Workbook

Web Engineering
(SE – 206)

Name _____

Roll No _____

Batch _____

Year _____

Department _____

Workbook

Web Engineering
(SE – 206)

Prepared by

Mrs. Nazish Irfan
I.T Manager, CS&IT

Approved by

Chairman
Department of Computer Science & Information Technology

SE – 206: Web Engineering

Table of Contents

Note: The following software is used: Notepad, Browser IE, Chrome and Opera.

Lab # 1

Object:
Introduction to HTML.

Theory:
Hyper Text Markup Language (HTML) is the main markup language for displaying web pages and other information that can be displayed in a web browser.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>), within the web page content. HTML tags most commonly come in pairs like <h1> and </h1>, although some tags, known as empty elements, are unpaired, for example <img>. The first tag in a pair is the start tag, the second tag is the end tag (they are also called opening tags and closing tags). In between these tags web designers can add text, tags, comments and other types of text-based content.

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts in languages such as JavaScript which affect the behavior of HTML webpages.

HTML Page Structure.

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

HTML Elements.
HTML documents are defined by HTML elements. An HTML element is everything from the start tag to the end tag. The element content is everything between the start and the end tag. Some HTML elements have empty content Empty elements are closed in the start tag. Most HTML elements can have attributes. HTML Document can also contain nested elements.

The **<html> tag** tells the browser that this is an HTML document. The <html> element is also known as the root element. The <html> tag is the container for all other HTML elements

The **<head> element** is a container for all the head elements. The <head> element must include a title for the document, and can include scripts, styles, meta information, and more.

The **<body> tag** defines the document's body. The <body> element contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

HTML Attributes.

Attributes provide additional information about an element Attributes are always specified in the start tag Attributes come in name/value pairs like: name="value". Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed.

Some attributes that are standard for most HTML elements:

Attribute	Value	Description
class	Classname	Specifies a classname for an element.
id	Id	Specifies a unique id for an element.
style	style_definition	Specifies an inline style for an element.
title	tooltip_text	Specifies extra information about an element (displayed as a tool tip).

Exercise:

1. Which of the following is a properly formed HTML document?

A.
```
<html>
<head>
<title>My document</title>
<body>This is my web page</body>
</title>
</html>
```

B.
```
<HTML>
<HEAD>
<TITLE>My document</TITLE>
<BODY>This is my web page
</HTML>
```

C.
```
<html>
<head><title>My document</title></head>
<body>This is my web page</body>
</html>
```

D.
```
<HTML>
<HEAD>
<TITLE>My document</HEAD>
<BODY>This is my web page</BODY>
</HTML>
```

2. What are the advantages and disadvantages of HTML.

Lab # 2

Object:
Exploring Basic HTML Elements.

Theory:
HTML Headings.
Headings are defined with the <h1> to <h6> tags. <h1> defines the most important heading. <h6> defines the least important heading. Browsers automatically add some empty space (a margin) before and after each heading. Use HTML headings for headings only. Don't use headings to make text BIG or bold. Search engines use your headings to index the structure and content of your web pages.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

Attribute	Value
align.	left. right. center. justify.

HTML Lines.
The <hr/> tag creates a horizontal line in an HTML page. The hr element can be used to separate content.

```
<p>This is a paragraph</p>
<hr />
<p>This is a paragraph</p>
<hr />
```

HTML Comments.
Comments can be inserted into the HTML code to make it more readable and understandable. Comments are ignored by the browser and are not displayed.

```
<!-- This is a comment -->
```

HTML Paragraphs.
Paragraphs are defined with the <p> tag.

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Attribute	Value
align.	left. right. center. justify.

HTML Line Breaks.
Use the
 tag if you want a line break (a new line) without starting a new paragraph.

* This is
a para
graph with line breaks *

HTML Formatting Tags.

HTML uses tags like <b> and <i> for formatting output, like bold or italic text.

HTML Text Formatting Tags.

Tag	Description
<b>	Defines bold text.
<big>	Defines big text.
	Defines emphasized text.
<i>	Defines italic text.
<small>	Defines small text.
	Defines strong text.
<sub>	Defines subscripted text.
<sup>	Defines superscripted text.
<ins>	Defines inserted text.
<del>	Defines deleted text.

HTML "Computer Output" Tags.

Tag	Description
<code>	Defines computer code text.
<kbd>	Defines text.
<samp>	Defines sample computer code.
<tt>	Defines teletype text.
<var>	Defines a variable.
<pre>	Defines preformatted text.

HTML Hyperlinks.

A hyperlink (or link) is a word, group of words, or image that you can click on to jump to a new document or a new section within the current document. When we move the cursor over a link in a Web page, the arrow will turn into a little hand.

Links are specified in HTML using the <a> tag.

The <a> tag can be used in two ways:

- To create a link to another document, by using the href attribute.
- To create a bookmark inside a document, by using the name attribute.

The HTML code for a link is simple. It looks like this:

Link text

The href attribute specifies the destination of a link.

Visit NEDUET

HTML Links - The target Attribute.
The target attribute specifies where to open the linked document. The example below will open the linked document in a new browser window or a new tab:

Visit NEDUET!

HTML Links - The name Attribute.
The name attribute specifies the name of an anchor. The name attribute is used to create a bookmark inside an HTML document. Bookmarks are not displayed in any special way. They are invisible to the reader.

A named anchor inside an HTML document:

Useful Tips Section

Create a link to the "Useful Tips Section" inside the same document:

Visit the Useful Tips Section

Or, create a link to the "Useful Tips Section" from another page:

Visit the Useful Tips Section

HTML Links –Email Address.
Anchors can also link to email addresses. When someone clicks on this type of anchored link, their default email program initiates an email message to the linked address.

myemail@neduet.edu.pk

HTML Images - The <img> Tag and the Src Attribute.

In HTML, images are defined with the <img> tag. The <img> tag is empty, which means that it contains attributes only, and has no closing tag. To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display. Syntax for defining an image:

**

Attribute	**Value**	**Description**
alt.	text.	Specifies an alternate text for an image.
src.	url.	Specifies the URL of an image.
align.	top. bottom. middle. left. right.	Specifies the alignment of an image according to surrounding element.
border.	pixels.	Specifies the width of the border around an image.
height, width.	pixels %	Specifies the height and width of an image.
hspace.	pixels.	Specifies the whitespaces on left & right side of an image.
vspace.	pixels.	Specifies the whitespaces on top & bottom of an image.
usemap.	#mapname.	Specifies an image as a client side image map.

HTML Images - The Alt Attribute.
The required alt attribute specifies an alternate text for an image, if the image cannot be displayed. The value of the alt attribute is an author-defined text:

**

The alt attribute provides alternative information for an image if a user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

HTML Images - Set Height and Width of an Image.
The height and width attributes are used to specify the height and width of an image.
The attribute values are specified in pixels by default:

**

HTML Image Map.
The <map> tag is used to define a client-side image-map. An image-map is an image with clickable areas. The name attribute of the <map> element is required and it is associated with the <img>'s usemap attribute and creates a relationship between the image and the map.

The <map> element contains a number of <area> elements that defines the clickable areas in the image map. The <area> element is always nested inside a <map> tag.

```
<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap"
/>
<map name="planetmap">
        <area shape="rect" coords="0,0,82,126" href="sun.html" alt="Sun" />
       <area shape="circle" coords="90,58,3" href="merc.html" alt="Mercury" />
       <area shape="circle" coords="124,58,8" href="venus.html" alt="Venus" />
</map>
```

HTML Tables.
Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc.

```
<table border="1">
<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>
```

HTML Tables and the Border Attribute.

If you do not specify a border attribute, the table will be displayed without borders. Sometimes this can be useful, but most of the time, we want the borders to show. To display a table with borders, specify the border attribute:

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

HTML Table Headers.

Header information in a table are defined with the <th> tag. All major browsers display the text in the <th> element as bold and centered.

```
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<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>
```

HTML Table Tags.

Tag	**Description**
<table>	Defines a table.
<th>	Defines a table header.
<tr>	Defines a table row.
<td>	Defines a table cell.
<caption>	Defines a table caption.
<colgroup>	Defines a group of columns in a table for formatting.
<col/>	Defines attribute values for one or more columns.
<thead>	Groups the header content in a table.
<tbody>	Groups the body content in a table.
<tfoot>	Groups the footer content in a table.

HTML Unordered Lists.

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag. The list items are marked with bullets (typically small black circles).

```
<ul>
<li>Floppies</li>
<li>Harddisks</li>
</ul>
```

Output:

Floppies
Harddisks

HTML Ordered Lists.

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag. The list items are marked with numbers.

```
<ol>
<li>Floppies</li>
<li>Harddisks</li>
</ol>
```

Output:

1. *Floppies.*
2. *Harddisks*

HTML Definition Lists.

A definition list is a list of items, with a description of each item. The <dl> tag defines a definition list. The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list):

```
<dl>
<dt>Keyboard</dt>
<dd>- an input device</dd>
<dt>Printer</dt>
<dd>- an output device</dd>
</dl>
```

Output:

Keyboard
- an input device
Printer
- an output device

HTML and .

HTML elements can be grouped together with and .

HTML Block Elements.

Most HTML elements are defined as block level elements or as inline elements. Block level elements normally start (and end) with a new line when displayed in a browser.
Examples: <h1>, <p>, <ul>, <table>.

HTML Inline Elements.

Inline elements are normally displayed without starting a new line.
Examples: <b>, <td>, <a>, <img>

The HTML Element.

The HTML element is a block level element that can be used as a container for grouping other HTML elements. The element has no special meaning. Except that,

because it is a block level element, the browser will display a line break before and after it. When used together with CSS, the element can be used to set style attributes to large blocks of content. Another common use of the element, is for document layout. It replaces the "old way" of defining layout using tables. Using tables is not the correct use of the <table> element. The purpose of the <table> element is to display tabular data.

