2.6 DHTML WITH JAVASCRIPT

- One of the most popular uses of JavaScript is DHTML (Dynamic HyperText Markup Language).
- DHTML is the combination of HTML and JavaScript.DHTML is using JavaScript to modify the CSS styles of HTML elements.
- DHTML is the combination of several built-in browser features in fourth generation browsers that enable a web page to be more dynamic.
- The HTML document acts as a reference to the DHTML. The DHTML can change the visibility, position, contents, background colour, z-index, clipping, size of the already positioned element. New elements can also be added to the HTML document.

Differences between HTML and DHTML

HTML	DHTML
A plain page without any styles and Scripts called as HTML.	A page with HTML, CSS, DOM and Scripts called as DHTML.
HTML sites will be slow upon client-side technologies.	DHTML sites will be fast enough upon client-side technologies.

HTML stands for only static pages. It is referred as a static HTML and static in nature.	DHTML is Dynamic HTML means HTML+JavaScript. Hence it is referred as a dynamic HTML.

Components of DHTML

Dynamic HTML includes the following components: HTML, Cascading Style Sheets, Scripting and the Document Object Model.

- **HTML**:
  - HTML defines the structure of a Web page, using such basic elements as headings, forms, tables, paragraphs and links.
- **Cascading Style Sheets (CSS)**:
  - A style sheet controls the formatting of HTML elements.
  - Style sheets are used to specify page margins, point sizes and leading.
  - Cascading Style Sheets is a method to determine precedence and to resolve conflicts when multiple styles are used.
- **Scripting**:
  - Scripting provides the mechanisms to interpret user actions and produce client-side changes to a page.
  - DHTML can communicate with several scripting languages but JavaScript is widely used.
- **Document Object Model (DOM)**:
  - The DOM outlines Web page content in a way that makes it possible for HTML elements, style sheets and scripting languages to interact with each other.
  - The W3C defines the DOM as a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure, and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented stage.

Positioning elements in DHTML

- There are two types of positioning: absolute and relative.
- **Absolute positioning** allows to place an element anywhere in relation to the page.
- **Relative positions** the element based on offset values.
- Most DHTML is done with absolutely positioned elements. Relatively positioned elements do not accept the 'clip' style and do not allow their clipping to be changed.

JavaScript and Cascading Style Sheets (CSS)

- Cascading Style Sheets are the standard way to define the presentation of the DHTML pages, from fonts and colors to the complete layout of a page. They are much more efficient than using HTML.
- CSS files are termed "cascading" stylesheets because of two reasons: one style sheet can cascade, or have influence over, multiple pages. Similarly, many CSS files can define a single page.
- CSS is the most important feature of DHTML. The CSS has various style properties.There are 3 ways to implement css commands into the site:

    1. Use one CSS file for all pages.
    2. Integrate CSS commands into the head of each of the documents.
    3. Use the style attribute to put CSS code directly into a HTML element.

Common style properties of CSS

1. **Position** -Specifies how the block should be positioned on the page with respect to other page elements.
    - position:absolute- Block is positioned absolutely within the browser window,relative to <BODY> block.
    - position:relative- Block is positioned relative to its parent block, if any, or else normal flow of page.
    - position:static- Block is positioned according to standard HTML layout rules.
2. **width**-Specifies the width at which the block's contents should wrap. Width may be in measured units (50px), as a percentage of the parent block's width (50%), or auto which wraps the block according to its parent's width.

    Examples: width:50px or width:50%
3. **height**-Specifies the height of the block, measured in units (50px), percentage of the parent block's height (50%), or auto. The height of the block will be forced to the minimum necessary to display its contents.

    Examples: height:50px or height:50%
4. **left**-Specifies the offset of the left edge of the block in accordance with the position attribute. Positive measures (5px) are offset towards the right while negative measures (-5px) are offset towards the left.

    Examples: left:5px or left:-5px

5. **top-** Specified the offset from the top edge of the block in accordance with the position attribute. Positive measures (5px) are offset towards the bottom of the page while negative measures (-5px) are offset towards the top of the page.

   Examples: top:10px or top:-10px

6. **clip-**Specifies a rectangular portion of the block which is visible. The syntax of this property is different for different browsers.

   1. MSIE: clip:rect(top right bottom left)

      Example:clip:rect(0px 30px 50px 0px)

   2. Netscape: clip:rect(left,top,right,bottom)

      Example:clip:rect(0,0,30,50)

7. **visibility-**Specifies whether a block should be visible. If not visible, the block will notappear on the page, although you can make it visible later using JavaScript. The possible values for this property vary between browsers.

