the gire-gimble,
      taking great care to observe the ipsum lorum!   For bor-rath-outgrabe!
      We really should patent this! */

   return (jabberwocky*2);
}
```

You should note that while comments are useful for maintaining the code, they are a liability itself in Javascript since they will be transmitted along with the code to each and every page load, which can create substantial bandwidth penalties and increase the load time of your page for users.

This doesn't mean you shouldn't comment your code, just that once your code is "finished" you should make a backup copy with the comments, then strip out all the comments in the file which is actually sent to the user. You can automate this process with a minimizing application which you can find at http://www.crockford.com/javascript/jsmin.html and an on-line javascript version at http://fmarcia.info/jsmin/test.html .

The result of minimizing your Javascript is a tiny, compact file which is a fraction of the size of the original which will save you bandwidth and provide speedier page-load time for your visitors. However the result is also a very unmaintainable source-code mess which is why you should keep a separate, unminimized (and heavily commented) version of the original file.

(*) Of special consideration, you should note that the browser itself is ALWAYS looking for a </script> tag to mark the end of your Javascript and if it finds that tag, intact, in-one-piece, be it in a string or a comment, it is going to stop processing Javascript at that point and restart-processing HTML.

```
var x=5;

/* The browser will break the Javascript when it sees this </script> tag.
   Everything from tag forward is now being processed as HTML!
   This is a bad thing!  To avoid this you need to avoid using this
   tag anywhere in your Javascript, and if
   you must have it, you should break the string out like this... */

  document.writeln('</scr'+'ipt>');
```

Variables

Javascript is not a strongly typed language which means you rarely have to concern yourself with the type of data a variable is storing, only what the variable is storing and in Javascript, variables can store anything, even functions.

```
var thisIsAString = 'This is a string';
var alsoAString = '25';
var isANumber = 25;
var isEqual = (alsoAString==isANumber);  // This is true, they are both 25.
var isEqual = (alsoAString===isANumber); // False one is a number, the other a string.
var concat=alsoAString + isANumber;      // concat is now 2525
var addition=isANumber + isANumber;      // addition is now 50
var alsoANumber=3.05;     // is equal to 3.05 (usually).
var floatError=0.06+0.01; // is equal to 0.06999999999999999
var anExponent=1.23e+3;   // is equal to 1230
var hexadecimal = 0xff;   // is equal to 255.
var octal = 0377;         // is equal to 255.
var isTrue = true;   // This is a boolean, it can be true or false.
var isFalse= false;  // This is a boolean, it can be true or false
var isArray = [0, 'one', 2, 3, '4', 5]; // This is an array.
var four = isArray[4]; // assign a single array element to a variable.
                       // in this case four = '4'
var isObject = { 'color': 'blue',  // This is a Javascript object
                 'dog': 'bark',
                 'array': [0,1,2,3,4,5],
                 'myfunc': function () { alert('do something!'); }
               }
var dog = isObject.dog;  // dog now stores the string 'bark';
isObject.myfunc(); // creates an alert box with the value "do something!"
var someFunction = function() {
                     return "I am a function!";
                   }
var alsoAFunction = someFunction; //No () so alsoAFunction becomes a function
var result = alsoAFunction(); // alsoAFunction is executed here because ()
                              // executes the function so result stores the
                              // return value of the function which is
                              // "I am a function!"
```

A variable may not be a Javascript reserved word or begin with a number or any symbol other than $ or _. In Internet explorer you should also avoid variable names with the same name as html elements you have named. For instance…

```
var someDiv = document.getElementByID('someDiv');
```

…will cause problems in Internet Explorer because the variable name and the division name are identical.

In recent years a convention has formed around the use of the $ symbol as various libraries like Prototype and JQuery use it to look up a named HTML element. For most purposes if you see $('something') in Javascript you should read that as being *document.getElementById('something')*. This is not standard Javascript, in order for *$('something')* to work, you need to be using a Javascript framework which will define $ as doing something (Like JQuery, Prototype, etc).

The use of a leading underscore (_) is generally useful to indicate a global variable or a variable that has been set outside the current scope.

A final consideration on variables is that functions themselves can be defined like, and act like variables. Once a function has been defined it can be passed to other functions as an argument (A process knows as lambda), or assigned to other variables just like a string, array or any other Javascript object. Generally if you use a function without trailing parenthesis (), the function is treated like a variable and can be passed and assigned. Trailing parenthesis INVOKE the function, executing it and passing back the return value (if any).

Please note that this is a very broad summary overview of Javascript's data types. For more information please see the other articles in this series (listed at the top of the page) which go into exhaustive detail on each Javascript type.

Variable Scope

Variables in Javascript have FUNCTION scope. That is, all variables are global unless they are explicitly defined inside a function and even then child-functions have access to their parent's variables. If a function defines a new variable WITHOUT using the *var* keyword, that variable will be **global** in scope.

```
var global = 'this is global';
function scopeFunction() {
   alsoGlobal = 'This is also global!';
   var notGlobal = 'This is private to scopeFunction!';

   function subFunction() {
      alert(notGlobal); // We can still access notGlobal in this child function.

      stillGlobal = 'No var keyword so this is global!';
      var isPrivate = 'This is private to subFunction!';
   }

   alert(stillGlobal); // This is an error since we haven't executed subfunction
   subFunction();      // execute subfunction
   alert(stillGlobal); // This will output 'No var keyword so this is global!'
   alert(isPrivate);   // This generate an error since isPrivate is private to
                       // subfunction().

   alert(global);      // outputs: 'this is global'
}
alert(global);      // outputs: 'this is global'
alert(alsoGlobal);  // generates an error since we haven't run scopeFunction yet.
scopeFunction();
alert(alsoGlobal);  // outputs: 'This is also global!';
alert(notGlobal);   // generates an error.
```

The concept that a variable will continue to exist, and can be referenced after the function that created it has ceased executing is known as CLOSURE. In the above example, *stillGlobal*, and *alsoGlobal* can be considered closures because they persist after the function that creates them has ceased to operate. You can do some pretty fancy stuff with it later on, but it's not terribly hard to understand once you associate it with creating a global scoped variable inside a function.

Special Keywords

Javascript has a few pre-defined variables with special meaning.

NaN -- Not a Number (Generated when an arithmetic operation returns an invalid result). *NaN* is a weird construct. For one, it is NEVER equal to itself so you can't simply check to see if *3/'dog' == 'NaN'*. You must use the construct *isNaN(3/dog)* to determine if the operation failed. In boolean operations *NaN* evaluates to false, however 0 also evaluates to false so use *isNaN*. Since *NaN* is never equal to itself you can use this simple trick as well:

if (result != result) { alert('Not a Number!'); }

Infinity is a keyword which is returned when an arithmetic operation overflows Javascript's precision which is in the order of 300 digits. You can find the exact minimum and maximum range for your Javascript implementation using *Number.MAX_VALUE* and *Number.MIN_VALUE*.

null is a reserved word that means "empty". When used in boolean operation it evaluates to false.

Javascript supports *true* and *false* as boolean values.

If a variable hasn't been declared or assigned yet (an argument to a function which never received a value, an object property that hasn't been assigned a value) then that variable will be given a special *undefined* value. In boolean operations *undefined* evaluates as *false*. Here's an example…