The HTML Element.
The HTML element is an inline element that can be used as a container for text. The element has no special meaning. When used together with CSS, the element can be used to set style attributes to parts of the text.

HTML Layouts - Using Elements.
The div element is a block level element used for grouping HTML elements.

The following example uses five div elements to create a multiple column layout:

```
<!DOCTYPE html>
<html>
<body>
<div id="container" style="width:500px">

<div id="header" style="background-color:#FFA500;">
<h1 style="margin-bottom:0;">Main Title of Web Page</h1></div>

<div id="menu"
style="backgroundcolor:#FFD700;height:200px;width:100px;float:left;">
<b>Menu</b><br />
HTML<br />
CSS<br />
JavaScript</div>

<div id="content" style="background-
color:#EEEEEE;height:200px;width:400px;float:left;">
Content goes here</div>

<div id="footer" style="background-color:#FFA500;clear:both;text-align:center;">
</div>

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

HTML Layouts - Using Tables.
A simple way of creating layouts is by using the HTML <table> tag. Multiple columns are created by using or <table> elements. CSS are used to position elements, or to create backgrounds or colorful look for the pages. The following example uses a table with 3 rows and 2 columns - the first and last row spans both columns using the colspan attribute:

```
<!DOCTYPE html>
<html>
<body>
<table width="500" border="0">
<tr>
<td colspan="2" style="background-color:#FFA500;">
<h1>Main Title of Web Page</h1>
</td>
</tr>
<tr valign="top">
<td style="background-color:#FFD700;width:100px;text-align:top;">
<b>Menu</b><br />
HTML<br />
CSS<br />
JavaScript
</td>
<td style="background-color:#EEEEEE;height:200px;width:400px;text-align:top;">
Content goes here</td>
</tr>
<tr>
<td colspan="2" style="background-color:#FFA500;text-align:center;">
</td>
</tr>
</table>

</body>
</html>
```

HTML Forms and Input.

HTML Forms are used to select different kinds of user input. HTML forms are used to pass data to a server. A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements. The <form> tag is used to create an HTML form:

```
<form>
    input elements
</form>
```

Attribute	Value	Description
action.	URL.	Specifies where to send the form-data when a form is submitted.
method.	get/post.	Specifies the HTTP method to use when sending form-data.
name.	name.	Specifies the name of a form.
target.	_blank. _self. _parent. _top.	Specifies where to display the response that is received after submitting the form.

HTML Form - Event Attributes.
HTML Forms tag supports event attributes. The value for all attributes is script. Some of these are given below:

Attribute	**Description**
onclick.	Script to be run on a mouse click.
onmousedown.	Script to be run when mouse button is pressed.
onmousemove.	Script to run when mouse pointer moves.
onmouseover.	Script to run when mouse pointer moves over an element.
onkeypress.	Script to be run when a key is pressed and released.
onsubmit.	Script to be run when a form is submitted.

HTML Forms - The Input Element.
The most important form element is the input element. The input element is used to select user information. An input element can vary in many ways, depending on the type attribute. An input element can be of type text field, checkbox, password, radio button, submit button, and more. The most used input types are described below:

Text Fields.
<input type="text" /> defines a one-line input field that a user can enter text into:

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

Password Field.
<input type="password" /> defines a password field:

```
<form>
Password: <input type="password" name="pwd" />
</form>
```

Radio Buttons.
<input type="radio" /> defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
<input type="radio" name="sex" value="male" /> Male<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

Checkboxes.
<input type="checkbox" /> defines a checkbox. Checkboxes let a user select ONE or MORE options of a limited number of choices.

```
<form>
<input type="checkbox" name="vehicle" value="Bike" /> I have a bike<br />
<input type="checkbox" name="vehicle" value="Car" /> I have a car
</form>
```

Submit Button.

<input type="submit" /> defines a submit button. A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="html_form_action.asp" method="get">
Username: <input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

Exercise:

1. Write the HTML tags for :
   a. A paragraph that is a description of a book, include the title of the book as well as its author. Names and titles should be underlined, italicized and bolded.

   b. Print two lists with any information you want. One list should be an ordered list and the other should be unordered list.

   c. Create a page with a link at the top of it when clicked will jump all the way to the bottom of the page and at the bottom of the page there should be a link to jump back to the top of the page.

   d. Create some links to various search engines.

   e. Display an image that when clicked will link to a search engine of your choice and the page should open in a new window.

2. Design a form to take input from the user to generate his / her CV.

Lab # 3

Object:
Introduction to HTML5

Theory:
HTML5 is the new standard for HTML. The previous version of HTML, HTML 4.01, came in 1999. The web has changed a lot since then. HTML5 is still a work in progress. However, the major browsers support many of the new HTML5 elements and APIs.

HTML5 is cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG). WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.

Some rules for HTML5 were established:
- New features should be based on HTML, CSS, DOM, and JavaScript
- Reduce the need for external plugins (like Flash)
- Better error handling
- More markup to replace scripting
- HTML5 should be device independent
- The development process should be visible to the public

The HTML5 <!DOCTYPE>.
In HTML5 there is only one <!doctype> declaration, and it is very simple:

```
<!DOCTYPE html>
```

Minimum HTML5 Document.
Below is a simple HTML5 document, with the minimum of required tags:

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>
<body>
The content of the document.....
</body>
</html>
```

HTML5 - New Features.
Some of the most interesting new features in HTML5:

- The <canvas> element for 2D drawing
- The <video> and <audio> elements for media playback
- Support for local storage
- New content-specific elements, like <article>, <footer>, <header>, <nav>, <section>
- New form controls, like calendar, date, time, email, url, search

Browser Support for HTML5.

HTML5 is not yet an official standard, and no browsers have full HTML5 support. But all major browsers (Safari, Chrome, Firefox, Opera, Internet Explorer) continue to add new HTML5 features to their latest versions.

New Elements in HTML5.

The internet has changed a lot since HTML 4.01 became a standard in 1999. Today, some elements in HTML 4.01 are obsolete, never used, or not used the way they were intended to. These elements are removed or re-written in HTML5. To better handle today's internet use, HTML5 includes new elements for better structure, better form handling, drawing, and for media content.

New Semantic/Structural Elements.

Tag	Description
<article>	Defines an article.
<aside>	Defines content aside from the page content
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it.
<command>	Defines a command button that a user can invoke
<details>	Defines additional details that the user can view or hide
<summary>	Defines a visible heading for a <details> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc
<figcaption>	Defines a caption for a <figure> element
<footer>	Defines a footer for a document or section
<header>	Defines a header for a document or section
<hgroup>	Groups a set of <h1> to <h6> elements when a heading has multiple levels.
<mark>	Defines marked/highlighted text
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links
<progress>	Represents the progress of a task
<ruby>	Defines a ruby annotation (for East Asian typography)
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<rp>	Defines what to show in browsers that do not support ruby annotations
<section>	Defines a section in a document
<time>	Defines a date/time
<wbr>	Defines a possible line-break

Removed Elements.

The following HTML 4.01 elements are removed from HTML5:
<acronym>, <applet>, <basefont>, <big>, , <dir>, <font>, <frame>, <frameset>, <noframes>, <strike>, <tt>

Exercise:

1. Write down the advantages and disadvantages of HTML5.

Lab # 4

Object:
Exploring HTML5 Elements.

Theory:
HTML5 Video.
There has not been a standard for showing a video/movie on a web page until now. Today, most videos are shown through a plug-in (like flash). However, different browsers may have different plug-ins.

HTML5 defines a new element which specifies a standard way to embed a video/movie on a web page: the <video> element. The control attribute adds video controls, like play, pause, and volume. It is also a good idea to always include width and height attributes. If height and width are set, the space required for the video is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the video, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the video loads).

Text should be inserted between the <video> and </video> tags for browsers that do not support the <video> element.

The <video> element allows multiple <source> elements. <source> elements can link to different video files. The browser will use the first recognized format.

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  Your browser does not support the video tag.
</video>
```

Attribute	Value	Description
autoplay.	autoplay.	Specifies that the video will start playing as soon as it is ready.
controls.	controls.	Specifies that video controls should be displayed (such as a play/pause button etc).
height.	pixels.	Sets the height of the video player.
loop.	loop.	Specifies that the video will start over again, every time it is finished
muted.	muted.	Specifies that the audio output of the video should be muted.
poster.	URL.	Specifies an image to be shown while the video is downloading, or until the user hits the play button.
preload.	auto, metadata, none.	Specifies if and how the author thinks the video should be loaded when the page loads.
src.	URL.	Specifies the URL of the video file.
width.	pixels.	Specifies the width of the video player.

HTML5 Audio.

There has not been a standard for playing audio files on a web page until now. Today, most audio files are played through a plug-in (like flash). However, different browsers may have different plug-ins.

HTML5 defines a new element which specifies a standard way to embed an audio file on a web page: the <audio> element. Text should be inserted between the <audio> and </audio> tags for browsers that do not support the <audio> element.

The <audio> element allows multiple <source> elements. <source> elements can link to different audio files. The browser will use the first recognized format.

```
<audio controls="controls">
  <source src="song.ogg" type="audio/ogg" />
  <source src="song.mp3" type="audio/mpeg" />
  Your browser does not support the audio element.
</audio>
```

Attribute	Value	Description
autoplay.	autoplay.	Specifies that the audio will start playing as soon as it is ready.
controls.	controls.	Specifies that audio controls should be displayed (such as a play/pause button etc).
loop.	loop.	Specifies that the audio will start over again, every time it is finished
preload.	auto, metadata, none.	Specifies if and how the author thinks the audio should be loaded when the page loads.
src.	URL.	Specifies the URL of the audio file.

HTML5 Drag and Drop.

Drag and drop is a part of the HTML5 standard. Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location. In HTML5, drag and drop is part of the standard, and any element can be draggable.