   1. MSIE

      visibility:inherit- Block inherits the visibility property of its parent.

      visibility:visible- Block is visible.

      visibility:hidden-Block is hidden or invisible

   2. Netscape:

      visibility:inherit- Block inherits the visibility property of its parent.

      visibility:show- Block is visible.

      visibility:hide- Block is invisible.

8. **z-index-** Specifies the "**stacking order**" of blocks, should they happen to overlap other positioned blocks. A block is assigned a z-index, which is any integer. When blocks overlap, that which has the greater positive z-index appears above a block with a lower z-index. Blocks with an equal z-index value are stacked according to the order in which they appear in the source code (bottom-to-top: first block defined appears on bottom, last block defined appears on top).

   Example: z-index:3

9. **backgroundcolor-**Specifies the background color for the block.

   Example:background-color:green or background-color:FF8F00

10. **backgroundimage**-Specifies a background image for the block.

    Example: background-image:url('images/cat.jpg')

Advantages of CSS

- Pages download faster.Less code and the pages are shorter and neater.
- The look of the site is kept consistent throughout all the pages that work off the same stylesheet.
- Updating the design and general site maintenance are made much easier, and errors caused by editing multiple HTML pages occur far less often.
- The ID field is most important, because id will be used to reference the positioned element. Browsers call these positioned elements as **layers.**

Grouping elements in CSS:

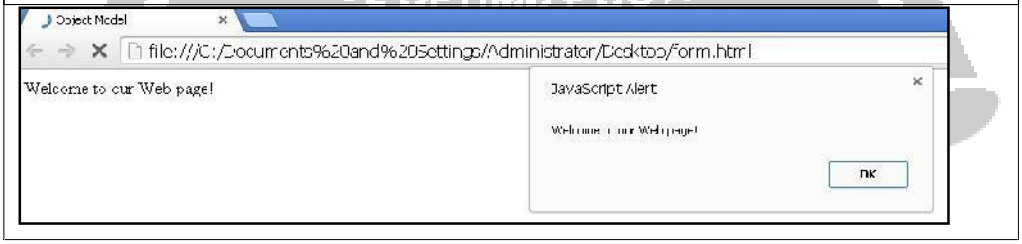
1. : The element is what you could call a neutral element which does not add anything to the document itself. But with CSS, can be used to add visual features to specific parts of text in your documents.
2. :Whereas is used within a block-level element as seen in the previous example, is used to group one or more block-level elements.

Object Referencing

- The simplest way to reference an element in a DHTML document is by using the element's id attribute. The element is represented as an object, and its various XHTML attributes become properties that can be manipulated by scripting.

Changing text using DHTML

```
<html><head><title>Object  Model</title>
<script type = "text/javascript">
 function start()
 {alert(  pText.innerText  );
Text.innerText = "Thanks for coming.";}
</script></head>
<body onload = "start()">
<p id = "pText">Welcome to our Web page!</p>
</body></html>
```



- The onload calls JavaScript start function when document loading completes.Start() displays an alert box containing the value of pText.innerText.
- The object pText refers to the p element whose id is set to pText. The **innerText property** of the object refers to the text contained in that element in this example it is Welcome to our Web page!.
- The start() sets the innerText property of pText to a different value. Changing the text displayed on screen is a Dynamic HTML ability called **dynamic content.**

Collections all and children

- **Collections** are arrays of related objects on a page. Collections provide an easy way of referring to any specific element without an id.
- There are several special collections in the object model.The **all collection** contains all the XHTML elements in a document.
- The **length property** of the collection specifies the size of the collection. The **children** collection of a specific element contains that element's child elements. For example, an html element has only two children—the head element and the body element.