```
function doAlert(sayThis) {
   if (sayThis===undefined) {   // Check to see if sayThis was passed.
      sayThis='default value';  // It wasn't so give it a default value
   }                            // End check
   alert(sayThis);              // Toss up the alert.
}
```

Arithmetic Operators

There's nothing particularly exotic about the way Javascript does arithmetic.

Operator	Description
+	Addition (also string concatention)
-	Subtration (also unary minus)
*	Multiplication
/	Division
%	Remainder of division (or modulus)
++	pre or post increment
--	pre or post decrement

++ and -- are C style operators their position around the variable they are operating on is important. If the operator appears BEFORE the variable they will be evaluated immediately, if they appear AFTER the variable they will be evaluated only after the entire line has been resolved. For instance…

```
var x = 5;
var y = x++;      // y=5, x=6
var x = 5;
var y = ++x;      // y=6, x=6
```

In the first example y is assigned the value of x (5) and then x is incremented by one because the ++ operator appears AFTER x. In the second example, x is incremented by one first because ++ appears BEFORE the x, so y is given the value 6.

How this works between C and PhP and Javascript can be somewhat different. There was a most excellent discussion about this on reddit if you'd like to delve into those differences.

Additional shorthand operators exist when working on the same variable. For instance…

```
x = x + 5;  // is the same as...
x += 5;
```

Logical and Comparison Operators

Javascript supports equality checking (==) and *identity* checking (===). Equality checks for equality regardless of type. Therefore 25 and '25' will evaluate as true. Identity checking checks not only for equality but type equality as well so 25 and '25' will evaluate as false because, while both are 25, one is a string and the other a number. Note that a single equal sign is an assignment statement! x=5 will assign 5 to x, while x==5 will see if x is equal to 5, and x===5 will check to see if x is identical to 5.

In addition to == and ===, you can check for not equal (!=) and not identical (!==).

Operator	Description
=	Assignment x=5; // assigns 5 to x
==	Equality, is x==5?
===	Identity, is x 5 and a number as well?
!=	Not equal, is x unequal to 5?
!==	Not identical, is x unequal to the Number 5?
!	Not, if not(false) is true.
||	OR, is (x==5) OR (y==5)
&&	And, is (x==5) AND (y==5)
<	Less than. is x less than 5?
<=	Less than or equal. is x less than or equal to 5?
>	Greater than. is x greater than 5?
>=	Greater than or qual. is x greater than or equal to 5?

Conditionals: IF

The *if* statement lets you execute a block of code if some test is passed.

```
var x=5;
if (x==5) {
   alert('x is equal to 5!');
}
```

You can also use an *else* clause to execute code if the test fails.

```
var x=5;
if (x==5) {
   alert('x is equal to 5!');
} else {
   alert('x is not equal to 5!');
}
```

An *elseif* statement also exists which allows for better formating of long conditional tests.

```
var x=5;
if (x==1) {
   alert('x is equal to 1!');
} else if (x==2) {
   alert('x is equal to 2!');
} else if (x==5) {
   alert('x is equal to 5!');
} else {
   alert('x isn't 1, 2 or 5!');
}
```

Conditionals: SWITCH

If you're going to be doing a large number of tests, it makes sense to use a switch statement instead of nested ifs. Switches in javascript are quite powerful, allowing evaluations on both the switch and the case.

```
var x=5;
switch (x) {
   case 1: alert('x is equal to 1!'; break;
   case 2: alert('x is equal to 2!'; break;
   case 5: alert('x is equal to 5!'; break;
   default: alert("x isn't 1, 2 or 5!");
}
```

Note that if you omit the *break* statement that ALL of the code to the end of the switch statement will be executed. So if x is actually equal to 5 and there is no break statement, an alert for "x is equal to 5" will appear as well as an alert for "x isn't 1,2, or 5!".

Sometimes it makes more sense to do the evaluation in the case statement itself. In this case you'd use true, false, or an expression which evaluates to true or false in the switch statement.

```
var x=5;
switch (true) {
   case (x==1): alert('x is equal to 1!'; break;
   case (x==2): alert('x is equal to 2!'; break;
   case (x==5): alert('x is equal to 5!'; break;
   default: alert("x isn't 1, 2 or 5!");
}
```

Conditionals: Shorthand Assignment

Javascript supports more advanced constructs. Often you will see code like the following…

```
function doAddition(firstVar, secondVar) {
   var first = firstVar  ||  5;
   var second= secondVar || 10;
   return first+second;
}
doAddition(12);
```

Here Javascript uses a logical OR (||) to determine if the passed variables actually have a value. In the example we call *doAddition* with a value of 12 but we neglect to pass a second argument. When we create the *first* variable *firstVar* is a non-falsey value (IE it's actually defined) so Javascript assigns firstVar to first. *secondVar* was never passed a value so it is *undefined* which evaluates to *false* so here the variable *second* will be assigned a default value of 10.

You should note that zero evaluates as false so if you pass zero as either *firstVar* or *secondVar* the default values will be assigned and NOT zero. In our example above it is impossible for *first* or *second* to be assigned a zero.

In psuedo code...

```
var someVariable = (assign if this is truthy) || (assign this if first test evaluates false)
```

Conditionals: Ternary Operators

Ternary operators are a shorthand if/else block who's syntax can be a bit confusing when you're dealing with OPC (Other People's Code). The syntax boils down to this.

```
var userName = 'Bob';
var hello = (userName=='Bob') ? 'Hello Bob!' : 'Hello Not Bob!';
```

In this example the statement to be evaluated is *(userName=='Bob')*. The question marks ends the statement and begins the conditionals. If *UserName* is, indeed, Bob then the first block *'Hello Bob!'* will be returned and assigned to our *hello* variable. If *userName* isn't Bob then the second block ('Hello Not Bob!') is returned and assigned to our *hello* variable.

In psudeo code…

```
var someVariable = (condition to test) ? (condition true) : (condition false);
```

The question mark (?) and colon (:) tend to get lost in complex expressions as you can see in this example taken from wikipedia (but which will also work in Javascript if the various variables are assigned...)

```
for (i = 0; i < MAX_PATTERNS; i++)
   c_patterns[i].ShowWindow(m_data.fOn[i] ? SW_SHOW : SW_HIDE);
```

So while quick and efficient, they do tend to reduce the maintainability/readability of the code.

Loops: FOR

The *for* loop follows basic C syntax, consisting of an initialization, an evaluation, and an increment.

```
for (var i=0; (i<5); i++) {
   document.writeln('I is equal to '+i+'<br>');
}
// outputs:
// I is equal to 0
// I is equal to 1
// I is equal to 2
// I is equal to 3
// I is equal to 4
```

This is actually an extreme simplification of what a for statement can do. On the other end of the spectrum, consider this shuffle prototype which will randomly shuffle the contents of an array. Here, everything is defined, and executed within the context of the for statement itself, needing no additional block to handle the code.

```
Array.prototype.shuffle = function (){
   for(var rnd, tmp, i=this.length; i; rnd=parseInt(Math.random()*i), tmp=this[--
i], this[i]=this[rnd], this[rnd]=tmp);
};
```

Loops: FOR/IN

Javascript has a variant of the for loop when dealing with Javascript objects.

Consider the following object...

```
var myObject = { 'animal'  : 'dog',
                 'growls'  : true,
                 'hasFleas': true,
                 'loyal'   : true }
```

We can loop through these values with the following construct.