```
 <!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function allowDrop(ev)
{
ev.preventDefault();
}

function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
```

```
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69" />

</body>
</html>
```

Make an Element Draggable.

First of all: To make an element draggable, set the draggable attribute to true:

```
<img draggable="true" />
```

What to Drag - ondragstart and setData().

Then, specify what should happen when the element is dragged. In the example above, the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged. The dataTransfer.setData() method sets the data type and the value of the dragged data:

```
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
```

In this case, the data type is "Text" and the value is the id of the draggable element ("drag1").

Where to Drop – ondragover.

The ondragover event specifies where the dragged data can be dropped. By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element. This is done by calling the event.preventDefault() method for the ondragover event:

```
event.preventDefault()
```

Do the Drop – ondrop.

When the dragged data is dropped, a drop event occurs. In the example above, the ondrop attribute calls a function, drop(event):

```
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
```

Code explained:

- Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop).
- Get the dragged data with the dataTransfer.getData("Text") method. This method will return any data that was set to the same type in the setData() method.
- The dragged data is the id of the dragged element ("drag1").
- Append the dragged element into the drop element.

HTML5 Canvas.

The HTML5 <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript). The <canvas> element is only a container for graphics; you must use a script to actually draw the graphics.

A canvas is a draw able region defined in HTML code with height and width attributes. Canvas has several methods for drawing paths, boxes, circles, characters, and adding images.

Create a Canvas.

A canvas is specified with the <canvas> element. Specify the id, width, and height of the <canvas> element:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Draw With JavaScript.

The <canvas> element has no drawing abilities of its own. All drawing must be done inside a JavaScript:

```
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
</script>
```

JavaScript uses the id to find the <canvas> element:

```
var c=document.getElementById("myCanvas");
```

Then, create a context object:

```
var ctx=c.getContext("2d");
```

The getContext("2d") object is a built-in HTML5 object, with many methods to draw paths, boxes, circles, characters, images and more. The next two lines draws a red rectangle:

```
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
```

The fillStyle attribute makes it red, and the fillRect attribute specifies the shape, position, and size.

Attribute	Value	Description
height.	pixels.	Specifies the height of the canvas.
width.	pixels.	Specifies the width of the canvas.

HTML5 New Input Type.

HTML5 has several new input types for forms. These new features allow better input control and validation.

- Color.
- Date.
- Datetime.
- datetime-local.
- Email.
- Month.
- Number.
- range.
- search.
- tel.
- time.
- url.
- week.

Input Type: color.

The color type is used for input fields that should contain a color.

Select your favorite color: <input type="color" name="favcolor" />

Input Type: date.

The date type allows the user to select a date.

Birthday: <input type="date" name="bday" />

Input Type: datetime.

The datetime type allows the user to select a date and time (with time zone).

Birthday (date and time): <input type="datetime" name="bdaytime" />

Input Type: datetime-local.

The datetime-local type allows the user to select a date and time (no time zone).

Birthday (date and time): <input type="datetime-local" name="dtime" />

Input Type: month.

The month type allows the user to select a month and year.

Birthday (month and year): <input type="month" name="bdaymonth" />

Input Type: number.

The number type is used for input fields that should contain a numeric value. You can also set restrictions on what numbers are accepted.

Quantity (between 1 & 5):<input type="number" name="quantity" min="1" max="5" />

Input Type: range.

The range type is used for input fields that should contain a value from a range of numbers. You can also set restrictions on what numbers are accepted.

<input type="range" name="points" min="1" max="10" />

Use the following attributes to specify restrictions on number & range:
max - specifies the maximum value allowed.
min - specifies the minimum value allowed.
step - specifies the legal number intervals.
value - Specifies the default value.

Input Type: search.
The search type is used for search fields (a search field behaves like a regular text field).

Search Google: <input type="search" name="googlesearch" />

Input Type: tel.

Telephone: <input type="tel" name="usrtel" />

Input Type: time.
The time type allows the user to select a time.

Select a time: <input type="time" name="usr_time" />

Input Type: url.
The url type is used for input fields that should contain a URL address. The value of the url field is automatically validated when the form is submitted.

Add your homepage: <input type="url" name="homepage" />

Input Type: week.
The week type allows the user to select a week and year.

Select a week: <input type="week" name="week_year" />

HTML5 Form Element.
HTML5 has the following new form elements:
<datalist>.
<keygen>.
<output>.

<datalist> Element.
The <datalist> element specifies a list of pre-defined options for an <input> element. The <datalist> element is used to provide an "autocomplete" feature on <input> elements. Users will see a drop-down list of pre-defined options as they input data. Use the <input> element's list attribute to bind it together with a <datalist> element.

```
<input list="browsers" />
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

<keygen> Element.
The purpose of the <keygen> element is to provide a secure way to authenticate users. The <keygen> tag specifies a key-pair generator field in a form. When the form is

submitted, two keys are generated, one private and one public. The private key is stored locally, and the public key is sent to the server. The public key could be used to generate a client certificate to authenticate the user in the future.

```
<form action="demo_keygen.asp" method="get">
 Username: <input type="text" name="usr_name" />
 Encryption: <keygen name="security" />
 <input type="submit" />
</form>
```

<output> Element

The <output> element represents the result of a calculation (like one performed by a script).

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
 <input type="range" name="a" value="50" />100
 +<input type="number" name="b" value="50" />
 =<output name="x" for="a b"></output>
</form>
```

HTML5 Form Attribute.

HTML5 has several new attributes for <form> and <input>.

New attributes for <form>:

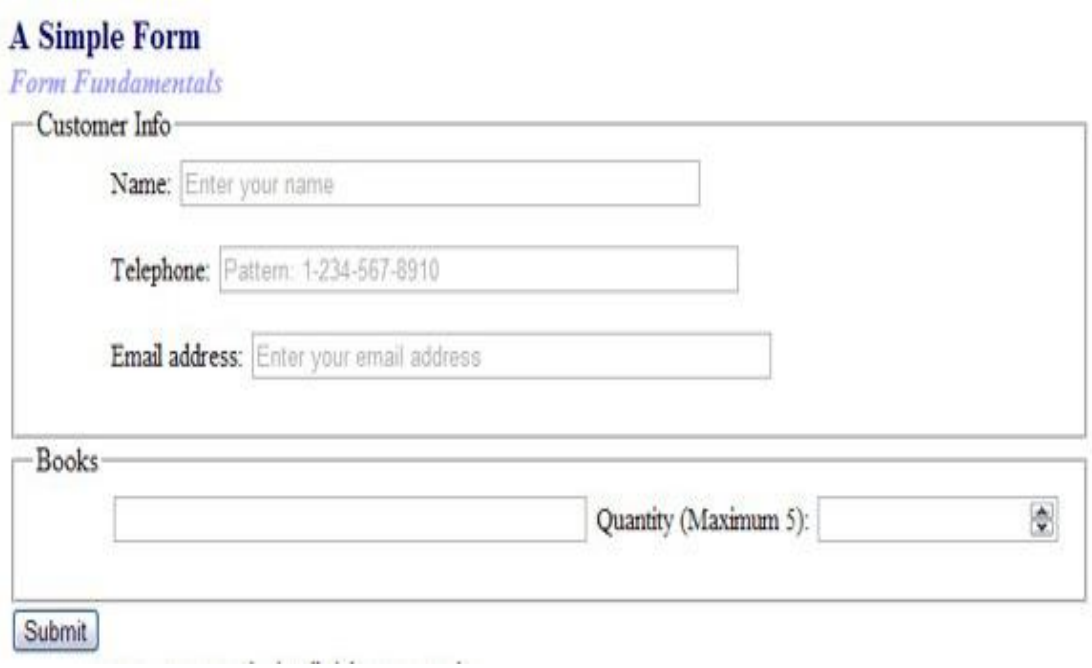
- autocomplete.
- novalidate.

New attributes for <input>:

- autocomplete.
- Autofocus.
- Form.
- Formaction.
- Formenctype.
- Formmethod.
- Formnovalidate.
- Formtarget.
- height and width.
- List.
- min and max.
- multiple.
- pattern (regexp).
- Placeholder.
- Required.
- Step.

Exercise:

1. Create the form shown in the figure below.



When creating the code for your form, you must use the HTML5 tags that are appropriate to replicate the form and fulfill all the specifications listed.

a) Code the form with autocomplete active.
b) The Name field you create should have autofocus, placeholder text, and be required. Don't forget to select the appropriate type for this field as well as all the fields that follow.
c) The Telephone field should have placeholder text, a pattern to restrict entry, and be required.
d) The Email address field should have placeholder text and allow multiple entries. This field should also be required.
e) The Books field should have a data list. You can select the content you would like to list.
f) The Quantity (Maximum 5) field should have a minimum value of 1 and a maximum value of 5.

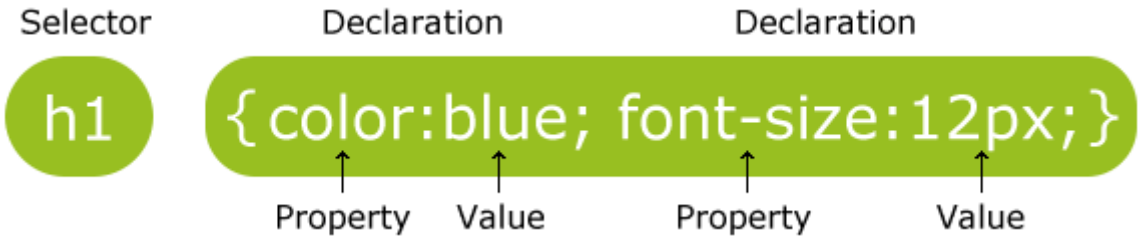
Lab # 5

Object:
Introduction to Cascading Style Sheets.

Theory:
CSS stands for Cascading Style Sheets. Styles define how to display HTML elements. HTML was never intended to contain tags for formatting a document. HTML was intended to define the content of a document.

Development of large web sites, where fonts and color information were added to every single page, became a long and expensive process. To solve this problem, the World Wide Web Consortium (W3C) created CSS. In HTML 4.0, all formatting could be removed from the HTML document, and stored in a separate CSS file. All browsers today support CSS.

CSS defines HOW HTML elements are to be displayed. Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single file! A CSS rule has two main parts: a selector, and one or more declarations:



The selector is normally the HTML element you want to style. Each declaration consists of a property and a value. The property is the style attribute you want to change. Each property has a value.

A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly brackets:

p {color:red;text-align:center;}

To make the CSS more readable, you can put one declaration on each line, like this:

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

CSS Comments.
A CSS comment begins with "/*", and ends with "*/".

The id and class Selectors.

In addition to setting a style for a HTML element, CSS allows us to specify our own selectors called "id" and "class".

The id Selector.

The id selector is used to specify a style for a single, unique element. The id selector uses the id attribute of the HTML element, and is defined with a "#". The style rule below will be applied to the element with id="para1":

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

The class Selector.

The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements. This allows you to set a particular style for many HTML elements with the same class. The class selector uses the HTML class attribute, and is defined with a ".".