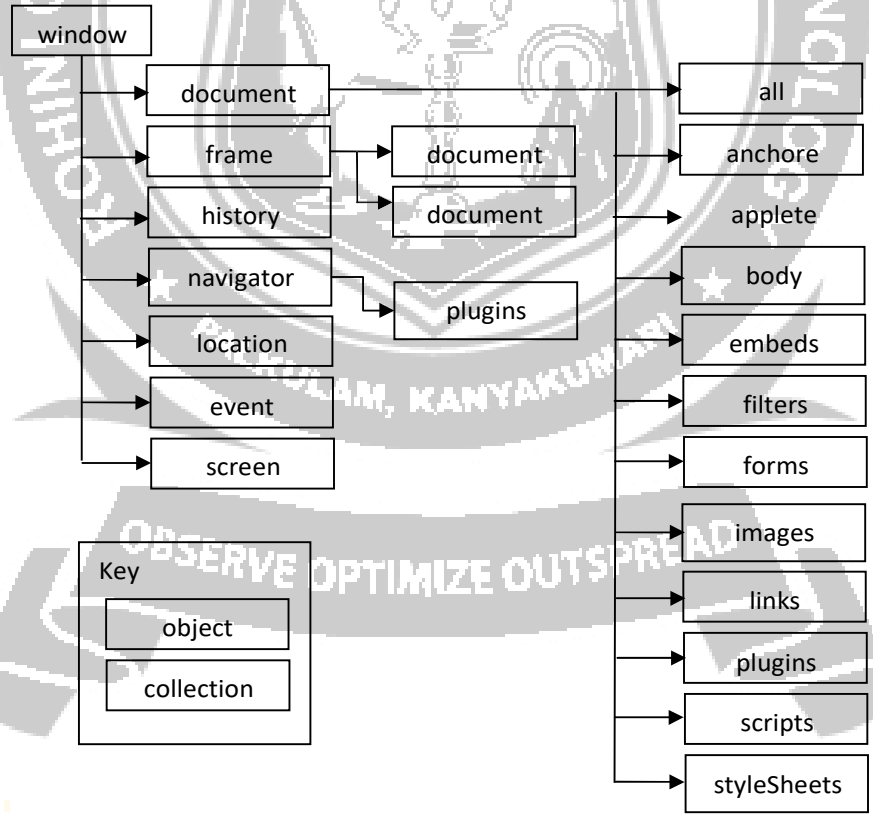


Figure 2.2 Object collections

Object	Description
window	This object represents the browser window and provides access to the document object contained in the window. If the window contains frames, a separate window object is created automatically for each frame, to provide access to the document rendered in that frame. Frames are considered to be subwindows in the browser.
document	This object represents the XHTML document rendered in a window.The document object provides access to every element in the XHTMLdocument and allows dynamic modification of the XHTML document.body This object provides access to the body element of an XHTML document.
history	This object keeps track of the sites visited by the browser user. The object provides a script programmer with the ability to move forward and backward through the visited sites, but for security reasons does not allow the actual site URLs to be manipulated.
navigator	This object contains information about the Web browser, such as the name of the browser, the version of the browser etc.
location	This object contains the URL of the loaded document. When this object is set to a new URL, the browser immediately switches (navigates) to the new location.
Event	This object can be used in an event handler to obtain information about the event that occurred.
screen	The object contains information about the computer screen for the computer on which the browser is running. Information such as the width and height of the screen in pixels can be used to determine the size at which elements should be rendered in a Web page.

Collections	Descriptions
all	Many objects have an all collection that provides access to every element contained in the object. For example, the body object's all collection provides access to every element in the body element of an XHTML document.
anchors	This collection contains all anchor elements (a) that have a name or id attribute. The elements appear in the collection in the order they were defined in the XHTML document.
applets	This collection contains all the applet elements in the XHTML document. Currently, the most common applet elements are Java applets.

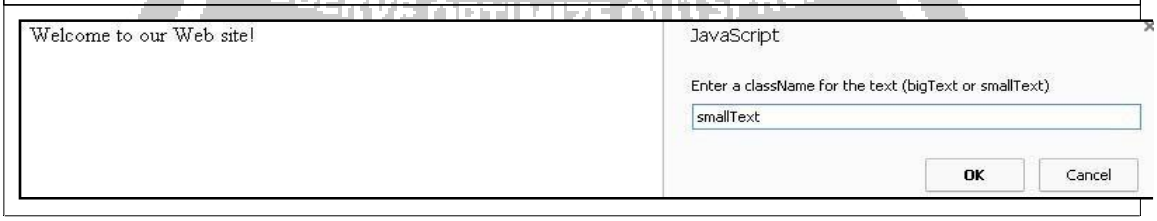
embeds	This collection contains all the embed elements in the XHTML document.
forms	This collection contains all the form elements in the XHTML document. The elements appear in the collection in the order they were defined in the XHTML document.
frames	This collection contains window objects that represent each frame in the browser window. Each frame is treated as its own subwindow.

Dynamic Styles

An element's style can be changed dynamically. Often such a change is made in response to user events.The Dynamic HTML object model also allows to change the **class** attribute of an element—instead of changing many individual styles at a time.

