



UNIVERSITY OF DIYALA

# *Web Design and Programming*

## *Lecture 4:*

### *Images , Tables And Objects*

*Instructor: Ali A. Al-Ani*



Department of Computer Science  
College of Science



UNIVERSITY OF DIYALA

## *Objectives*

1. Learn how to add images into our documents using the *<img> element*.
2. Learn how to make an image a link.
3. Learn how to divide an image up into sections so that different parts of the image link to different pages, this is known as an image map.
4. Learn how to use the *<object>* element to insert all manner of objects into pages, from MP3s and Flash movies to Active X controls and even images.
5. Learn how to add table into our documents using the *<table>* element.

Department of Computer Science  
College of Science



## *Adding Images to Your Site*

- Images and graphics can really bring our site to life. but we should be careful when using images on the Web because if we don't prepare images correctly, they can really slow down the speed it takes for a page to load, and slow sites frustrate users.
- So choosing the right format for our images and saving them correctly will help make our site faster and result in happier visitors.
- Images are usually added to a site using the `<img>` element. It must carry the **src** attribute indicating the source of the image For example:

``



## *Adding Images Using the <img> Element*

- The `<img>` element can carry all of the universal and UI event attributes:  
*Src, alt, align, border, height, width*  
*hspace, vspace, ismap, usemap, longdesc, name*
- The **src** attribute is required to specify the URL of the image to load `src="url"`. The URL can be an absolute URL or a relative, just like the URLs when linking between pages.
- Images for our site should always reside on our server. It is not good practice to link to images on other sites because if the owner of the other site decides to move that image our users will no longer be able to see the image on our site.



## *The alt Attribute*

- The **alt** attribute is required to specify a text alternative for the image in case the user cannot see the image. It is important that the value of this attribute really describes the image, and two common reasons why images are not visible to users are:
  1. *Because the browser did not download the file correctly; the file cannot be found.*
  2. *Because the user has visual impairment that prevents him or her from seeing the image.*



## *The height and width Attributes*

- The height and width attributes specify the height and width of the image: **height="120" width="180"**. The values for these attributes are always shown in pixels.
- Specifying the size of the image can help browsers lay out pages faster and more smoothly because they can allocate the correct amount of space to the image and continue to render the rest of the page before the image has finished loading.
- If we reduce the size of the image using the height and width attributes, the user will still have to download the full-sized image, which takes longer than a special small version and uses up more bandwidth.



## *The border, hspace and vspace Attributes*

- The ***border*** attribute specifies the width of the border around the image in pixels: ***border="2"***.
- The ***hspace*** and ***vspace*** attributes can be used to control the amount of whitespace around an image.

***hspace="10", vspace="14"***

- The ***hspace*** and ***vspace*** attributes are particularly helpful because text can flow around an image and, unless there is a gap between the text and the image, the text becomes hard to read and doesn't look as professional.



## *Using Images as Links*

- It's easy to turn an image into a link; rather than putting text between the opening ***<a>*** element, as we saw in the last chapter, we can place an image inside these tags. Images are often used to create graphical buttons or links to other pages. For example

***<a href="../index.html">  </a>***

- Note also that the image in this example is not a very good example of a link, as it does not tell us where the link is going to take us. If we use images as links, you should make it clear what will happen if the user clicks the link.



## *Image Maps*

- Image maps allow you to specify several links that correspond to different areas of one single image, so that when users click different parts of the image they get taken to different pages. There are two types of image maps:
  1. *Server-side image maps*
  2. *Client-side image maps*
- Image maps are particularly helpful when the image needs to be divided up in irregular shapes, such as maps. However, if the image can be divided up in a grid, you might be better off chopping up an image manually and putting it together in a table.



## *Image Maps*





## Server-Side Image Maps

- With server-side images, the `<img>` element (inside an `<a>` element) carries a special *ismap* attribute, which tells the browser to send the *server x, y coordinates* representing where the user's mouse was when he or she clicked the image map.
- Then a script on the server is used to determine which page the user should be sent to based on the coordinates fed to it.
- For example, look at the following link, where the `<img>` element carries the *ismap* attribute with a value of *ismap*.

```
<a href="map.aspx">  </a>
```



## Server-Side Image Maps

- Now, if the user clicks the image 50 pixels to the right of the top-left corner of the image and 75 pixels down from the that same corner, the browser will send this information with the URL like so:

*<http://www.example.org/location/map.aspx?50,75>*

- The thing about a server-side image map is that there needs to be a *script, map file, or application on the server* that can process the coordinates and know which *page* the user should then be sent to.
- The implementation of image maps will vary depending on what kind of server we are running on.