```
var myObject = { 'animal'  : 'dog',
                 'growls'  : true,
                 'hasFleas': true,
                 'loyal'   : true }

for (var property in myObject) {
   document.writeln(property + ' contains ' + myObject[property]+'<br>');
}
// Outputs:
// animal contains dog
// growls contains true
// hasFleas contains true
// loyal contains true
```

What this essentially does is assign the property name to the variable *property*. We can then access *myObject* through an associative array style syntax. For instance the first itteration of the loop assigns *animal* to *property* and *myObject["animal"]* will return dog.

There is a big caveat here in that properties and methods added by prototyping will also show up in these types of loops. Therefore it's best to always check to make sure you are dealing with data and not a function as such...

```
for (var property in myObject) {
   if (typeof(myObject[property]) != 'function') {
      document.writeln(property + ' contains ' + myObject[property]+'<br>');
   }
}
```

The typeof check to screen out functions will ensure that your for/in loops will extract only data and not methods that may be added by popular javascript libraries like Prototype.

Loops: WHILE

while loops in Javascript also follow basic C syntax and are easy to understand and use.

The *while* loop will continue to execute until its test condition evaluates to false or the loop encounters a *break* statement.

```
var x = 1;
while (x<5) {
   x = x +1;
}
var x = 1;
while (true) {
   x = x + 1;
   if (x>=5) {
      break;
   }
}
```

Sometimes it makes more sense to evaluate the test condition at the end of the loop instead of the beginning. So for this Javascript supports a *do/while* structure.

```
var x=1;
do {
   x = x + 1;
} while (x < 5);
```

Bringing It All Together

So now you know how to do basic input and output, how to do conditional branching and how to do loops. That's pretty much everything you need to know about any programming language to get started! So lets wrap everything we've done so far into one simple web application.

First we'll define our web page.

```
<html>
   <head>
      <title>My First Javascript</title>
   </head>
   <body>
      Hello World!
   </body>
</html>
```

Now to this we will add an input line, a drop down box, and a submit button. Notice we give an ID to the form elements, and have the submit button call a function when it's clicked. A function which we will write later. We've also added some descriptive text.

```
<html>
   <head>
      <title>My First Javascript</title>
   </head>
   <body>
      Hello World!
      <p>Say what? <input id="sayThis" size=40>
      <P>How many times? <select id='howMany'>
                            <option value=1>1</option>
                            <option value=5 selected>5</option>
                            <option value=10>10</option>
                            <option value=20>20</option>
                         </select>
      <p><button onClick='doLoop()'>Do It!</button>
   </body>
</html>
```

Now we need to add a division where the output will occur. We'll add this right below the form elements, specifically the "Do It!" button.

```
<html>
   <head>
      <title>My First Javascript</title>
   </head>
   <body>
      Hello World!
      <p>Say what? <input id="sayThis" size=40>
      <P>How many times? <select id='howMany'>
                            <option value=1>1</option>
                            <option value=5 selected>5</option>
                            <option value=10>10</option>
                            <option value=20>20</option>
                         </select>
      <p><button onClick='doLoop()'>Do It!</button>
      <p><div id="results"></div>
   </body>
</html>
```

The script for this will be simple enough. When the button is clicked the *doLoop()* function will be called. doLoop() will lookup the value of the *sayThis* input field, then repeat that *howMany* times.

Here's the script itself, separate from the HTML...

```
  function doLoop() {
      var sayWhat = document.getElementById('sayThis').value;
      var maxLoop = document.getElementById('howMany').value;
      var str = ''; // where we'll store our output temporarily.
      for (var i=1; (i<=maxLoop); i++) {
         str=str+i+':'+sayWhat+'<br>';
      }
      document.getElementById("results").innerHTML=str;
   }
```

And the entire application, all together.

```
<html>
   <head>
      <title>My First Javascript</title>
   </head>
   <body>
      Hello World!
      <p>Say what? <input id="sayThis" size=40>
      <P>How many times? <select id='howMany'>
                           <option value=1>1</option>
                           <option value=5 selected>5</option>
                           <option value=10>10</option>
                           <option value=20>20</option>
                        </select>
      <p><button onClick='doLoop()'>Do It!</button>
      <p><div id="results"></div>
      <script type='text/html'>
         function doLoop() {
            var sayWhat = document.getElementById('sayThis').value;
            var maxLoop = document.getElementById('howMany').value;
            var str = ''; // where we'll store our output temporarily.
            for (var i=1; (i<=maxLoop); i++) {
               str=str+i+':'+sayWhat+'<br>';
            }
            document.getElementById("results").innerHTML=str;
         }
      </script>
   </body>
</html>
```

DHTML: Dynamic HTML

The above example is what's technically considered Dynamic HTML (or DHTML) because the contents of the page dynamically change based on user interaction after the page has finished loading. Going into great detail on DHTML is beyond the scope of this article but there is one small, trivial jump that can really change how you see HTML and Javascript.

This page has pulled an HTML element with the use of document.getElementById. Once we've had this assigned to a variable we've been able to manipulate the contents of that element with innerHTML, and in the case of forms find the value of the element with the value property.

There is also a style property which lets you access, and change, the CSS styles for that element. There is a nice CSS reference at IloveJackDaniels (http://www.ilovejackdaniels.com/cheat-sheets/css-cheat-sheet/). Most all of the properties you can access are listed on the left and right margin of the cheat sheet (the interior/brown areas aren't applicable). The only real trick is that where you see a dash in the CSS name, for Javascript you need to remove the dash and capitalize the next letter. So the CSS style *background-color* becomes *backgroundColor* in Javascript.

Here's an example which will change the background color of a division when you click on it's contents.

```
<div id="example" onClick="colorize()">Click on this text to change the
background color</div>
<script type='text/javascript'>
   function colorize() {
      var element = document.getElementById("example");
      element.style.backgroundColor='#800';
   }
</script>
```

Here we create a division with a name of *example* which will call a function called *colorize()* when that division is clicked. *colorize()* will lookup the *example* division and assign it to the Javascript variable *element*. Next, we assign a color of #800 (burgundy) to the element's *style.backgroundColor* property.

This actually made the text hard to read so in the next example we'll add another line to change the color of the text to white.

```
<div id="example" onClick="colorize()">Click on this text to change the
background color</div>
<script type='text/javascript'>
   function colorize() {
      var element = document.getElementById("example");
      element.style.backgroundColor='#800';
      element.style.color='white';
   }
</script>
```

And lets go ahead and center the text as well.