Dynamic styles

```
<html ><head><title>Object Model</title>
<style type = "text/css">
   .bigText { font-size: 3em; font-weight: bold }
   . smallText { font-size: .75em }
</style>
<script type = "text/javascript">
function start()
{
varinputClass = prompt("Enter a className for the text " + "(bigText or smallText)", "" );
pText.className = inputClass;}
</script></head>
<body onload = "start()">
<p id = "pText">Welcome to our Web site!</p>
</body></html>
```



The above example contains two classes: .bidText and .smallText. The **class** attribute applies a style class to an element.The class name always follows a period operator (.). Similar properties can be grouped into one class.The property name is followed by a colon (:) and the value of that property. Multiple properties are separated by semicolons (;).

Dynamic Positioning

In dynamic positioning, the XHTML elements can be positioned with scripting. This is done by declaring an element's CSS position property to be either absolute or relative, and then moving the element by manipulating any of the top, left, right or bottom CSS properties.

```
<html><head><title>Dynamic  Positioning</title>
<script type = "text/javascript">
var speed = 5;var count = 10; var direction = 1; varfirstLine = "Text growing";
varfontStyle = [ "serif", "sans-serif", "monospace" ];
varfontStylecount = 0;
function start()
 { window.setInterval( "run()", 100 ); }
function run()
{ count += speed;
 if ( ( count % 200 ) == 0 )
{ speed *= -1;
 direction = !direction;
pText.style.color =35 ( speed< 0 ) ? "red" : "blue" ;
firstLine = ( speed < 0 ) ? "Text shrinking" : "Text growing";
pText.style.fontFamily = fontStyle[ ++fontStylecount % 3 ];
 }
pText.style.fontSize = count / 3;
pText.style.left = count;
pText.innerHTML = firstLine + "<br /> Font size: " + count + "px";
 }
</script></head>
<body onload = "start()">
<p id = "pText" style = "position: absolute; left: 0; font-family: serif; color: blue">
Welcome!</p></body></html>
```

- In the above example the position of the element is varied on the page by accessing its CSS left attribute, scripting to vary the color, fontFamily and fontSize attributes, and the element's innerHTML property is used to alter the content of the element.
- This function set interval takes two parameters—a function name, and how often to run that function (in this case, every 100 milliseconds). The setTimeout() takes the same parameters but instead waits the specified amount of time before calling the named function only once.

Frame collection

- Frames are seen as collection in DHTML.Usage of frames makes the web page more interactive.

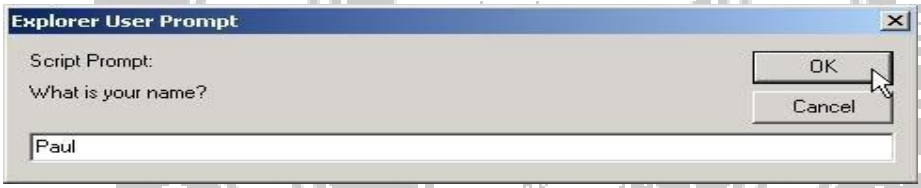
Cross frames

```
<html><head><title>Frames collection</title></head>
<frameset rows = "100, *">
<frame src = "frame.html" name = "upper" />
<frame src = "" name = "lower" /></frameset></html>
```

Frame.html

```
<html ><head><title>The frames collection</title>
<script type = "text/javascript">
```

```
function start()
{ var text = prompt( "What is your name?", "" );
parent.frames( "lower" ).document.write( "<h1>Hello, " + text + "</h1>" );
 }}</script></head><body onload = "start()">
<h1>Cross-frame scripting!</h1>
</body></html>
```



Filters and transitions

Applying filters to text and images causes changes that are persistent.**Transitions** are temporary phenomena. Applying a transition allows to transfer from one page to another with a pleasant visual effect.Filters and transitions do not add content to the pages. They just add

visual effects that work on some event.Filters and transitions are specified with the CSS **filter** property. Transitions give the same kind of graphics capabilities got from presentation software like Microsoft's PowerPoint.Filters are applied in the style attribute. The filter property's value is the name of the filter. Each filter has a property named enabled. If this property is set to true, the filter is applied. If it is set to false, the filter is not applied.