## *Client-Side Image Maps*

- Because server-side image maps rely on server technology, an alternative that worked on browsers was introduced and client-side image maps.
- Client-side image maps use code within the XHTML page to indicate which parts of the image should link to which pages. Because the code that divides up the sections of the image is on the browser, it is possible for the browser to offer extra information to users, either by showing them a URL in the status bar or as a tooltip when the mouse is hovered over the image.
- There are two methods of creating a client-side image map: using the *`<map>` and `<area>` elements inside an `<img>` element*



## *Client-Side Image Maps Using `<map>` and `<area>`*

- The image that is going to form the map is inserted into the page using the `<img />` element as normal, except it carries an extra attribute called ***usemap***.
- The value of the ***usemap*** attribute is the value of the name attribute on the ***`<map>` element***.
- The ***`<map>` element*** actually creates the map for the image and usually follows directly after the ***`<img />` element***. It acts as a container for the ***`<area>` elements*** that actually define the clickable hotspots. The ***`<map>` element*** carries only one attribute, the ***name attribute***, which is the name that identifies the map. This is how the ***`<img />` element*** knows which ***`<map>` element*** to use.



## *Client-Side Image Maps Using <map> and <area>*

- The *<area>* *element* specifies the shape and the coordinates that define the boundaries of each clickable hotspot. For example

```

```

```
<map name="gallery">
```

```
<area shape="circle" coords="154,150,59" href="foyer.html" target="_self">
```

```
<area shape="poly" coords="272,79,351,79,351,76" href="sgarden.html" target="_self" />
```

```
<area shape="rect" coords="325,224,488,286" href="workshop.html" target="_self" />
```

```
</map>
```



## *Client-Side Image Maps*

- As we can see, the value of the *usemap* attribute on the *<img />* element is *#gallery*, and this is used on the *<map>* element.
- Then the *<area>* elements define the sections of the image that are clickable.
- If we have two areas that overlap each other, the first one in the code will take precedence.
- The attributes that the *<area>* element can carry may look familiar from the *<a>* element.

*Accesskey alt shape coords href nohref target tabindex taborder notab*





## *The shape Attribute*

- The value of the **shape** attribute affects how the browser will use the coordinates specified in the **coords** attribute. If we do not specify a shape attribute, IE usually assumes the area is a rectangle. The following table shows the possible values of the shape attribute.

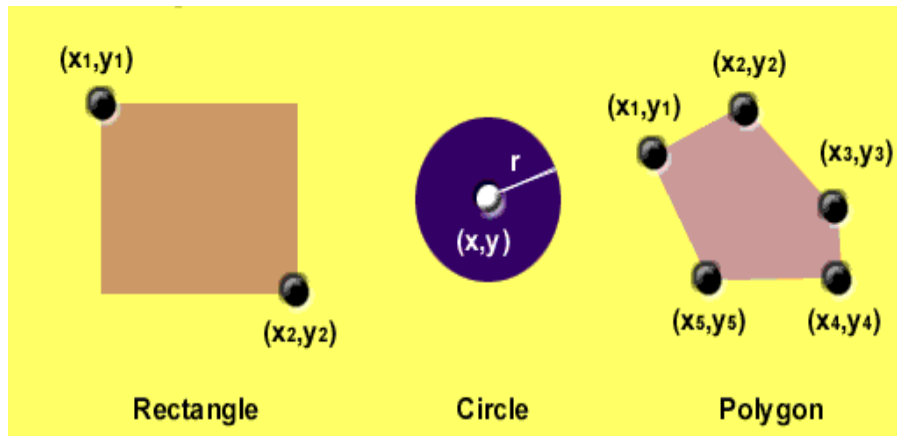
<i>Value</i>	<i>Shape Created</i>
<i>default</i>	<i>The whole of the image not defined in an area (should be specified last)</i>
<i>rectangle or rect</i>	<i>Rectangle</i>
<i>polygon or poly</i>	<i>Polygon</i>
<i>circle or circ</i>	<i>Circle</i>



## *The coords Attribute*

- The **coords** attribute specifies the area that is the clickable hotspot. The number of coordinates you specify depends on the shape you are creating (and have specified in the shape attribute).
  - A rectangle contains four coordinates.* The first two coordinates represent the top left of the rectangle, and the second two the bottom right.
  - A circle contains three coordinates;* the first two are the center of the circle, while the third is the radius in pixels.
  - A polygon contains two coordinates for each point of the polygon.* So a triangle would contain six coordinates, a pentagon would contain ten, and so on.

## *The coords Attribute*



## *The href, nohref and alt Attributes*

- The **href** attribute works just like the **href** attribute for an `<a>` element; its value is the URL of the page you want to load when the user clicks that part of the image.
- If you do not have an **href** attribute, you must use a **nohref** attribute indicating that the area will not take you anywhere, it takes a value of **nohref**.
- **The alt Attribute:** The **alt** attribute specifies a text alternative for that section of the image.



## Tables

- Tables are commonly used to display all manner of data, such as timetables, financial reports, and sports results. So when we want to display information in rows and columns, we need to use the markup to create a table.
- The names of elements in XHTML refer to the type of markup they contain. So to create a table in XHTML we use the *<table> element*.
- Inside the *<table> element*, the table is written out row by row. A row is contained inside a *<tr> element*, which stands for table row. And each cell is then written inside the row element using a *<td> element*, which stands for table data.