```
<div id="example" onClick="colorize()">Click on this text to change the
background color</div>
<script type='text/javascript'>
   function colorize() {
      var element = document.getElementById("example");
      element.style.backgroundColor='#800';
      element.style.color='white';
      element.style.textAlign='center';
   }
</script>
```

The manipulation of an HTML element's CSS styles is incredibly powerful! You can hide, or display elements with the style.display property (*style.display=none*=invisible, *style.display=block*=visible) You can let the elements float over the page with *style.position* (absolute/relative/fixed), you can change their position on the page by setting *style.left* and *style.top*. The possibilities are literally endless.

Conclusion

As stated at the beginning of this article, this document is intended to bring a programmer up to speed in Javascript, and is not an exhaustive review of the language. This article is a part of a larger series of reference articles (listed at the top of the page). As you master the fundamentals introduced on this page, it's recommended that you visit the other reference articles to master the details of the language.

Javascript is a surprisingly deep and well considered language (with a few notable and notorious exceptions). It is simple enough for beginners to programming to understand and scales to become as advanced as you need it to be. While it may be frustrating to work around the various idiosyncrasies of all the browser models and versions, Javascript itself is a joy to work in, and when developing web-applications, the least and most unobtrusive of all the problems you will have to surmount.

Additional Resources

While there are many good Javascript books available. The general consensus is that the best introduction and reference to Javascript is Javascript: The Definitive Guide by David Flanagan. This book is endorsed by the Usenet group comp.lang.javascript as well as by Douglas Crockford and other notable people of influence within the Javascript community.

Version

This document is current as of Firefox 2.03, IE 7.0, JSCRIPT 5.6, ECMA-262 Edition 3, Javascript 1.7

License

PHP Tutorial From beginner to master

PHP is a powerful tool for making dynamic and interactive Web pages.

PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

In our PHP tutorial you will learn about PHP, and how to execute scripts on your server

Pre-requisites

Before you continue you should have a basic understanding of the following:

- HTML/XHTML
- JavaScript

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms

- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support

PHP Installation

What do you need?

Most people would prefer to install a all-in-one solution:

WampServer 2.0i [07/11/09] → **for Windows platform**
Includes :
- Apache 2.2.11
- MySQL 5.1.36
- PHP 5.3.0

http://www.wampserver.com/en/

http://lamphowto.com/ → **for Linux platform**

Already have a web server?

If your server supports PHP you don't need to do anything.

Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support. However, if your server does not

support PHP, you must install PHP. Here is a link to a good tutorial from PHP.net on how to install PHP5: http://www.php.net/manual/en/install.php

Download PHP

Download PHP for free here: http://www.php.net/downloads.php

Download MySQL Database

Download MySQL for free here: http://www.mysql.com/downloads/index.html

Download Apache Server

Download Apache for free here: http://httpd.apache.org/download.cgi

Download a nice text editor [Not required]

http://www.flos-freeware.ch/notepad2.html

PHP Syntax

PHP code is executed on the server, and the plain HTML result is sent to the browser.

Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with <? and end with ?>.

For maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

```
<?php
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

Note: The file must have a .php extension. If the file has a .html extension, the PHP code will not be executed.

Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

PHP Variables

A variable is used to store information.

Variables in PHP

Variables are used for storing a values, like text strings, numbers or arrays.

When a variable is declared, it can be used over and over again in your script.

All variables in PHP start with a $ sign symbol.

The correct way of declaring a variable in PHP:

```
$var_name = value;
```

New PHP programmers often forget the $ sign at the beginning of the variable. In that case it will not work.

Let's try creating a variable containing a string, and a variable containing a number:

```
<?php
$txt="Hello World!";
$x=16;
?>
```

PHP is a Loosely Typed Language

In PHP, a variable does not need to be declared before adding a value to it.

In the example above, you see that you do not have to tell PHP which data type the variable is.

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

In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

In PHP, the variable is declared automatically when you use it.

Naming Rules for Variables

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)

- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string), or with capitalization ($myString)

PHP String Variables

A string variable is used to store and manipulate text.

String Variables in PHP

String variables are used for values that contains characters.

In this chapter we are going to look at the most common functions and operators used to manipulate strings in PHP.

After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.

Below, the PHP script assigns the text "Hello World" to a string variable called $txt:

```
<?php
$txt="Hello World";
echo $txt;
?>
```

The output of the code above will be:

```
Hello World
```

Now, lets try to use some different functions and operators to manipulate the string.

The Concatenation Operator

There is only one string operator in PHP.

The concatenation operator (.) is used to put two string values together.

To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be:

```
Hello World! What a nice day!
```

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

The strlen() function

The strlen() function is used to return the length of a string.

Let's find the length of a string:

```
<?php
echo strlen("Hello world!");
?>
```

The output of the code above will be:

```
12
```

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string).

The strpos() function

The strpos() function is used to search for character within a string.

If a match is found, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string:

```
<?php
echo strpos("Hello world!","world");
?>
```

The output of the code above will be:

```
6
```

The position of the string "world" in our string is position 6. The reason that it is 6 (and not 7), is that the first position in the string is 0, and not 1.

Complete PHP String Reference

For a complete reference of all string functions, go to our complete PHP String Reference.

The reference contains a brief description, and examples of use, for each function!

PHP Operators

Operators are used to operate on values.

PHP Operators

This section lists the different operators used in PHP.

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y

+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Comparison Operators

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

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

PHP If...Else Statements

Conditional statements are used to perform different actions based on different conditions.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

The if Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
if (condition) code to be executed if condition is true;
```

The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code **only if the specified condition is true**.

The if...else Statement

Use the if....else statement to execute some code if a condition is true and another code if a condition is false.

Syntax

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>

</body>
</html>
```

The if...elseif....else Statement

Use the if....elseif...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition)
  code to be executed if condition is true;
elseif (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

PHP Switch Statement

Conditional statements are used to perform different actions based on different conditions.

The PHP Switch Statement

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

Syntax

```
switch (n)
{
case label1:
```

```
  code to be executed if n=label1;
  break;
case label2:
  code to be executed if n=label2;
  break;
default:
  code to be executed if n is different from both label1 and label2;
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The default statement is used if no match is found.

Example

```
<html>
<body>

<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>

</body>
</html>
```

PHP Arrays

An array stores multiple values in one single variable.

What is an Array?

A variable is a storage area holding a number or text. The problem is, a variable will hold only one value.

An array is a special variable, which can store multiple values in one single variable.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own index so that it can be easily accessed.

In PHP, there are three kind of arrays:

- **Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

---

Numeric Arrays

A numeric array stores each array element with a numeric index.

There are two methods to create a numeric array.

1. In the following example the index are automatically assigned (the index starts at 0):

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

2. In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Example

In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

```
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

The code above will output:

```
Saab and Volvo are Swedish cars.
```

Associative Arrays

An associative array, each ID key is associated with a value.

When storing data about specific named values, a numerical array is not always the best way to do it.

With associative arrays we can use the values as keys and assign values to them.

Example 1

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Example 2

This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```

Multidimensional Arrays

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

Example

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );
```

The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
  (
  [0] => Peter
  [1] => Lois
  [2] => Megan
  )
[Quagmire] => Array
  (
  [0] => Glenn
  )
[Brown] => Array
  (
  [0] => Cleveland
  [1] => Loretta
  [2] => Junior
  )
)
```

Example 2

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

Complete PHP Array Reference

For a complete reference of all array functions, go to our complete PHP Array Reference.

The reference contains a brief description, and examples of use, for each function!

PHP Looping - While Loops

Loops execute a block of code a specified number of times, or while a specified condition is true.

PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In PHP, we have the following looping statements:

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

The while loop executes a block of code while a condition is true.

Syntax

```
while (condition)
  {
```

```
  code to be executed;
  }
```

Example

The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

The do...while Statement

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

Syntax

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

Example

The example below defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

```
<html>
<body>

<?php
$i=1;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=5);
?>

</body>
</html>
```

Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

The for loop and the foreach loop will be explained in the next section

PHP Looping - For Loops

Loops execute a block of code a specified number of times, or while a specified condition is true.

The for Loop

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

Syntax

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

Parameters:

- *init*: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment*: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

Note: Each of the parameters above can be empty, or have multiple expressions (separated by commas).

Example

The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

The foreach Loop

The foreach loop is used to loop through arrays.

Syntax

```
foreach ($array as $value)
  {
  code to be executed;
  }
```

For every loop iteration, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.