```
.center {text-align:center;}
```

We can also specify that only specific HTML elements should be affected by a class.

```
p.center {text-align:center;}
```

Ways to Insert CSS.

When a browser reads a style sheet, it will format the document according to it. There are three ways of inserting a style sheet:

- External style sheet.
- Internal style sheet.
- Inline style.

External Style Sheet.

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, we can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet should be saved with a .css extension.

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
```

Internal Style Sheet.

An internal style sheet should be used when a single document has a unique style. The internal styles are defined in the head section of an HTML page, by using the <style> tag.

```
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

Inline Styles.

An inline style loses many of the advantages of style sheets by mixing content with presentation. To use inline styles we use the style attribute in the relevant tag. The style attribute can contain any CSS property.

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

Multiple Style Sheets.

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

```
h3
{
color:red;
text-align:left;
font-size:8pt;
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
text-align:right;
font-size:20pt;
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color:red;
text-align:right;
font-size:20pt;
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

Multiple Styles Will Cascade into One

Styles can be specified inside an HTML element, inside the head section of an HTML page in an external CSS file

Cascading order.

When there is more than one style specified for an HTML element, the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

- Browser default.
- External style sheet.
- Internal style sheet (in the head section).
- Inline style (inside an HTML element).

Therefore, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).

CSS Background.

CSS background properties are used to define the background effects of an element. CSS properties used for background effects:

- background-color.
- background-image.
- background-repeat.
- background-attachment.
- background-position.

Background Color.

The background-color property specifies the background color of an element. The background color of a page is defined in the body selector:

```
body {background-color:#b0c4de;}
h1 {background-color:#6495ed;}
p {background-color:#e0ffff;}
div {background-color:#b0c4de;}
```

With CSS, a color is most often specified by:
a HEX value - like "#ff0000" ***OR*** an RGB value - like "rgb(255,0,0)" ***OR*** a color name - like "red"

Background Image.

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element. The background image for a page can be set like this:

```
body {background-image:url('paper.gif');}
body {background-image:url('bgdesert.jpg');}
```

Repeat Horizontally or Vertically.

By default, the background-image property repeats an image both horizontally and vertically. Some images should be repeated only horizontally or vertically, like this:

```
body
{
background-image:url('gradient2.png');
}
```

If the image is repeated only horizontally (repeat-x), the background will look better:

```
body
{
background-image:url('gradient2.png');
background-repeat:repeat-x;
```

```
}
```

Set position and no-repeat.

When using a background image, use an image that does not disturb the text. Showing the image only once is specified by the background-repeat property:

```
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
}
```

The background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much. The position of the image is specified by the background-position property:

```
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
background-position:right top;
}
```

CSS Text.

Text Color.

The color property is used to set the color of the text. The default color for a page is defined in the body selector.

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

Text Alignment.

The text-align property is used to set the horizontal alignment of a text. Text can be centered, or aligned to the left or right, or justified. When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

```
h1 {text-align:center;}
p.date {text-align:right;}
p.main {text-align:justify;}
```

Text Decoration.

The text-decoration property is used to set or remove decorations from text. The text-decoration property is mostly used to remove underlines from links for design purposes:

```
a {text-decoration:none;}
```

It can also be used to decorate text:

```
h1 {text-decoration:overline;}
h2 {text-decoration:line-through;}
h3 {text-decoration:underline;}
h4 {text-decoration:blink;}
```

Text Transformation.

The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

p.uppercase {text-transform:uppercase;}
p.lowercase {text-transform:lowercase;}
p.capitalize {text-transform:capitalize;}

Text Indentation.
The text-indentation property is used to specify the indentation of the first line of a text.

p {text-indent:50px;}

CSS Font.

CSS font properties define the font family, boldness, size, and the style of a text.

CSS Font Families.
In CSS, there are two types of font family names:

- generic family - a group of font families with a similar look (like "Serif" or "Monospace")
- font family - a specific font family (like "Times New Roman" or "Arial")

Font Family.
The font family of a text is set with the font-family property. The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font. Start with the font we want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available. More than one font family is specified in a comma-separated list:

p{font-family:"Times New Roman", Times, serif;}

Font Style.
The font-style property is mostly used to specify italic text. This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

p.normal {font-style:normal;}
p.italic {font-style:italic;}
p.oblique {font-style:oblique;}

Font Size.
The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, font size adjustments should not be made so as to make paragraphs look like headings, or headings look like paragraphs. Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs. The font-size value can be an absolute or relative size.

Absolute size: Sets the text to a specified size. Does not allow a user to change the text size in all browsers (bad for accessibility reasons) Absolute size is useful when the physical size of the output is known.

Relative size: Sets the size relative to surrounding elements. Allows a user to change the text size in browsers

Set Font Size With Pixels.

Setting the text size with pixels gives you full control over the text size:

```
h1 {font-size:40px;}
h2 {font-size:30px;}
p {font-size:14px;}
```

Set Font Size With Em.

To avoid the resizing problem with older versions of Internet Explorer, many developers use em instead of pixels. The em size unit is recommended by the W3C. 1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px. The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */
h2 {font-size:1.875em;} /* 30px/16=1.875em */
p {font-size:0.875em;} /* 14px/16=0.875em */
```

The text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers. Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em.

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

```
body {font-size:100%;}
h1 {font-size:2.5em;}
h2 {font-size:1.875em;}
p {font-size:0.875em;}
```

CSS Links.

Styling Links.

Links can be styled with any CSS property (e.g. color, font-family, background, etc.). Special for links are that they can be styled differently depending on what state they are in. The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

```
a:link {color:#FF0000;}      /* unvisited link */
a:visited {color:#00FF00;}  /* visited link */
a:hover {color:#FF00FF;}  /* mouse over link */
a:active {color:#0000FF;}  /* selected link */
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Common Link Styles.

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

Background Color

The background-color property specifies the background color for links:

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

CSS Lists.

The CSS list properties allows to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

List.

In HTML, there are two types of lists:

- unordered lists - the list items are marked with bullets
- ordered lists - the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker.

Different List Item Markers.

The type of list item marker is specified with the list-style-type property:

```
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}
ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

Some of the values are for unordered lists, and some for ordered lists.

An Image as The List Item Marker.

To specify an image as the list item marker, use the list-style-image property:

```
ul
{
list-style-image: url('sqpurple.gif');
}
```

The code above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari. If the image-marker is to be placed equally in all browsers, a crossbrowser solution is required

Crossbrowser Solution.

```
ul
{
list-style-type: none;
padding: 0px;
```

```
margin: 0px;
}
ul li
{
background-image: url(sqpurple.gif);
background-repeat: no-repeat;
background-position: 0px 5px;
padding-left: 14px;
}
```

Explained:
For ul:

- Set the list-style-type to none to remove the list item marker
- Set both padding and margin to 0px (for cross-browser compatibility)

For all li in ul:

- Set the URL of the image, and show it only once (no-repeat)
- Position the image where you want it (left 0px and down 5px)
- Position the text in the list with padding-left

CSS Tables.

The look of an HTML table can be greatly improved with CSS.

Table Borders.

To specify table borders in CSS, use the border property.

```
table, th, td
{
border: 1px solid black;
}
```

The table in the example above has double borders. This is because both the table and the th/td elements have separate borders. To display a single border for the table, use the border-collapse property.

Collapse Borders.

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

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

Table Width and Height.

Width and height of a table is defined by the width and height properties. The example below sets the width of the table to 100%, and the height of the th elements to 50px:

```
table
{
width:100%;
}
th
{
height:50px;
}
```

Table Text Alignment.

The text in a table is aligned with the text-align and vertical-align properties. The text-align property sets the horizontal alignment, like left, right, or center:

```
td
{
text-align:right;
}
```

The vertical-align property sets the vertical alignment, like top, bottom, or middle:

```
td
{
height:50px;
vertical-align:bottom;
}
```

Table Padding

To control the space between the border and content in a table, use the padding property on td and th elements:

```
td
{
padding:15px;
}
```

Table Color

The example below specifies the color of the borders, and the text and background color of th elements:

```
table, td, th
{
border:1px solid green;
}
th
{
background-color:green;
color:white;
}
```

CSS Box Model.

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content. The box model allows us to place a border around elements and space elements in relation to other elements. The box model allows us to place a border around elements

and space elements in relation to other elements. The image below illustrates the box model:

Margin - Clears an area around the border. The margin does not have a background color, it is completely transparent
Border - A border that goes around the padding and content. The border is affected by the background color of the box
Padding - Clears an area around the content. The padding is affected by the background color of the box
Content - The content of the box, where text and images appear

Width and Height of an Element
When setting the width and height properties of an element with CSS, just set the width and height of the content area. To calculate the full size of an element, also add the padding, borders and margins. The total width of the element in the example below is 300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

Let's do the math:
250px (width) + 20px (left and right padding) + 10px (left and right border) + 20px (left and right margin) = 300px

The total width of an element should be calculated like this:
Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should be calculated like this:
Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Exercise:

1. What are the advantages and disadvantages of CSS.

2. What is the advantage of CSS web design over tables.

Lab # 6

Object:
Introduction to Javascript.

Theory:
A scripting language is a lightweight programming language that supports the writing of scripts. Scripts are code lines that can be interpreted and executed "on-the-fly", without explicit compile and link steps.

JavaScript gives HTML designers a programming tool. It is the most popular scripting language in the world. It is the standard language used in web pages, but it is also widely used by desktop apps, mobile phone apps, and internet servers. JavaScript was designed to add interactivity to HTML pages. JavaScript is Case Sensitive.

JavaScript can:
- manipulate HTML and CSS.
- both read and change the content and style of HTML elements.
- be used to validate data, like validating forms input.
- be used to store and retrieve information on the visitor's computer.
- be set to execute when something happens, like when a user clicks on an HTML element.

JavaScript was invented by Brendan Eich at Netscape, and first appeared in Netscape Navigator (a no longer existing web browser) in1995. First it was called Mocha, then LiveScript, and finally JavaScript.

The JavaScript standard was adopted by the industry standard association ECMA in 1997. The standard (called ECMAScript-262) was approved as an international ISO standard in 1998. The development of ECMAScript is still in process.

The <script> Tag.
To insert a JavaScript into an HTML page, use the <script> tag. The <script> and </script> tells where the JavaScript starts and ends. The lines between the <script> and </script> contain the JavaScript:

```
<script>
alert("My First JavaScript");
</script>
```

Manipulating HTML Elements.
To access an HTML element from a JavaScript, use the document.getElementById(id) method, and an "id" attribute to identify the HTML element:

Access the HTML element with the specified id, and change its content:

```
<!DOCTYPE html>
```

```
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo">My First Paragraph</p>
<script>
document.getElementById("demo").innerHTML="My First
JavaScript";
</script>
</body>
</html>
```

The example below writes a <p> element directly into the HTML document output:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<script>
document.write("<p>My First JavaScript</p>");
</script>
</body>
</html>
```

JavaScript Functions and Events.

The JavaScript statement in the example above, is executed when the page loads, but that is not always what we want. Sometimes we want to execute a JavaScript when an event occurs, such as when a user clicks a button. Then we can write the script inside a function, and call the function when the event occurs.

Scripts in <head> and <body>.

We can place an unlimited number of scripts in our document, and we can have scripts in both the <body> and the <head> section at the same time. It is a common practice to put functions in the <head> section, or at the bottom of the page. This way they are all in one place and do not interfere with page content.

Using an External JavaScript.

Scripts can also be placed in external files. External files often contain code to be used by several different web pages. External JavaScript files have the file extension .js. To use an external script, point to the .js file in the "src" attribute of the <script> tag:

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

We can place the script in the head or body. The script will behave as it was located exactly where you put the <script> tag in the document. External scripts cannot contain <script> tags.

JavaScript Statements.

JavaScript is a sequence of statements to be executed by the browser. JavaScript statements are "commands" to the browser. The purpose of the statements is to tell the browser what to do. This JavaScript statement tells the browser to write "Hello Dolly" inside an HTML element with id="demo":

```
document.getElementById("demo").innerHTML="Hello Dolly";
```

Semicolon.

Semicolon separates JavaScript statements. Normally a semicolon is added at the end of each executable statement. Using semicolons also makes it possible to write many statements on one line. Ending statements with semicolon is optional in JavaScript.

JavaScript Code.

JavaScript code (or just JavaScript) is a sequence of JavaScript statements. Each statement is executed by the browser in the sequence they are written.

```
document.getElementById("demo").innerHTML="Hello Dolly";
document.getElementById("myDIV").innerHTML="How are you?";
```

JavaScript Code Blocks.

JavaScript statements can be grouped together in blocks. Blocks start with a left curly bracket, and end with a right curly bracket. The purpose of a block is to make the sequence of statements execute together. (also called JavaScript Functions).

```
function myFunction()
{
document.getElementById("demo").innerHTML="Hello Dolly";
document.getElementById("myDIV").innerHTML="How are you?";
}
```

JavaScript Comments.

Comments will not be executed by JavaScript. Comments can be added to explain the JavaScript, or to make the code more readable. Single line comments start with //. Multi line comments start with /* and end with */.

```
// Write to a heading:
    document.getElementById("myH1").innerHTML="Welcome to my Homepage";
// Write to a paragraph:
    document.getElementById("myP").innerHTML="This is my first paragraph.";

/*
The code below will write to a heading and to a paragraph, and will represent the start of my homepage:*/
```

JavaScript Variables.

Variable can have a short names, like x and y, or more descriptive names, like age, sum, or, totalvolume. JavaScript variables can also be used to hold text values, like: name="John Doe".

Declaring (Creating) JavaScript Variables.

Creating a variable in JavaScript is most often referred to as "declaring" a variable. JavaScript variables are declared using the ***var*** keyword:

```
var carname;
carname="Volvo";
OR
var carname="Volvo";
```

JavaScript Data Types.

There are many types of JavaScript variables which include String, Number, Boolean, Array, Object, Null, Undefined.

JavaScript Strings.

A string is a variable which stores a series of characters.

```
var carname="Volvo XC60";
var carname='Volvo XC60';

var answer="It's alright";
var answer="He is called 'Johnny'";
var answer='He is called "Johnny"';
var answer='It\'s alright';
var answer="He is called \"Johnny\"";
```

Each characters in a string can be accessed with [position]:

```
var character=carname[7];
```

String indexes are zero-based, which means the first character is [0], the second is [1], and so on.

JavaScript Numbers.

JavaScript has only one type of number. Numbers can be with, or without decimals:

```
var pi=3.14;
var x=34;
```

The maximum number of decimals is 17. Extra-large or extra small numbers can be written with scientific notation:

```
var y=123e5;    // 12300000
var z=123e-5;   // 0.00123
```

JavaScript Booleans.

Booleans can have only two values: true or false.

```
var x=true
var y=false
```

Booleans are often used in conditional testing.

JavaScript Arrays.

The following code creates an Array called myCars:

```
var cars=new Array();
cars[0]="Saab";
cars[1]="Volvo";
cars[2]="BMW";
```

or (condensed array):

```
var cars=new Array("Saab","Volvo","BMW");
```

or (literal array):

```
var cars=["Saab","Volvo","BMW"];
```

Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

JavaScript Objects.

An object is delimited by curly braces. Inside the braces the object's properties are defined as name and value pairs (name : value). The properties are separated by commas:

```
var person={firstname:"John", lastname:"Doe", id:5566};
```

The object (person) in the example above has 3 properties: fistname, lastname, and id.

```
var person={
firstname : "John",
lastname  : "Doe",
id        :  5566
};
```

The object properties can be addressed in two ways:

```
name=person.lastname;
name=person["lastname"];
```

Null or Undefined.

Non-existing is the value of a variable with no value. Variables can be emptied by setting the value to null;

```
cars=null;
person=null;
```

Declaring Variable Types.

When declaring a new variable, declare its type using the "new" keyword:

```
var carname=new String;
var x=      new Number;
var y=      new Boolean;
var cars=   new Array;
var person= new Object;
```

All variables in JavaScript are objects. When we declare a variable we create a new object.

JavaScript Functions.

A function is a block of code that executes only when you tell it to execute. It can be when an event occurs, like when a user clicks a button, or from a call within your script, or from a call within another function. Functions can be placed both in the <head> and in the <body> section of a document, just make sure that the function exists, when the call is made.

How to Define a Function.

```
function functionname()
{
some code
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
alert("Hello World!");
}
</script>
</head>

<body>
<button onclick="myFunction()">Try it</button>
</body>
</html>
```

Calling a Function with Arguments.

When we call a function, we can pass along some values to it, these values are called arguments or parameters. These arguments can be used inside the function. We can send as many arguments as we like, separated by commas (,)

```
myFunction(argument1,argument2)
```

Declare the argument, as variables, when you declare the function:

```
function myFunction(var1,var2)
{
some code
}
```

The variables and the arguments must be in the expected order. The first variable is given the value of the first passed argument etc.

```
<button onclick="myFunction('Harry Potter','Wizard')">Try it</button>
```

```
<script>
function myFunction(name,job)
{
alert("Welcome " + name + ", the " + job);
}
</script>
```

The function is flexible, we can call the function using different arguments, and different welcome messages will be given:

```
<button onclick="myFunction('Harry Potter','Wizard')">Try it</button>
<button onclick="myFunction('Bob','Builder')">Try it</button>
```

Functions With a Return Value.

Sometimes you want your function to return a value back to where the call was made. This is possible by using the *return* statement. When using the *return* statement, the function will stop executing, and return the specified value.

```
function myFunction()
{
var x=5;
return x;
}
```

The function above will return the value 5.

The return statement is also used to simply exit a function. The return value is optional:

```
function myFunction(a,b)
{
if (a>b)
{
  return;
}
x=a+b
}
```

Local & Global JavaScript Variables.

A variable declared (using var) within a JavaScript function becomes LOCAL and can only be accessed from within that function. (the variable has local scope). We can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared. Local variables are deleted as soon as the function is completed.

Variables declared outside a function, become GLOBAL, and all scripts and functions on the web page can access it.

If we assign a value to variable that has not yet been declared, the variable will automatically be declared as a GLOBAL variable.

This statement:

```
carname="Volvo";
```

will declare the variable carname as a global variable , even if it is executed inside a function.

JavaScript Operators.

JavaScript Arithmetic Operators.

Arithmetic operators are used to perform arithmetic between variables and/or values. The arithmetic operators include + , - , * , /, %, ++ and --.