Filter	Description	Syntax
Flipv	Mirrors text vertically	< style = "filter: flipv ">Text</style>
Fliph	Mirrors text horizontally	< style = "filter: fliph ">Text</style>
Chroma	Applies transparency effects dynamically, without using a graphics editor to hard-code transparency into the image. This filter must be set andthen enabled explicitly.	chromaImg.filters("chroma").color = theColor; chromaImg.filters("chroma").enabled = true;
mask	Allows to create an image mask, in which the background of an element is a solid color and the foreground of an element is transparent to the image or color behind it.	< style =”filter: mask(color = CCFFFF)">
invert	Applies a negative image effect—dark areas become light, and light areas become dark.	< style = "filter: invert">
Gray	The gray filter applies a grayscale image effect, in which all color is stripped from the image and all that remains is brightness data.	< style = "filter:gray">
Xray	The xray filter applies an x-ray effect, which basically is an inversion of the grayscale effect.	< style = "filter:xray">
Shadow	This filter creates a shadowing effect that gives the text a three-dimensional appearance. Property direction of the shadowfilter determines in which direction the shadow effect is applied and Property color specifies the color of the shadow that is applied to	<style = “filter: shadow(direction = 0, color = red)>

	the text. The direction is given in angles.	
Alpha	The alpha filter also is used for transparency effects not achievable with the chroma filter. The style parameter takes value of 0 for uniform opacity, 1 for linear gradient, 2 for circular gradient and 3 for rectangular gradient. The opacity and finishopacity properties are both percentages that determineat what percent opacity the specified gradient starts and finishes, respectively. Additional attributes are startX, startY, finishX and finishY specifies at what x-y coordinates the gradient starts and finishes in that element.	<style = “filter: alpha(style = 2, opacity = 100,finishopacity = 0)">
Glow	The glow filter adds an aura of color around text. The color and strength can both be specified as parameters.	< style = “filter: glow(color = red, strength = 5)”>
Blur	The blur filter creates an illusion of motion by blurring text or images in a certain direction and can be applied in any of eight directions, and its strength can vary. The add property, when set to true, adds a copy of the original image over the blurred image, creating a more subtle blurring effect.	<style = "filter: blur(add = 0, direction = 0, strength = 0)">
Wave	The wave filter allows you to apply sine-wave distortions to text and images on your WebPages. The add property, like the blur filter, adds a copy of the text or image underneath the filtered effect. The freq and phase property determines the frequency and phase shift of the wave	<style = " filter: wave(add = 0, freq = 1, phase = 0,strength = 10)”>

	applied respectively. Strength is the amplitude of the sine wave that is applied.	
dropShadow	The dropShadowfilte drops shadow we applied to images.	<style=" filter: dropShadow(offx = 0, offy = 0, color = black) ">
Light	The light filter simulates the effect of a light source shining on the page.	< style = "filter: light">

Filters

```
<html><head><title>  filters</title></head><body>
<div style="position:absolute; top:125; left:2; filter:mask">
</div><imgsrc="cat.gif" width=400 height=200 style="filter:invert">
</body></html>
```

Transitions	Description	Syntax
blendTrans	This creates a smooth fade-in/fade-out effect. The duration determines how long the transition takes.	<style=": blendTrans(duration = 3)">
revealTrans	This allows the transition by using professional-style transitions, from box out to random dissolve. The transition paramerter is the index of the element as specified in the transition array. There are 24 tansitions.	<style=" filter: revealTrans(duration = 2, transition = 0)">

Transitions

```
<html><head><script language="text/javascript">
Function blendOut()
{f.filters("blendTrans").apply();
f.style visibility="hidden";
f.filters("blendTrans").play();
}</script></head>
<body><div id="f" onClick="blendOut()" style="filter:blendTrans(duration=5)">
This is a nice effect</div>
</body></html>
```

Data binding with tabular control

Before the advent of DHTML, the data manipulations were done on the server thus increasing the server load and the network load. With DHTML, these manipulations can be done directly on the client without involving the server and the network.**Data binding** is a process that allows an Internet user to manipulate Web page elements using a Web browser. It employs dynamic HTML and does not require complex scripting or programming.With data binding, data need no longer reside exclusively on the server. The data can be maintained on the client. The data storage is well distinguished from the XHTML markup on the page. In DHTML, larger amount of data will be sent to the client on the first request. Changes done to the data the client side do not propagate back to the server. To bind external data to XHTML elements, Internet Explorer employs software that iscapable of connecting the browser to live data sources. These are known as **Data Source Objects(DSOs).**

Tabular Data Control

The **Tabular Data Control (TDC)** is an ActiveX control that is added to a page with the object element. Data are stored in a separate file and will not be a part of the XHTML document. TDC is one way of accessing data from DSO.