Example

The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>

<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>

</body>
</html>
```

Output:

```
one
two
three
```

PHP Functions

The real power of PHP comes from its functions.

In PHP, there are more than 700 built-in functions.

PHP Built-in Functions

For a complete reference and examples of the built-in functions, please visit our PHP Reference.

PHP Functions

In this chapter we will show you how to create your own functions.

To keep the browser from executing a script when the page loads, you can put your script into a function.

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

You may call a function from anywhere within a page.

Create a PHP Function

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

Syntax

```
function functionName()
{
code to be executed;
}
```

PHP function guidelines:

- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)

Example

A simple function that writes my name when it is called:

```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes
```

PHP Functions - Adding parameters

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

Parameters are specified after the function name, inside the parentheses.

Example 1

The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

Example 2

The following function has two parameters:

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>
```

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

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```

PHP Functions - Return values

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

Example

```
<html>
<body>

<?php
function add($x,$y)
{
$total=$x+$y;
return $total;
}

echo "1 + 16 = " . add(1,16);
?>

</body>
</html>
```

Output:

```
1 + 16 = 17
```

PHP Forms and User Input

The PHP $_GET and $_POST variables are used to retrieve information from forms, like user input.

PHP Form Handling

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will **automatically** be available to your PHP scripts.

Example

The example below contains an HTML form with two input fields and a submit button:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>
</html>
```

When a user fills out the form above and click on the submit button, the form data is sent to a PHP file, called "welcome.php":

"welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br />
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

Output could be something like this:

```
Welcome John!
You are 28 years old.
```

The PHP $_GET and $_POST functions will be explained in the next chapters.

Form Validation

User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and reduces the server load.

You should consider server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page. The

user will then get the error messages on the same page as the form. This makes it easier to discover the error.

The built-in $_GET function is used to collect values in a form with method="get".

The $_GET Function

The built-in $_GET function is used to collect values from a form sent with method="get".

Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send (max. 100 characters).

Example

```
<form action="welcome.php" method="get">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

When the user clicks the "Submit" button, the URL sent to the server could look something like this:

```
http://www.w3schools.com/welcome.php?fname=Peter&age=37
```

The "welcome.php" file can now use the $_GET function to collect form data (the names of the form fields will automatically be the keys in the $_GET array):

```
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

When to use method="get"?

When using method="get" in HTML forms, all variable names and values are displayed in the URL.

Note: This method should not be used when sending passwords or other sensitive information!

However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

Note: The get method is not suitable for large variable values; the value cannot exceed 100 characters.

The built-in $_POST function is used to collect values in a form with method="post".

The $_POST Function

The built-in $_POST function is used to collect values from a form sent with method="post".

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

Note: However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file).

Example

```
<form action="welcome.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

When the user clicks the "Submit" button, the URL will look like this:

```
http://www.w3schools.com/welcome.php
```

The "welcome.php" file can now use the $_POST function to collect form data (the names of the form fields will automatically be the keys in the $_POST array):

```
Welcome <?php echo $_POST["fname"]; ?>!<br />
You are <?php echo $_POST["age"]; ?> years old.
```

When to use method="post"?

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

The PHP $_REQUEST Function

The PHP built-in $_REQUEST function contains the contents of both $_GET, $_POST, and $_COOKIE.

The $_REQUEST function can be used to collect form data sent with both the GET and POST methods.

Example

```
Welcome <?php echo $_REQUEST["fname"]; ?>!<br />
You are <?php echo $_REQUEST["age"]; ?> years old.
```

More on Php Forms

http://myphpform.com/php-form-tutorial.php

09/03/18 03:28:10 C:\xampp\htdocs\shan\1.html

```
    <!-- 1. Write a JavaScript to design a simple calculator to perform the following operations:
sum, product, difference and quotient. -->
<!DOCTYPE HTML>
<html>
    <head>
        <style>
            table, td, th
            {
                border: 1px solid black;
                width: 33%;
                text-align: center;
                background-color: DarkGray;
            }
            table { margin: auto; }
            input { text-align:right; }
        </style>
        <script type="text/javascript">

            function calc(clicked_id)
            {
                var val1 = parseFloat(document.getElementById("value1").value);
                var val2 = parseFloat(document.getElementById("value2").value);

                if(isNaN(val1)||isNaN(val2))
                    alert("ENTER VALID NUMBER");

                else if(clicked_id=="add")
                    document.getElementById("answer").value=val1+val2;

                else if(clicked_id=="sub")
                    document.getElementById("answer").value=val1-val2;

                else if(clicked_id=="mul")
                    document.getElementById("answer").value=val1*val2;

                else if(clicked_id=="div")
                    document.getElementById("answer").value=val1/val2;

            }
```

```
40              function cls()
41              {
42                  value1.value="0";
43                  value2.value="0";
44                  answer.value="";
45              }
46          </script>
47      </head>
48
49      <body>
50          <table>
51              <tr><th colspan="4"> SIMPLE CALCULATOR </th></tr>
52
53              <tr><td>value1</td><td><input type="text" id="value1" value="0"/></td>
54                  <td>value2</td><td><input type="text" id="value2" value="0"/> </td></tr>
55
56              <tr><td><input type="button" value="Addition" id = "add" onclick="calc(this.id)"/></td>
57                  <td><input type="button" value="Subtraction" id = "sub" onclick="calc(this.id)"/></td>
58                  <td><input type="button" value="Multiplication" id = "mul" onclick="calc(this.id)"/></td>
59                  <td><input type="button" value="Division" id ="div" onclick="calc(this.id)"/></td></tr>
60              <tr><td>Answer:</td><td> <input type="text" id="answer" value="" disabled/></td>
61                  <td colspan="2"><input type="button" value="CLEAR ALL" onclick="cls()"/></td></tr>
62          </table>
63      </body>
64  </html>
65
```

SIMPLE CALCULATOR			
value1	23	value2	11
Addition	Subtraction	Multiplication	Division
Answer:	34	CLEAR ALL	

09/03/18 04:04:55 C:\xampp\htdocs\shan\2.html

```
<!-- 2. Write a JavaScript that calculates the squares and cubes of the numbers from 0 to 10
and outputs HTML text that displays the resulting values in an HTML table format. -->


<!DOCTYPE HTML>
<html>
    <head>
        <style>
            table,tr, td
            {
                border: solid black;
                width: 33%;
                text-align: center;
                background-color:lightblue;
            }
            table { margin: auto; }
        </style>

        <script>
            document.write( "<table> <tr><th colspan='3'> NUMBERS FROM 0 TO 10 WITH THEIR SQUARES AND
CUBES </th></tr>");
            document.write( "<tr> <td>Number</td> <td>Square</td> <td>Cube</td> </tr>" );
            for(var n=0; n<=10; n++)
            {
                document.write( "<tr><td>" + n + "</td><td>" + n*n + "</td><td>" + n*n*n + "</td></tr>");
            }
            document.write( "</table>" ) ;
        </script>
    </head>
</html>

```

NUMBERS FROM 0 TO 10 WITH THEIR SQUARES AND CUBES		
Number	Square	Cube
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