JavaScript Assignment Operators.

Assignment operators are used to assign values to JavaScript variables. They include = , += , -+ , *= , /= , %=

The + Operator Used on Strings.

The + operator can also be used to add string variables or text values together. To add two or more string variables together, use the + operator.

txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
The result of txt3 will be:
What a very nice day

JavaScript Comparison and Logical Operators.

Comparison and Logical operators are used to test for *true* or *false*.

Comparison operators are used in logical statements to determine equality or difference between variables or values. They include == , != , > , < , >= , <= ,

Logical operators are used to determine the logic between variables or values. They include && , || , !

Conditional Operator.

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

variablename=(condition)?value1:value2

Conditional Statements.

In JavaScript 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 the condition is true and another code if the condition is false.
- if...else if....else statement - use this statement to select one of many blocks of code to be executed.
- switch statement - use this statement to select one of many blocks of code to be executed.

If Statement.

```
if (time<20)
  {
  x="Good day";
  }
```

If...else Statement.

```
if (time<20)
  {
  x="Good day";
  }
else
  {
  x="Good evening";
  }
```

If...else if...else Statement.

```
if (time<10)
  {
  x="Good morning";
  }
else if (time<20)
  {
  x="Good day";
  }
else
  {
  x="Good evening";
  }
```

JavaScript Switch Statement.

```
var day=new Date().getDay();
switch (day)
{
case 0:
  x="Today it's Sunday";   break;
case 1:
  x="Today it's Monday";   break;
case 2:
  x="Today it's Tuesday";   break;
case 3:
  x="Today it's Wednesday";   break;
case 4:
  x="Today it's Thursday";   break;
case 5:
  x="Today it's Friday";   break;
case 6:
  x="Today it's Saturday";   break;
default:
        "Have a good day"; break;
}
```

JavaScript Loops.

In JavaScript, there are two different kind of loops:

- for - loops through a block of code a specified number of times
- while - loops through a block of code while a specified condition is true

The for Loop.

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

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

The while Loop.

The while loop loops through a block of code while a specified condition is true.

```
while (i<5)
 {
 x=x + "The number is " + i + "<br />";
 i++;
 }
```

The do...while Loop.

The do...while loop is a variant of the while loop. This loop will execute the block of code once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
do
 {
 x=x + "The number is " + i + "<br />";
 i++;
 }
while (i<5);
```

JavaScript Break and Continue Statements.

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

```
for (i=0;i<10;i++)
 {
 if (i==3)
  {
  break;
  }
 x=x + "The number is " + i + "<br />";
 }
```

The continue statement will break the current iteration and continue the loop with the next value.

```
for (i=0;i<=10;i++)
 {
 if (i==3)
  {
  continue;
  }
 x=x + "The number is " + i + "<br />";
 }
```

JavaScript For...In Statement.

The for...in statement loops through the properties of an object.

```
var person={fname:"John",lname:"Doe",age:25};
for (x in person)
  {
  txt=txt + person[x];
  }
```

JavaScript Try...Catch and Throw Statement.

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

```
<!DOCTYPE html>
<html>
<head>
<script>
var txt="";
function message()
{
try
 {
  adddlert("Welcome guest!");
  }
catch(err)
 {
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.message + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
 }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()"
/>
</body>
</html>
```

The throw statement allows you to create an exception. The exception can be a string, integer, Boolean or an object.

```
<!DOCTYPE html>
<html>
<body>
<script>
var x=prompt("Enter a number between 5 and 10:","");
try
 {
 if(x>10)
 {
  throw "Err1";
```

```
  }
 else if(x<5)
  {
  throw "Err2";
  }

 else if(isNaN(x))
  {
  throw "Err3";
 }
}
catch(err)
 {
  if(err=="Err1")
 {
   document.write("Error! The value is too high.");
 }
  if(err=="Err2")
  {
   document.write("Error! The value is too low.");
   }
  if(err=="Err3")
   {
   document.write("Error! The value is not a number.");
   }
  }
</script>
</body>
</html>
```

JavaScript Special Characters.

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

\' single quote
\" double quote
\\ backslash
\n new line
\r carriage return
\t tab
\b backspace

JavaScript Popup Boxes.

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box

Alert Box.

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

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
```

```
{
alert("I am an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="myFunction()" value="Show alert box" />
</body>
</html>
```

Confirm Box.

A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

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

Prompt Box.

A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

```
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  x="Hello " + name + "! How are you today?";
}
```

To display line breaks inside a popup box, use a back-slash followed by the character n.

```
alert("Hello\nHow are you?");
```

JavaScript Events.

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript. Every element on a web page has certain events which can trigger a JavaScript. For example, Clicking a button (or any other HTML element), A page is finished loading, An image is finished loading, Moving the mouse-cursor over an element, Entering an input field, Submitting a form, A keystroke. Events are normally used in combination with functions.

HTML Event Attributes.

To assign events to HTML elements we can use the event attributes.

To assign an onclick event to a button element:

<button id="myBtn" onclick="displayDate()">Try it</button>

onload and onunload.

The onload and onunload events are triggered when the user enters or leaves the page. The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information. The onload and onunload events can be used to deal with cookies.

<body onload="checkCookies()" />

onfocus, onblur and onchange.

The onfocus, onblur and onchange events are often used in combination with validation of form fields.

<input type="text" onchange="checkEmail()" />

onsubmit.

The onsubmit event can be used to validate form fields before submitting it.

<form method="post" action="mySubmit.asp" onsubmit="checkForm()">

onmouseover, onmouseout.

The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element.

onmousedown, onmouseup and onclick.

The onmousedown, onmouseup, and onclick events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

JavaScript Objects.

JavaScript is an Object Based Programming language, and allows us to define our own objects and make our own variable types. An object is just a special kind of data. An object has properties and methods.

Properties: Properties are the values associated with an object.

```
<script>
var txt="Hello World!";
document.write(txt.length);
</script>
```

Methods: Methods are the actions that can be performed on objects.

```
<script>
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

JavaScript String Object.

The String object is used to manipulate a stored piece of text.

```
var txt="Hello world!";
document.write(txt.length); document.write(txt.toUpperCase());
```

JavaScript Date Object.

The Date object is used to work with dates and times. The date object can also be used with comparison statements. Date objects are created with the Date() constructor. There are four ways of initiating a date:

```
new Date() // current date and time
new Date(milliseconds) //milliseconds since 1970/01/01
new Date(dateString)
new Date(year, month, day, hours, minutes, seconds, milliseconds)
```

We can easily manipulate the date by using the methods available for the Date object.

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
myDate.setDate(myDate.getDate()+5);
```

JavaScript Array Object.

An array is a special variable, which can hold more than one value, at a time. An array can be defined in three ways.

```
var myCars=new Array(); // regular array (add an optional integer
myCars[0]="Saab";       // argument to control array's size)
myCars[1]="Volvo";
myCars[2]="BMW";
```

OR

```
var myCars=["Saab","Volvo","BMW"]; // literal array
```

OR

```
var myCars=new Array("Saab","Volvo","BMW"); // condensed array
```

JavaScript Boolean Object.

The Boolean object represents two values: "true" or "false".

```
var myBoolean=new Boolean();
```

JavaScript Math Object.

The Math object allows you to perform mathematical tasks. The Math object includes several mathematical constants and methods.

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

Mathematical Constants.

JavaScript provides eight mathematical constants that can be accessed from the Math object. These are:

Math.E	*Math.LN2*
Math.PI	*Math.LN10*
Math.SQRT2	*Math.LOG2E*
Math.SQRT1_2	*Math.LOG10E*

JavaScript Browser Detection.

The Navigator object contains information about the visitor's browser name, version, and more.

```
<div id="example"></div>
<script>
txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
document.getElementById("example").innerHTML=txt;
</script>
```

JavaScript Cookies.

A cookie is a variable that is stored on the visitor's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With JavaScript, we can both create and retrieve cookie values.

Name cookie - The first time a visitor arrives to a web page, he or she must fill in her/his name. The name is then stored in a cookie. Next time the visitor arrives at a page, he or she could get a welcome message like "Welcome John Doe!" The name is retrieved from the stored cookie.

Date cookie - The first time a visitor arrives to a web page, the current date is stored in a cookie. Next time the visitor arrives at the page, he or she could get a message like "Your last visit was on Tuesday August 11, 2012!" The date is retrieved from the stored cookie

```
<!DOCTYPE html>
<html>
<head>
<script>
function getCookie(c_name)
{
var i,x,y,ARRcookies=document.cookie.split(";");
for (i=0;i<ARRcookies.length;i++)
{
  x=ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));
  y=ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);
  x=x.replace(/^\s+|\s+$/g,"");
  if (x==c_name)
 {
    return unescape(y);
    }

  }
}
function setCookie(c_name,value,exdays)
{
var exdate=new Date();
```

```
exdate.setDate(exdate.getDate() + exdays);
var c_value=escape(value) + ((exdays==null) ? "" : "; expires="+exdate.toUTCString());
document.cookie=c_name + "=" + c_value;
}

function checkCookie()
{
var username=getCookie("username");
if (username!=null && username!="")
  {
  alert("Welcome again " + username);
  }
else
  {
  username=prompt("Please enter your name:","");
  if (username!=null && username!="")
    {
    setCookie("username",username,365);
    }
  }
}
</script>
</head>
<body onload="checkCookie()">
</body>
</html>
```

JavaScript Form Validation.

JavaScript can be used to validate data in HTML forms before sending off the content to a server. Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Required Fields.