- The object element will have the following properties :
  - Classid-specifies the TDC to be added to the page
  - Param tag-specifies the parameters for the object in the form of 'name-value' pairs.
- The following are the parameters taken by TDC:
  - Data URL: URL of data source.
  - UseHeader: If this value is true, then the first line of the data file has header field.
  - TextQualifier: Qualifiers are characters that are placed at both the ends of a filed.
  - FieldDelim: Characters that act as separaters between different data fields.
- **Record set** is the set of data from the data source. The following are the methods to access the record set:

Methods	Description
Move Next()	Moves the recordset to the next row in the data source.
Move First()	Moves to the first recordset in the file.
MoveLast()	Moves to the last recordset in the file.
Move Previous()	Moves to the previous recordset in the file.

Tabular Data Control

```
<html><head><title>Dynamic Recordset Viewing</title>
<object id = "Colors" classid = "CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value ="HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
<param name = "FieldDelim" value = "|" /></object>
<script type = "text/javascript">
varrecordSet = Colors.recordset;
function update()
{ h1Title.style.color = colorRGB.innerText; }
function move( whereTo )
{ switch ( whereTo )
{
case "first": recordSet.MoveFirst(); update(); break;
case "previous":recordSet.MovePrevious();
                if ( recordSet.BOF )
recordSet.MoveLast(); update(); break;
case "next": recordSet.MoveNext();
               if ( recordSet.EOF )
recordSet.MoveFirst(); update(); break;
case "last": recordSet.MoveLast();update(); break;
 } }</script>
<style type = "text/css">
input { background-color: khaki; color: green; font-weight: bold }
</style></head>
<body style = "background-color: darkkhaki">
<h1 style = "color: black" id = "h1Title"> XHTML Color Table</h1>
<span style = "position: absolute; left: 200; width: 270; border-style: groove; text-align: center;
background-color: cornsilk; padding: 10">
```

```
<strong>Color Name: </strong>
<span id = "colorName" style = "font-family: monospace"
datasrc = "#Colors" datafld = "ColorName">ABC</span><br />
<strong>Color RGB Value: </strong>
<span id = "colorRGB" style = "font-family: monospace"
datasrc ="#Colors" datafld ="ColorHexRGBValue">ABC
</span><br />
<input type = "button" value = "First"onclick = "move( 'first' );" />
<input type = "button" value = "Previous"onclick = "move( 'previous' );" />
<input type = "button" value = "Next" onclick = "move( 'next' );" />
<input type = "button" value = "Last" onclick = "move( 'last' );" />
</span></body></html>
```

Create a data sorce named HTMLStandardColors.txt that contains colors and its color values.

Binding to an image

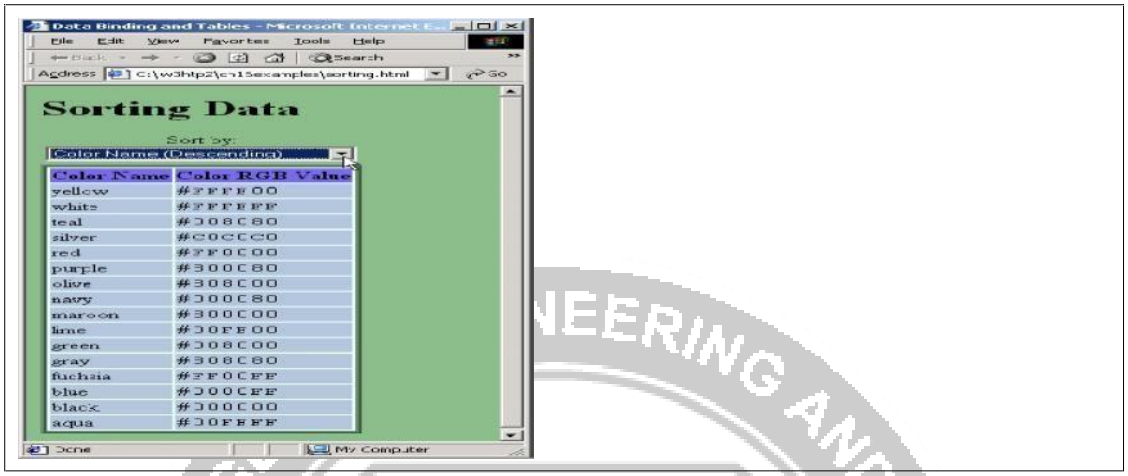
Many different types of XHTML elements for instance, image can be bound to data sources.Images are binded with data set by modifying the src attribute of the image.