09/03/18 04:01:15 C:\xampp\htdocs\shan\3.html

```
<!-- 3. Write a JavaScript code that displays text “TEXT-GROWING” with increasing font
size in the interval of 100ms in RED COLOR, when the font size reaches 50pt it
displays “TEXT-SHRINKING” in BLUE color. Then the font size decreases to 5pt. -->

<!DOCTYPE HTML>
<html>
    <head>
        <style>
            p {
                position: absolute;
                top: 50%;
                left: 50%;
                transform: translate(-50%, -50%);
            }
        </style>
    </head>
    <body>
        <p id="demo"></p>
        <script>
            var grow = setInterval(growing, 1000);
            var font_size = 5;
            var p =  document.getElementById("demo");

            function growing()
            {
                p.innerHTML = 'TEXT GROWING';
                p.setAttribute('style', "font-size: " + font_size + "px; color: red");
                font_size += 5;
                if(font_size >= 50 )
                {
                    clearInterval(grow);
                    shrink = setInterval(shrinking, 1000);
                }
            }

            function shrinking()
            {
                font_size -= 5;
                p.innerHTML = 'TEXT SHRINKING';
```

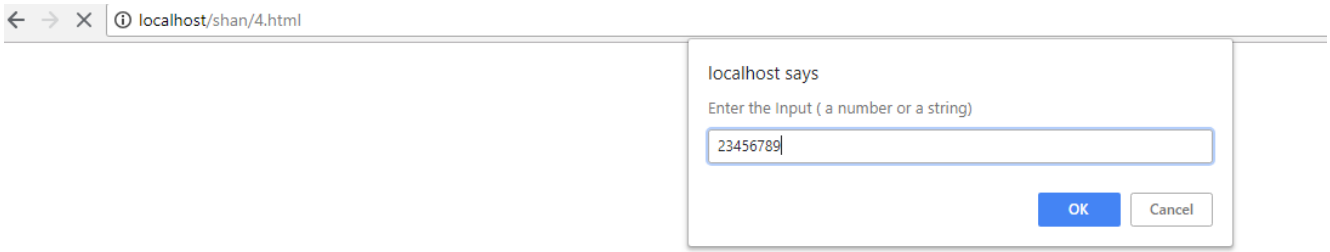
```
                p.setAttribute('style', "font-size: " + font_size + "px; color: blue");
                if(font_size === 5 )
                {
                    clearInterval(shrink);
                }
            }
        </script>
    </body>
</html>
```

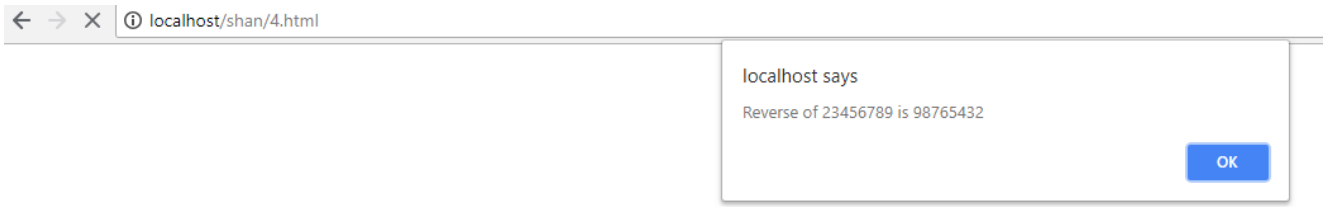
TEXT SHRINKING

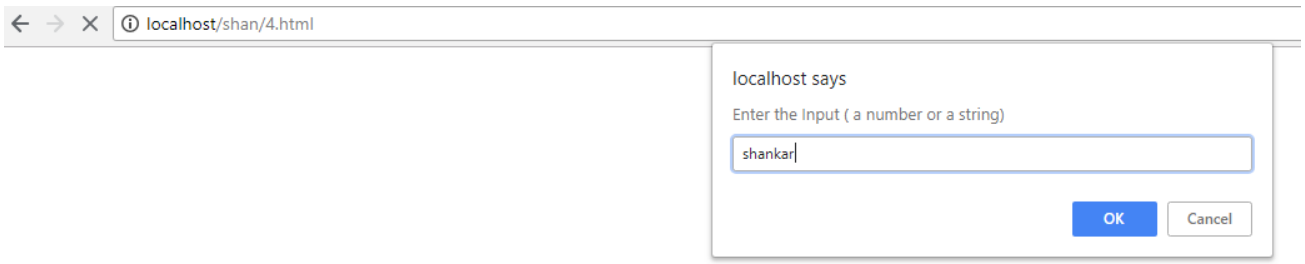
09/03/18 04:11:48 C:\xampp\htdocs\shan\4.html

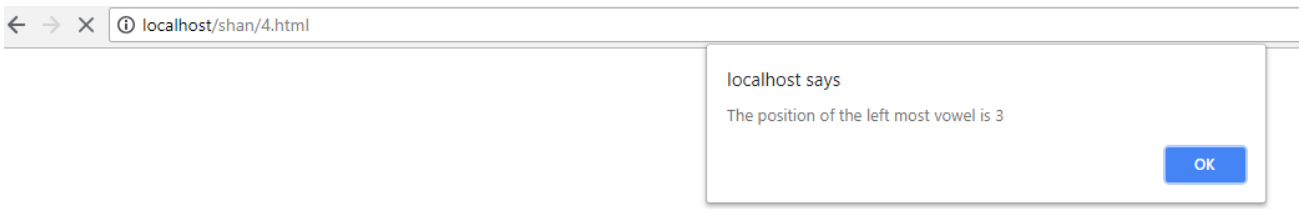
```
<!-- 4. Develop and demonstrate a HTML5 file that includes JavaScript script that uses
functions for the following problems:
a. Parameter: A string
b. Output: The position in the string of the left-most vowel
c. Parameter: A number
d. Output: The number with its digits in the reverse order -->

<!DOCTYPE HTML>
<html>
   <body>
      <script type="text/javascript">

         var str = prompt("Enter the Input ( a number or a string) ","");

         if(!(isNaN(str)))
         {
            var num,rev=0,remainder;
            num = parseInt(str);

            while(num!=0)
            {
               remainder = num%10;
               num = parseInt(num/10);
               rev = rev * 10 + remainder;
            }
            alert("Reverse of "+str+" is "+rev);

         }
         else
         {
            str = str.toUpperCase();
            for(var i = 0; i < str.length; i++)
            {
               var chr = str.charAt(i);
               if(chr == 'A' || chr == 'E' || chr == 'I' || chr == 'O' || chr == 'U')
                  break;
            }
            if( i < str.length )
               alert("The position of the left most vowel is " +(i+1));
```

```
40              else
41                 alert("No vowel found in the entered string");
42           }
43        </script>
44     </body>
45  </html>
46
47
```


localhost/shan/4.html
localhost says
Enter the Input (a number or a string)
23456789
OK
Cancel


localhost/shan/4.html
localhost says
Reverse of 23456789 is 98765432
OK


localhost/shan/4.html
localhost says
Enter the Input (a number or a string)
shankar
OK
Cancel


localhost/shan/4.html
localhost says
The position of the left most vowel is 3
OK

09/03/18 04:15:59 C:\xampp\htdocs\shan\5.xml

```
<!-- 5. Design an XML document to store information about a student in an engineering college affiliated
to VTU. The information must include USN, Name, and Name of the College, Branch, Year of Joining, and
email id. Make up sample data for 3 students. Create a CSS style sheet and use it to display the document.
-->
<?xml-stylesheet type="text/css" href="5.css" ?> <!-- THIS LINE IS VERY IMPORTANT -->
<!DOCTYPE HTML>
<html>
    <head>
        <h1>  STUDENTS DESCRIPTION </h1>
    </head>
    <students>
            <student>
                <USN>USN        :   1BY15CS001</USN>
                <name>NAME      :   AMIT</name>
                <college>COLLEGE:   BMSIT</college>
                <branch>BRANCH  :   Computer Science and Engineering</branch>
                <year>YEAR      :   2019</year>
                <e-mail>E-Mail  :   amit@gmail.com</e-mail>
            </student>

            <student>
                <USN>USN        :   1BY15IS011</USN>
                <name>NAME      :   DANISH</name>
                <college>COLLEGE:   BMSIT</college>
                <branch>BRANCH  :   Information Science and Engineering</branch>
                <year>YEAR      :   2019</year>
                <e-mail>E-Mail  :   danish@gmail.com</e-mail>
            </student>

            <student>
                <USN>USN        :   1BY15EC021</USN>
                <name>NAME      :   PALLAVI</name>
                <college>COLLEGE:   BMSIT</college>
                <branch>BRANCH  :   Electronics and Communication Engineering</branch>
                <year>YEAR      :   2019</year>
                <e-mail>E-Mail  :   pallavi@gmail.com</e-mail>
            </student>
    </students>
</html>
```