The function below checks if a field has been left empty. If the field is blank, an alert box alerts a message, the function returns false, and the form will not be submitted:

```
function validateForm()
{
var x=document.forms["myForm"]["fname"].value;
if (x==null || x=="")
  {
  alert("First name must be filled out");
  return false;
  }
}
```

The function above could be called when a form is submitted:

```
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm()" method="post">
First name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

E-mail Validation.

The function below checks if the content has the general syntax of an email. This means that the input data must contain an @ sign and at least one dot (.). Also, the @ must not be the first character of the email address, and the last dot must be present after the @ sign, and minimum 2 characters before the end:

```
function validateForm()
{
var x=document.forms["myForm"]["email"].value;
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
  {
  alert("Not a valid e-mail address");
  return false;
  }
}
```

The function above could be called when a form is submitted:

```
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm();" method="post">
Email: <input type="text" name="email">
<input type="submit" value="Submit">
</form>
```

JavaScript Timing Events.

With JavaScript, it is possible to execute some code at specified time-intervals. This is called timing events. It's very easy to time events in JavaScript. The two key methods that are used are:

- setInterval() - executes a function, over and over again, at specified time intervals

  setInterval("javascript function",milliseconds);
- setTimeout() - executes a function, once, after waiting a specified number of milliseconds

  setTimeout("javascript function",milliseconds);

The clearInterval() method is used to stop further executions of the function specified in the setInterval() method.

clearInterval(intervalVariable)

The clearTimeout() method is used to stop the execution of the function specified in the setTimeout() method.

clearTimeout(timeoutVariable)

To be able to use the clearInterval() and clearTimeout() method, you must use a global variable when creating the timeout method

Exercise:

1. Define a function max() that takes two numbers as arguments and returns the largest of them. Use the if-else construct.

2. Write a function that takes a single character and returns true of it is vowel and false otherwise.

3. Define a function sum() and multiply() that sums and multiplies all numbers in an array. E.g; sum([1,2,3,4]) should return 10 and multiply([1,2,3,4]) should return 24.

4. Define a function reverse() that reverses the string supplied to it.

5. Create a HTML form that has number of textboxes when the form runs in the browser fill the textboxes with data. Write a script that verifies that all the text boxes have been filled. If a textbox is left empty pop up an alert indicating which box has been left empty. When the OK button of alert is clicked, the focus must be transferred to that specific textbox otherwise display THANK YOU.

Lab # 7

Object:
Introduction to ASP.NET.

Theory:
ASP.NET is a server side scripting technology that enables scripts (embedded in web pages) to be executed by an Internet server. It is a program that runs inside the windows server component IIS (Internet Information Services). An ASP.NET file can contain HTML, XML, and scripts. Scripts in an ASP.NET file are executed on the server. An ASP.NET file has the file extension ".aspx".

- When an ASP is incorporated into Web site, the following step happens:
- The user brings up a Web site where the default page has the extension .aspx.
- The browser requests the ASP file from the Web server, instead of processing itself like in .html extension.
- The server-side script begins to run with ASP.
- ASP processes the requested file sequentially (top-down), executes any script commands contained in the file, and produces an HTML Web page.
- The result, which is 100% pure HTML code, is sent back to the browser. Because the script runs on the server, the Web server does all of the processing and standard HTML pages can be generated and sent to the browser. This means that the Web pages are limited only by what your Web server supports. Another benefit of having your script resides on the server is that the user cannot “view source” on the original script and code. Instead, the user sees only the generated HTML as well as non-HTML content, such as XML, on the pages that are being viewed.

The ASP .Net code can either be written inside the ASPX page or it can be included as a .cs or vb .net file. When the ASPX page is embedded with the code of either C# or VB .Net, the ASP .Net run time automatically compiles the code into an assembly and loads it. If the code is kept in a separate source file either as a VB or C# file, it has to be compiled by the programmer, which will be used by the run time for further execution.

Creating A New Website.
Open the Microsoft Visual Studio 2010. Go to File New Website the following dialog box appears.

The website can be built either using Visual C#, Visual Basic. Select the default ASP.Net Website from the template and type in the name for the project after selecting the language. Now a new blank website is created.

The default.aspx is included in the project. To rename it, right click on the Default.aspx select Rename from the popup menu and then name it to First.aspx.

Source View.
Source view displays the HTML markup for the Web page, which you can edit. By default, all HTML elements and scripts are displayed when the Source View is initially

selected. Elements can be dragged from the Toolbox, just as they are dragged when editing a Web page in Design view, and then see their markup inserted into the document. To select Source view, click the Source tab at the bottom of the HTML Designer window.

Design View.
Design view displays ASP.NET Web pages, master pages, content pages, HTML pages, and user controls. It allows you to add text and elements and then position and size them and set their properties using special menus or the Properties window. To select Design view, click the Design tab at the bottom of the HTML Designer window.

Properties Window.
It's a window where you can edit properties or events for the controls. To access Properties Window, Select Properties Window on the View menu.

Solution Explorer.
Solution Explorer provides you with an organized view of your projects and their files as well as ready access to the commands that pertain to them. A toolbar associated with this window offers commonly used commands for the item you highlight in the list. To access Solution Explorer, select Solution Explorer on the View menu

Toolbox's Auto Hide Option
The toolbox navigator is present on the left end of the screen. If the mouse is scrolled over it, the toolbox appears presenting many control options that can be added to a website. The auto hide option presented by icon can be clicked to make the toolbox locked on the left side of the screen turning the icon to icon.

Move the control on the screen.
To drag it to the desired position on the page, go to Format - Set Position□Absolute. This option can also be applied to the controls using Cascading Style Sheets. The Cascading Style Sheets or CSS allow you to control the layout and look of your page easily.// CSS tags or properties are easy to use and affect the look and feel or style of your pages by setting its Property: CSS Class to the already defined Style Sheet.

Adding some controls on the website.
Double click on the Button icon present in the Standard toolbox to add it to the website. Drag to the desired location. The default ID for this Button is Button1. To change this ID go to properties window, type the ID as OK Button. Also change the Text property of this control to OK. Drag a label on the website and remove the text label from its properties. Double click the OKButton to open its code and add the following code:

```
protected void OKBotton_Click(object sender, EventArgs e)
{
Label1.Text = "Hello User" ;
}
Use Ctrl+Shift+S to save the project
```

Debugging the Website.
Press F5 or the play button in the standard toolbar to start debugging. A window displays a message that Debugging Not Enabled. Select the option Add web.config file with debugging enabled and click OK. The Visual Studio Web Developer 2010 launches a mini testing web server that will work for the purpose of testing the website just created

locally. A popup on the taskbar shows ASP.Net Development Server http://localhost:50701/First/. The website is now open on the website explorer. When the OK button is clicked, a welcome message, Hello User is displayed on the website

To make it more user friendly, add a textbox and change its ID to NameTextBox. Add a Label on its left side and change its text to "Enter Your Name:". Add the following code to OKButton:

```
protected void OKBotton_Click(object sender, EventArgs e)
{
Label1.Text = "Hello " + NameTextBox.Text ;
}
```

Where + is used for string concatenation.

Adding a Hyperlink.

From the toolbox drag a Hyperlink control to the webpage. Change its Text property to **NED University** and its NavigateUrl property to **http://www.neduet.edu.pk.** To open this link in a new browser, change its Target property to _blank.

Exercise:

1. Write down the advantages of server side scripting in ASP.Net

.

Lab # 8

Object:
Exploring some controls and the concept of field validation in ASP.NET.

Theory:
Add a New Web Form to the Project.
A new web form can be added to the project. To do so open Solution Explorer right click on the root of the project and click on Add New item… A number of different types of templates are displayed for the selection of the developer. Select Web Form and give this new item an appropriate name.

Two check boxes at the bottom of the page allow the separation of the code and the selection of a master page. The restructured code-behind schema of Visual Studio 2008 supports separation of code but makes it optional. The language selected for the project is displayed in the language box.

Once a new page is open, select Default.aspx → Set as Start page to enable the debugger to run this page whenever the project is debugged.

Adding a Table.
To insert a table on a webpage, click Table → Insert Table. An Insert Table window pops up enabling a user to select from the available templates or creating a customized table with required number of cells and cell formatting.

Add a Dropdown List.
Drag the Drop down list from the standard toolbox. Change its **ID** to **DeptDropDown**. Click on the arrow on upper right hand corner of the control to see the various options available. Select the **Edit Items** to add a number of options from which the user can select. The ListItem Collection Editor opens in a pop up window. Click Add to add a new entry in the drop down list and type in the name of the entry in the Text property. Click Add to enter another option or OK to exit.

Adding Radio Buttons.
To add radio buttons on a webpage, drag the number of radio buttons required (here consider two radio buttons are being dragged) from the standard toolbox. Rename them as **studentRadioButton** and **facultyRadioButton**. Also change their **text** property to Student and faculty respectively. In the **GroupName** property, write **User** for both of the radio buttons which makes them to belong to same group.

Field Validation.
Validation controls are available in the toolbox. Some of these are discussed below:

Required Field Validator.
Required field validator is used for required fields. Fields that cannot be left empty is validated using the required field validator. A good example will be a TextBox which is used for taking the username as the input. Since the user must enter the username in the TextBox in order to access the website. The required field validator can be used with the TextBox control to validate the input. If user does not enter any data in the TextBox than an error message is displayed and further processing of the request will be stopped.

In order to set the required validator on the TextBox control just drag and drop the validator on the webform and set its ControltoValidate property to the TextBox id that is to be validated.

Regular Expression Validator.

Regular Expression Validator is used to check the user input against some built in regular expressions. The control provides the RegularExpression collection property that can be used to select the desired Regular Expression. Some of the built in expressions are Email, postal code, telephone number and many more. If the user input does not match the regular expression expected an error will occur and the request for the resources will not be authorized unless the user corrects the input data.

Range Validator.

The Range Validator control is used to check whether the input control contains value in the specified range. You can check the range of values against different data types such as String, Date, Integer and so on.

The two most important properties of the range validator control is the Maximum value and the minimum value. The range can be used to restrict the user data. Suppose if the user is entering a roll number so the range can be set from 1 to 500.

Compare Validator.

The compare validator control is used to compare the input server control's value. The compare validator can be used to compare against a value or another control. If both the ControlToCompare and ValueToCompare properties are set for a CompareValidator control, the ControlToCompare property takes precedence.

It can be used to compare whether the user has selected a value from the dropdown list or not. For this purpose, select the dropdown list from **ControlToValidate** property and in **ValueToCompare** write down the string e.g. Select a Department (which must be the first entity in the dropdown list) and set the **Operator** property to **NotEqual**.

Another good use of the compare validator is to check whether the passwords entered by the user in the two TextBoxes while registering for a website are same or not. This validator also performs validation on the client side as well as the server side. So, no postback will be required when doing the comparison and hence resources will be saved and performance will increase.

.

Exercise:

1. Design a Login.aspx Page and apply field validations on it.

Lab # 9

Object:
Exploring the working of Login controls and some basic Postback controls in ASP.Net.

Theory:
ASP.Net Website Administration Tool.
In ASP.Net, a login page can be inserted very easily by a series of mouse clicks i.e. no time is spent in writing extensive code for the authentication purposes. Open the ASP.Net Configuration from the Website menu to view the website administration tools. This option provides, amongst other facilities, the ability to update the security settings for a website. The figure 1 shown below will appear when the security tab is selected.

Various options are available here, through which new user can be created or an existing user can be edited. Roles of the users can be created and the access rules can also be created or edited.

To create a new role, click on the Enable Roles option and then click on the Create or manage roles link. Enter the name of the role (e.g. Administrator or Premium User) and click OK. Now once a role is created, its users can be defined and the access rules for the user of this role can be set. To add a new user, first set the authentication type to From the Internet. Select the Create User link and fill up the form for a new user. Make sure to enter a password with having minimum 7 characters including a non-alphanumeric character. Close this explorer window and return to the Visual Studio environment

Creating a login page.
Add a Web form to the project and name it Login.aspx. In its design view, drag the Login control from the Login Toolbox. From the Smart Tag, this control can be formatted or the text boxes or the labels present in it can be edited too. In the DestinationPageUrl property, write down the name of the webpage which the user can visit after logging in. On the Home Page, a login hyperlink can be created which will link to this login page or set the login page as the start page.

The User login name and status can also be displayed in the webpage. Drag the Login Status and Login Name control on the webpage to do so. The Change Password control can be used to change the existing user‟s password. A new can sign up if the Create user Wizard is added to the webpage

What is PostBack?
A Postback is an action taken by an interactive webpage, when the entire page and its contents are sent to the server for processing some information and then, the server post the same page back to the browser. This is done to verify user names and passwords, for example, when processing an on-line order form data, or other similar tasks that require server interaction.

Each Asp.Net page, when loaded, goes through a regular creation and destruction cycle like Initialization, Page load etc., in the beginning and unload while closing it. This Postback is a read only property with each Asp.Net Page (System.Web.UI.Page) class. This is false when the first time the page is loaded and is true when the page is submitted

and processed. This enables users to write the code depending on if the PostBack is true or false (with the use of the function Page.IsPostBack).

Working on a PostBack event.

For some WebPages it may be required to fill up the entities in a drop down list depending upon selection made by user. For instance, the user is filling up a form that asks for the country and city the user belongs. Instead of typing in the country, the user can simple select from the drop down list. As soon as the country is selected, the drop down list of the city becomes populated with the cities of that country. This action is taken in response of the user‟s action and is made possible by the PostBack event. Add a textbox, a button and label to the webpage and write the following code:

```
protected void Page_Load(object sender, EventArgs e)
{
if (Page.IsPostBack == false)
{
TextBox1.Text = "";
}
}
protected void Button1_Click(object sender, EventArgs e)
{
Label1.Text = TextBox1.Text;
}
```

This code helps the user to enter some text in the textbox and when the button is clicked it copies that text to the label. If you want to make sure that the textbox is empty at the time the page is first time loaded then you write TextBox1.Text = "";. But this will empty the textbox as well as the label whenever the button is clicked. To empty the contents of the textbox for only the first time the condition if (Page.IsPostBack == false)is inserted. The IsPostBack is a property that is false only when the page is loaded for the first time.

Adding a Calendar Control.

Whenever it is required to take date as input from the user, a useful control called Calendar can be added from the standard toolbox. To illustrate its working, add a textbox and set its ReadOnly property to True. Also change its ID to dateTextbox. Double click on the control to open its coding and add the following code:

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
dateTextBox.Text = Calendar1.SelectedDate.ToString("MM/dd/yyyy");
}
```

The date clicked by the user is Postback and is displayed in the textbox which is uneditable.

Lab # 10

Object:
Working with Databases in ASP.Net

Theory:
Microsoft provides a tool window in Visual Studio 2008 IDE called the "Server Explorer" which is a management console for any server (or system) and is used to open databases, manage and control the data of different databases, manage system services, and so on. Whenever a project is opened in ASP.Net the server explorer is also visible often tabbed with the Solution explorer

ActiveX Data Objects.
ADO.NET is a set of classes that expose data access services to the .NET programmer. ADO.NET provides functionality to developers writing managed code, consistent access to data sources such as Microsoft SQL Server, as well as data sources exposed through OLE DB and XML. Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data. Its primary objects are:

DataSet objects are in-memory representations of data. They contain multiple Datatable objects, which contain columns and rows, just like normal database tables. You can even define relations between tables to create parent-child relationships. The DataSet is specifically designed to help manage data in memory and to support disconnected operations on data.

Connection object creates the connection to the database. Microsoft Visual Studio .NET provides two types of Connection classes: the SqlConnection object, which is designed specifically to connect to Microsoft SQL Server, and the OleDbConnection object, which can provide connections to a wide range of database types like Microsoft Access and Oracle.

Command objects are used to execute commands to a database across a data connection. They provide three methods that are used to execute SQL commands on the database:

ExecuteNonQuery Executes commands that have no return values such as INSERT, UPDATE or DELETE.

ExecuteScalar Returns a single value from a database query.

ExecuteReader Returns a result set by way of a DataReader object.

DataReader object provides a forward-only, read-only, connected stream record set from a database. It allows you to obtain the results of a SELECT statement from a command object.

DataAdapter object contains a reference to the connection object and opens and closes the connection automatically when reading from or writing to the database. Additionally, the data adapter contains command object references for SELECT, INSERT, UPDATE, and DELETE operations on the data

Working with SQL Database.

Creating a Database.

Right click on the root directory on the Solution Explorer window and select Add New Item. From the pop-up window select the SQL Database and assign a name to the database (e.g. StudentDB.mdf). To add a new table click on Server Explorer then right click the Tables Option shown beneath the database filename and click Add New Table. Insert all the column names of the database along with their data types in the database. Also rename the table with an appropriate name.

Connecting to Database.

Drag a SQL Data source from the Data Toolbox into the webpage. Click on the arrow present on the upper right corner of the Data Source (known as Smart tag) as shown in figure 2. Select Configure Data Source to make a new connection. A popup window will appear asking the user to choose the database from a drop down list. Select the desired database and click Next. After passing through a series of steps in which the connection string will be saved and also the required columns in the table are selected, the database connection is established.

Displaying the Data on the Webpage.

To view the data present in a table on the webpage, drag the table to be shown on to the design window or drag the GridView or DetailsView in the data Toolbox and click on the arrow on its upper right corner and in choose the data source option, select the desired SQLDataSource.

Working with SQL database using ADO.NET.

Creating a Database.

Right click on the root directory on the Solution Explorer window and select Add New Item. From the pop-up window select the SQL Database and assign a name to the database (e.g. ItemDB.mdf). To add a new table click on Server Explorer then right click the Tables Option shown beneath the database filename and click Add New Table. Insert all the column names of the database along with their data types in the database. Also rename the table with an appropriate name.

Consider an example to create the columns with name “ID” and “CategoryName” and select appropriate data types. Select the column ID as primary key. Also rename the table as “Categories”.

To display the database records.

The SqlConnection object holds the connection string that connects the database engine to the ItemDB.mdf database. To provide appropriate connection string to this connection object, drag a SQL Data source from the Data Toolbox into the webpage. Select Configure Data Source to make a new connection. A popup window will appear asking the user to choose the database from a drop down list. Select the desired database. Open the ConnectionString tab below to get the required string. This connection string is to be copied in the code.

The SqlCommand.ExecuteReader( ) method creates an SqlDataReader object to read these records. The DataGrid is connected to the data reader through its DataGrid.DataSource property.

When the DataGrid.DataBind( ) method executes, database records are moved from the database to the DataGrid, which displays them one record per row.

From the toolbox, drag a DataGrid to the Web form. Switch to code view by double-clicking the form. Add this line to the using statements at the beginning of Default.aspx.cs.

```
using System.Data.SqlClient;
```

Insert this code into the Page Load method:

```
private void Page_Load(object sender, System.EventArgs e)
{
if(!IsPostBack)
ReadData( );
}
```

Add the ReadData method just after the Page Load method

```
public void ReadData() {
SqlDataReader rdr = null;
SqlConnection conn = null; try {
conn = new SqlConnection("Data Source=(local);Initial Catalog=Northwind;
Integrated Security=SSPI");
// Open the connection
conn.Open();
 // 1. Instantiate a new command with a query and connection
SqlCommand cmd = new SqlCommand("select CategoryName from Categories",
conn);
// 2. Call Execute reader to get query results
rdr = cmd.ExecuteReader();
Datagrid1.DataSource = rdr;
Datagrid1.DataBind ();
// print the CategoryName of each record
while (rdr.Read()) { Console.WriteLine(rdr[0]); } }
 finally {
 // close the reader
 if (rdr != null) { rdr.Close(); } // Close the connection
if (conn != null) { conn.Close(); } } }
```

Press F5 to launch the Web application under the debugger. The contents of the database should appear in the browser.