<imgdatasrc=”cat.jpeg” datafld=”image”>

The previous example, changes the color based on the recordset. Now by changing the datasrc and datafld attribute as above will make the recordset to traverse through various images (create an image data source). Omit the update();

Binding to a table and sorting the table data: Tables could also be binded to the data source and sorted.

```
<html ><head><title>Data Binding and Tables</title>
<object id="Colors" classid="CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value ="HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
<param name = "FieldDelim" value = "|" />
</object></head>
<body style = "background-color: darkseagreen">
<h1>Sorting Data</h1>
<table datasrc="#Colors" style="border-style: ridge; border-color: darkseagreen;
background-color: lightcyan">
<caption> Sort by: <select onchange = "Colors.Sort = this.value; Colors.Reset();">
<option value = "ColorName">Color Name (Ascending) </option>
<option value = "-ColorName">Color Name (Descending) </option>
<option value = "ColorHexRGBValue">Color RGB Value (Ascending)</option>
<option value = "-ColorHexRGBValue">Color RGB Value (Descending)</option>
</select></caption>
<thead><tr style = "background-color: mediumslateblue">
<th>Color Name</th><th>Color RGB Value</th></tr></thead>
<tbody><tr style = "background-color: lightsteelblue">
<td><span datafld = "ColorName"></span></td>
<td><span datafld = "ColorHexRGBValue" style = "font-family:
monospace"></span></td>
</tr></tbody></table></body></html>
```



Specify the column by which to sort in the Sort property of the TDC. This example sets property Sort to the value of the selected option tag (this.value) when the onchange event is fired.By default, a column is sorted in ascending order. To sort in descending order, the column name is preceded with a minus sign (-).

Data binding elements

The following elements allow data binding:

Element	Bindable attribute
A	Href
Frame	Href
Iframe	Href
Img	Src
Div	Contained text
Input type=”button”	value
Input type=”chaeckbox”	Checked
Input type=”hidden”	value
Input type=”password”	Value
Input type=”radio”	Checked
Input type=”text”	Value
Marquee	Contained text

Param	Value
Select	select option
Span	Contained text
Table	Contained elememts
Textarea	Contained text

Structures graphics and ActiveX controls

The Structured Graphics Control is an ActiveX control that can be add to the page with an object element. It is easily accessible through scripting.

The Structured Graphics Control is a web interface that is used for visual presentations.The Structured Graphics control facilitates the creation of simple shapes by using functions that can be called via scripting or through param tags inside object elements.

The name attribute of the param tag method determines the order in which the function specified in the value attribute is called. The distortion of shapes like translation, rotation can also be done. To provide interaction with the user, the Structured Graphics Control can process the Dynamic HTML mouse events onmouseup, onmousedown, onmousemove, onmouseover, onmouseout, onclick and ondblclick. By default, the Structured Graphics Control does not capture mouse events, because doing so takes a small amount of processing power.The Structure Graphics Control allows you to keep a set of method calls in a separate source file and to invoke those methods by calling the SourceURL function.The following are the functions available for Structured Graphics Control:

Function	Description
SetLineColor(Rvalue, Gvalue, Bvalue)	sets the color of lines and borders of shapes that are drawn. It takes an RGB triplet in decimal notation as its three parameters.
SetLineStyle(line style, line width)	Draws a line. A value of 1 for line style creates a solid line (the default). A value of 0 does not draw any lines or borders, and a value of 2 creates a dashed line.
SetFillColor(color)	Sets the foreground color with which to fill shapes.
SetFillStyle(color style)	Sets the style in which a shape is filled with color; a value of 1 fills shapes with the solid color declared with the SetFillColor method.
Oval(x, y, height, width, direction)	Places the oval in specifies x-y location. The last parameter specifies the clockwise rotation of the oval relative to the x-axis, expressed in degrees.