09/03/18 04:17:02 C:\xampp\htdocs\shan\5.css

```
1  /*5.css*/
2
3  student
4  {
5      display:block; margin-top:10px; color:Navy;
6  }
7  USN
8  {
9      display:block; margin-left:10px;font-size:14pt; color:Red;
10 }
11 name
12 {
13     display:block; margin-left:20px;font-size:14pt; color:Brown;
14 }
15 college
16 {
17     display:block; margin-left:20px;font-size:12pt; color:Orange;
18 }
19 branch
20 {
21     display:block; margin-left:20px;font-size:12pt; color:Pink;
22 }
23 year
24 {
25     display:block; margin-left:20px;font-size:14pt; color:Green;
26 }
27 e-mail
28 {
29     display:block; margin-left:20px;font-size:12pt; color:Blue;
30 }
31
```

STUDENTS DESCRIPTION

USN : 1BY15CS001
NAME : AMIT
COLLEGE: BMSIT
BRANCH : Computer Science and Engineering
YEAR : 2019
E-Mail : amit@gmail.com

USN : 1BY15IS011
NAME : DANISH
COLLEGE: BMSIT
BRANCH : Information Science and Engineering
YEAR : 2019
E-Mail : danish@gmail.com

USN : 1BY15EC021
NAME : PALLAVI
COLLEGE: BMSIT
BRANCH : Electronics and Communication Engineering
YEAR : 2019
E-Mail : pallavi@gmail.com

09/03/18 04:21:49 C:\xampp\htdocs\shan\6.php

```php
<!-- 6. Write a PHP program to keep track of the number of visitors visiting the web page
and to display this count of visitors, with proper headings. -->

<?php

    print "<h3><center> Welcome to my Blog!</center></h3>";

    $fname="counter.txt";
    $fp = fopen($fname,"r");
    $hits= fscanf($fp,"%d");
    fclose($fp);

    $hits[0]++;
    $fp = fopen($fname,"w");
    fprintf($fp,"%d",$hits[0]);
    fclose($fp);

    print "Total number of views: ".$hits[0];
?>
```

Welcome to my Blog!

Total number of views: 39

09/03/18 04:25:44 C:\xampp\htdocs\shan\7.php

```
1  <!-- 7. Write a PHP program to display a digital clock which displays the current time of the server. -->
2
3  <!DOCTYPE HTML>
4  <html>
5  <head>
6      <meta http-equiv="refresh" content="2"/>
7
8      <style>
9          p
10         {
11             color:white;
12             font-size:40px;
13             text-align: center;
14         }
15         body
16         {
17             background-color:black;
18         }
19     </style>
20
21     <p>
22         <?php
23         date_default_timezone_set('Asia/Kolkata');
24         echo date(" H: i : s  a");
25         ?>
26     </p>
27
28 </head>
29 </html>
```


localhost/shan/7.php
16: 25 : 59 pm

09/04/18 09:09:07 C:\xampp\htdocs\shan\8a.php

```
<!-- 8. Write the PHP programs to do the following:
a. Implement simple calculator operations.
b. Find the transpose of a matrix.
c. Multiplication of two matrices.
d. Addition of two matrices. -->

<!DOCTYPE HTML>
<html>
  <head>
    <style>
      table, td, th
      {
        border: 1px solid black;
        width: 33%;
        text-align: center;
        background-color: DarkGray;
      }
      table { margin: auto; }
      input,p { text-align:right; }
    </style>
  </head>

  <body>

    <form method="post">
      <table>
        <caption><h2> SIMPLE CALCULATOR </h2></caption>
        <tr><td>Value 1:</td><td><input type="text" name="num1"/></td>
          <td rowspan="2"><input type="submit" name="submit" value="calculate">  </td></tr>
          <tr><td>Value 2:</td><td> <input type="text" name="num2"/> </td> </tr>
    </form>

        <?php
              if(isset($_POST['submit'])) // it checks if the input submit is filled
              {
                $num1 = $_POST['num1'];
                $num2 = $_POST['num2'];

                if(is_numeric($num1) and is_numeric($num2))
```

```
40                  {
41                    echo "<tr><td colspan='2'> Addition:</td><td><p>".($num1+$num2)."</p></td></tr>";
42                    echo "<tr><td colspan='2'> Subtraction:</td><td><p>".($num1-$num2)."</p></td></tr>";
43                    echo "<tr><td colspan='2'> Multiplication:</td><td><p>".($num1*$num2)."</p></td></tr>";
44                    echo "<tr><td colspan='2'> Division:</td><td><p>".($num1/$num2)."</p></td></tr>";
45                    echo "</table>";
46                  }
47                  else
48                  {
49                    echo "<script type='text/javascript'>alert(' ENTER VALID NUMBERS');</script>";
50                  }
51                }
52              ?>
53          </table>
54    </body>
55  </html>
56
57
```

SIMPLE CALCULATOR

Value 1:	200	calculate
Value 2:	20	
Addition :		220
Subtraction :		180
Multiplication :		4000
Division :		10

09/03/18 04:35:30 C:\xampp\htdocs\shan\8b.php

```php
<!-- 8. Write the PHP programs to do the following:
b. Find the transpose of a matrix.
c. Multiplication of two matrices.
d. Addition of two matrices. -->
<?php
    $a = array(array(1,2,3),array(4,5,6),array(7,8,9));
    $b = array(array(7,8,9),array(4,5,6),array(1,2,3));

    $m=count($a);
    $n=count($a[2]);
    $p=count($b);
    $q=count($b[2]);

    echo "<b> The  First Matrix: </b> <br/>";
    for ($row = 0; $row < $m; $row++)
    {
        for ($col = 0; $col < $n; $col++)
            echo " ".$a[$row][$col];
        echo "<br/>";
    }

    echo "<b> The  Second Matrix: </b> <br/>";
    for ($row = 0; $row < $p; $row++)
    {
        for ($col = 0; $col < $q; $col++)
            echo "  ".$b[$row][$col];
        echo "<br/>";
    }

    echo "<b> The Transpose of the First  Matrix  is: </b> <br/>";
    for ($row = 0; $row < $m; $row++)
    {
        for ($col = 0; $col < $n; $col++)
            echo " ".$a[$col][$row];
        echo "<br/>";
    }
```

```php
    echo "<b> The Transpose of the Second  Matrix  is: </b> <br/>";
    for ($row = 0; $row < $p; $row++)
    {
        for ($col = 0; $col < $q; $col++)
            echo " ".$b[$col][$row];
        echo "<br/>";
    }

    if(($m===$p) and ($n===$q))
    {
        echo "<b>The Addition of Matrices  is:</b><br/>";
        for ($row = 0; $row < 3; $row++)
        {
            for ($col = 0; $col < 3; $col++)
                echo "  ".$a[$row][$col]+$b[$row][$col]." ";
            echo "<br/>";
        }
    }

    if($n===$p)
    {
        echo "<b>The Multiplication of Matrices: </b><br/>";
        $result=array();
        for ($i=0; $i < $m; $i++)
        {
            for($j=0; $j < $q; $j++)
            {
                $result[$i][$j] = 0;
                for($k=0; $k < $n; $k++)
                    $result[$i][$j] += $a[$i][$k] * $b[$k][$j];
            }
        }
        for ($row = 0; $row < $m; $row++)
        {
            for ($col = 0; $col < $q; $col++)
                echo " ".$result[$row][$col];
            echo "<br/>";
        }
    }
?>
```