## Tables

- The following is an example of a very basic table :

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



## ***Basic Table Elements and Attributes***

1. ***The <table> Element Creates a Table:*** The <table> element is the containing element for all tables. It can carry the following attributes:

1. ***All of the universal attributes.***

2. ***Basic event attributes for scripting.***

- The <table> element can also carry the following attributes.

*align bgcolor border cellpadding cellspacing  
dir frame rules summary width*



## ***Basic Table Elements and Attributes***

1. ***The align Attribute:*** the *align* attribute is still frequently used with tables. When used with the <table> element, it indicates whether the table should be aligned to the left (the default), right, or center of the page.
2. ***The bgcolor Attribute:*** the *bgcolor* attribute sets the background color for the table. The value of this attribute should be either a six-digit code known as a hex code or a color name. For example *bgcolor="#rrggbb"*
3. ***The border Attribute:*** If we use the border attribute, a border will be created around both the table and each individual cell.



## Basic Table Elements and Attributes

4. **The cellpadding Attribute:** The cellpadding attribute is used to create a gap between the edges of a cell and its contents. As we can imagine, if two cells both contain writing, and there is no gap between the edge of the cells and the writing, the contents can become hard to read. *cellpadding="5" or cellpadding="2%"*
5. **The cellspacing Attribute:** The cellspacing attribute is used to create a space between the borders of each cell. The value for this attribute can be either the amount of space you want to create between the cells in pixels or a percentage value (as a percentage of the width of the table). *cellspacing="6" or cellspacing="2%"*



## Basic Table Elements and Attributes

6. **The dir Attribute:** The dir attribute is supposed to indicate the direction of text that is used in the table. Possible values are *ltr* for left to right text and *rtl* for right to left (for languages such as Arabic): *dir="rtl"*
2. **The <tr> Element Contains Table Rows:** The <tr> element is used to contain each row in a table. It can carry five attributes. *Align, bgcolor, char, charoff, valign*
3. **The <td> and <th> Elements Represent Table Cells:** Every cell in a table will be represented by either a <td> element for cells containing table data or a <th> element for cells containing table headings.



## Basic Table Elements and Attributes

- By default the contents of a `<th>` element are usually displayed in a bold font, horizontally aligned in the center of the cell.
- The `<td>` and `<th>` elements can both carry the same set of attributes, each of which applies just to that cell.
- In addition to the universal attributes and the basic event attributes, the `<td>` and `<th>` elements can also carry the following attributes:

*abbr align axis bgcolor char charoff colspan*

*headers height nowrap rowspan scope valign width* The *abbr* Attribute



## The `<table>` Element's Other Attributes

- Now that we've seen the basics behind creating tables, but there is some more advanced issues, such as the following:
  1. **Splitting a table into three sections: a head, body, and foot** Captioning tables
  2. Using the *rowspan* and *colspan* attributes to make cells stretch over more than one row or column
  3. Grouping columns using the `<colgroup>` element Sharing attributes between unrelated columns using the `<col>` element
- **Homework :** *discuss the mentioned issue with example for each.*

**Note: this assignment will be included in the exams.**



## *Adding Other Objects with the <object> Element*

- The **<object>** element introduced in HTML 4 with the intention that it be used to embed *all media types* into documents, not just graphics but also MP3 files, Flash movies, QuickTime movies, JavaScript objects and so on. Also, the **<object>** element is used to include some other kind of software that is used to play or load these files. For example:

1. *Flash movies are played with the Flash Player;*
2. *Windows Media Files require Windows Media Player;*
3. *MP3s can be played in various players including Flash Player, Windows Media Player, and QuickTime Player.*



## *Adding Other Objects with the <object> Element*

- So when it comes to embedding audio, video, or Java/JavaScript programs on our web page, we not only need to have the file, but you also have to choose an application to embed into our page that will play/run the file. To embed an object into a page, we need to specify:
  1. The location of the code used to display or play the object.
  2. The actual data to be rendered (for example a movie, an audio , a program)
- The **<object>** element can carry all of the universal attributes, the UI event attributes, and the following attributes:

*Archive, border, class, id, codebase, codetype, data, declare,  
height, width, hspace, vspace, name, standby, tabindex, usemap*



### *The <object> Element's Attributes*

- The objects are added using the **<object>** element, while additional values are provided in the **<param>** element, which can be a child of the **<object>** element.
- The **<param>** element is used to pass parameters to an object. The kinds of parameters an object requires depend upon what the object does; for example, if an object has to load an MP3 player into the page, you will probably need to specify where the MP3 file can be found.
- As well as the universal attributes and basic events, the **<param>** element can carry the following attributes:

*name type value valuetype*



# *The End*