Arc(x,y,height,width,starting angle, size, rotation).	Draws an arc with the given parameters.
Pie(x,y,height,width,starting angle, size, rotation)	It fills in the arc with the foreground color, thus creating a pie shape.
Polygon(no of vertices, (x,y) coordinates of the sides of the polygon)	Constructs a polygon. If the no of vertices of the polygon is 3, then 3 pairs of (x,y) co-ordinates must be specified.
Rect(x, y, height, weight, rotation)	Constructs a rectangle.
RoundRect(x, y, height, weight, rotation, height of rounded arc, width of rounded arc)	Constructs a rounded rectangle.
SetFont()	Sets the font style to use when placing text with the Text method.
PolyLine(no of points in the line, x, y of other vertex)	Draws a line with multiple segments
SetTextureFill(x, y, location of texture, value)	Fills a shape with a texture. A last parameter of 0 specifies that the texture should be stretched to fit inside the shape. A last parameterof 1 would instead tile the texture as many times as necessary inside the shape.
Translate(x, y, z)	Moves a shape in coordinate space without deforming it. Its three parametersdetermine the relative distance to move along the x-, y- and z-axes, respectively.
Rotate(x, y, z)	Rotates shapes in three-dimensional space. The three parameters of the Rotate function specify rotation in the x-, y- and z-coordinate planes, respectively.
MouseEventsEnabled()	Turn event capturing on

Path, Sequencer and Sprite ActiveX Controls

The DirectAnimation Path Control allows to control the positions of elements on the page.The Path Control, the Sequencer Control and the Sprite Control allows a Web page designer to add certain multimedia effects to Web pages.This mechanism is more advanced than dynamic CSS positioning, because it allows to define paths that the targeted elements follow. This capacity to define paths gives the ability to create professional presentations, especially when integrated with other Dynamic HTML features such as filters and transitions.

Setting AutoStart attribute of the object to a nonzero value starts the element along a path as soon as the page loads. Setting a zero value prevents it from starting automatically, in which case a script would have to call the Play method to start the path.

- The **Path Control** also allows setting paths for multiple objects present on your page. To set paths for multiple objects, add a separate object tag for each object.
- The z-index of elements that overlap is determined by their order of declaration in the XHTML source (elements declared later in the XHTML file are displayed above elements declared earlier).
- A useful feature of the Path Control is the ability to execute certain actions at any point along an object's path. This capability is implemented with the AddTimeMarker method, which creates a time marker that can be handled with simple JavaScript event handling.
- The **Sequencer Control** provides a simpler interface for calling functions or performing actions at time intervals.The **oninit** event fires when the Sequencer Control has loaded.
- The Item object of the Sequencer Control creates a grouping of events using a common name.The **Sprite Control** allows the displaying animated images composed of individual frames.
- The object tag inserts the Sprite Control. The height and width CSS properties are needed to display the image correctly; they should be equal to the size of one frame in the file. Setting attribute
- Repeat to a nonzero value loops the animation indefinitely. **NumFrames** specifies how many frames are present in the animation source image.
- **Attributes NumFramesAcross** and **Num-FramesDown** specify how many rows and columns of frames there are in the animation file, respectively.**SourceURL** gives a path to the file containing the frames of the animation.
- The animated GIFs are most popular animation formats and are composed frames in the GIF image format. GIF images must be inserted into animated GIF files by using graphics applications such as Adobe PhotoShop elements.

Method	Description
Repeat()	Determines how many times the path will be traversed; setting the value to -1 specifies that the path should loop continuously
Duration()	Specifies the amount of time that it takes to traverse the path, in seconds.
Bounce()	When set to 1, reverses the element's direction on the path when it reaches the end. Setting the value to 0 returns the element to the beginning of the path when the path has been traversed.

PolyLine()	Creates a path with multiple line segments.
Target()	Specifies the id of the element that is targeted by the Path Control. Setting the CSS attribute position to absolute allows the Path Control to move an element around the screen. Otherwise, the element would be static, locked in the position determined by the browser when the page loads.
AddTimeMarker()	The first parameter determines the point at which our time marker is placed along the path, specified in seconds; when this point is reached, the onmarker event is fired. The second parameter gives an identifyingname to the event, which is later passed on to the event handler for the onmarker event. The last parameter specifies whether to fire the onmarker event every time the object's path loops pastthe time marker (value=0) or to fire the event just the first time that the time marker is passed (value= 1).
At(waiting time, action)	This method takes two parameters: How many seconds to wait, and what action to perform when that period of time has expired.
Play()	Starts the targeted element along the path.
Sprite Control()	Controls the rate at which frames are displayed
MouseEventsEnabled()	Enables or disables mouse events.
Stop()	Stops the animation in place

Advantages of Activex Controls: Platform Control, Active X Scripting and Easy To Use and Easy to Find

Disadvantages of Activex Controls: Requires User to Download Something, System Vulnerabilities, Built-in Malware and Spyware and Only Compatible with Microsoft Programs