The First Matrix:
1 2 3
4 5 6
7 8 9
The Second Matrix:
7 8 9
4 5 6
1 2 3
The Transpose of the First Matrix is:
1 4 7
2 5 8
3 6 9
The Transpose of the Second Matrix is:
7 4 1
8 5 2
9 6 3
The Addition of Matrices is:
8 10 12
8 10 12
8 10 12
The Multiplication of Matrices:
18 24 30
54 69 84
90 114 138

09/04/18 09:11:28 C:\xampp\htdocs\shan\9.php

```
1  <!-- 9. Write a PHP program named states.py that declares a variable states with value "Mississippi
   Alabama Texas Massachusetts Kansas". write a PHP program that does the following:
2  a. Search for a word in variable states that ends in xas. Store this word in element 0 of a list named
   statesList.
3  b. Search for a word in states that begins with k and ends in s. Perform a caseinsensitive comparison.
   [Note: Passing re.Ias a second parameter to method compile performs a case-insensitive comparison.] Store
   this word in element1 of statesList.
4  c. Search for a word in states that begins with M and ends in s. Store this word in element 2 of the list.
5  d. Search for a word in states that ends in a. Store this word in element 3 of the list. -->
6  <?php
7      $states = "Mississippi Alabama Texas Massachusetts Kansas";
8      $statesArray = [];
9      $split = explode(' ',$states);
10
11     echo "<b>Original Array </b>:<br>";
12     foreach ( $split as $i => $value )
13     {
14         print("STATES[$i]=$value<br>");
15     }
16
17     echo "<br><b> Searching for the said patterns..</b><br>";
18
19     foreach($split as $state)
20     {
21         if(preg_match( '/xas$/', ($state))) /*states that ends in xas*/
22             $statesArray[0] = ($state);
23     }
24
25     foreach($split as $state)
26     {
27         if(preg_match('/^k.*s$/i', ($state))) /*states that begins with k and ends in s.*/
28             $statesArray[1] = ($state);
29     }
30
31     foreach($split as $state)
32     {
33         if(preg_match('/^M.*s$/', ($state))) /*states that begins with M and ends in s.*/
34             $statesArray[2] = ($state);
35     }
```

```
36
37      foreach($split as $state)
38      {
39          if(preg_match('/a$/', ($state))) /*states that ends in a*/
40              $statesArray[3] = ($state);
41      }
42
43      echo "<br><b>Resultant Array :</b><br>";
44      foreach ( $statesArray as $array => $value )
45      {
46          print("STATES[$array]=$value<br>");
47      }
48  ?>
49
50
```

Original Array :
STATES[0]=Mississippi
STATES[1]=Alabama
STATES[2]=Texas
STATES[3]=Massachusetts
STATES[4]=Kansas

Searching for the said patterns..

Resultant Array :
STATES[0]=Texas
STATES[1]=Kansas
STATES[2]=Massachusetts
STATES[3]=Alabama

09/04/18 09:13:32 C:\xampp\htdocs\shan\10.php

```
<!-- 10. Write a PHP program to sort the student records which are stored in the database using selection sort. -->

<!DOCTYPE HTML>
<html>
<body>
    <style>
        table, td, th
        {
            border: 1px solid black;
            width: 33%;
            text-align: left;
            border-collapse:collapse;
            background-color:lightblue;
        }
        table { margin: auto; }
    </style>

    <?php
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "test";
        $a=[];

        // Create connection
        $conn = mysqli_connect($servername, $username, $password, $dbname);  //The MySQLi functions allows you to access MySQL database servers.
        // Check connection
        if ($conn->connect_error)
            die("Connection failed: " . $conn->connect_error);

        $sql = "SELECT * FROM student";
        $result = $conn->query($sql); // performs a query against the database

        echo "<br>";
        echo "<center>  BEFORE SORTING </center>";
        echo "<table border='2'>";
        echo "<tr>";
```

```
38          echo "<th>USN</th><th>NAME</th><th>Address</th> </tr>";
39
40          if ($result->num_rows > 0)
41          {
42              // output data of each row
43              while($row = $result->fetch_assoc()) // fetches a result row as an associative array
44              {
45                  echo "<tr>";
46                  echo "<td>". $row["usn"]."</td>";
47                  echo "<td>". $row["name"]."</td>";
48                  echo "<td>". $row["address"]."</td></tr>";
49                  array_push($a,$row["usn"]);
50              }
51          }
52          else
53          {
54              echo "Table is Empty";
55          }
56
57          echo "</table>";
58
59          $n=count($a);
60          $b=$a;
61          for ( $i = 0 ; $i < ($n - 1) ; $i++ )
62          {
63              $pos= $i;
64
65              for ( $j = $i + 1 ; $j < $n ; $j++ )
66              {
67                  if ( $a[$pos] > $a[$j] )
68                      $pos= $j;
69              }
70              if ( $pos!= $i )
71              {
72                  $temp=$a[$i];
73                  $a[$i] = $a[$pos];
74                  $a[$pos] = $temp;
75              }
76          }
77
78          $c=[];
79          $d=[];
```

```
80          $result = $conn->query($sql);
81
82          if ($result->num_rows > 0)
83          {
84              // output data of each row
85              while($row = $result->fetch_assoc())
86              {
87                  for($i=0;$i<$n;$i++)
88                  {
89                      if($row["usn"]== $a[$i])
90                      {
91                          $c[$i]=$row["name"];
92                          $d[$i]=$row["address"];
93                      }
94                  }
95              }
96          }
97
98          echo "<br>";
99          echo "<center>  AFTER SORTING <center>";
100         echo "<table border='2'>";
101         echo "<tr>";
102         echo "<th>USN</th><th>NAME</th><th>Address</th> </tr>";
103         for($i=0;$i<$n;$i++)
104         {
105             echo "<tr>";
106             echo "<td>". $a[$i]."</td>";
107             echo "<td>". $c[$i]."</td>";
108             echo "<td>". $d[$i]."</td></tr>";
109         }
110         echo "</table>";
111
112         $conn->close();
113     ?>
114
115 </body>
116 </html>
117
```

BEFORE SORTING

USN	NAME	Address
11	shankar	Bangalore
12	Abhi	Kolkata
3	Zeeshan	Mysore
44	trath	Tumkur

AFTER SORTING

USN	NAME	Address
3	Zeeshan	Mysore
11	shankar	Bangalore
12	Abhi	Kolkata
44	trath	Tumkur