Web Design
Course

LECTURE SIVEN
Introduction to HTML and XHTML

Computer Science
Collage of Education
AL_Mustansyria University
Forth Year

2018-2019

6.2- The rowspan and colspan Attributes

In many cases, tables have multiple levels of row or column labels in which one label covers two or more secondary labels. For example, consider the display of a partial table shown in Figure 18. In this table, the upper-level label Fruit Juice Drinks spans the three lower-level label cells. Multiple-level labels can be specified with the rowspan and colspan attributes.

Figure 18 Two levels of column labels

The colspan attribute specification in a table header or table data tag tells the browser to make the cell as wide as the specified number of rows below it in the table. For the previous example, the following markup could be used:

```
<tr>
  <th colspan = "3"> Fruit Juice Drinks </th>
</tr>
<tr>
  <th> Apple </th>
  <th> Orange </th>
  <th> Screwdriver </th>
</tr>
```

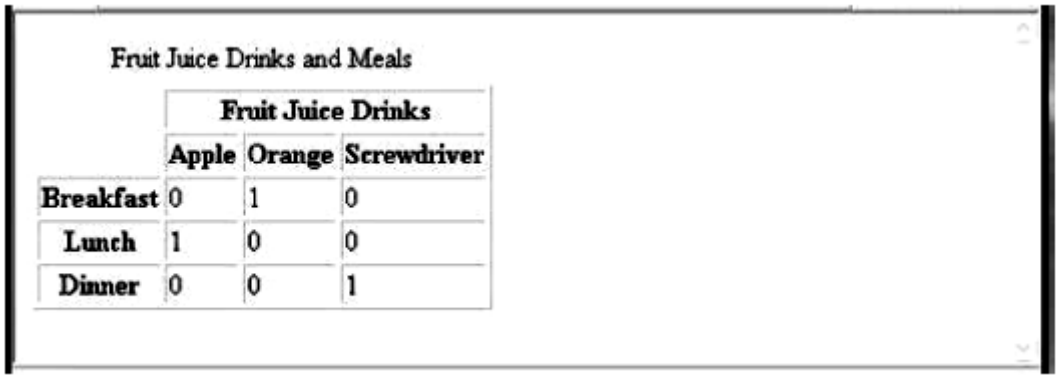
If there are fewer cells in the rows above or below the spanning cell than the colspan attribute specifies, the browser stretches the spanning cell over the number of cells that populate the column in the table. The rowspan attribute of the table heading and table data tags does for rows what colspan does for columns.

A table that has two levels of column labels and also has row labels must have an empty upper-left corner cell that spans both the multiple rows of column labels and the multiple columns. Such a cell is specified by including both rowspan and colspan attributes. Consider the following table specification, which is a minor modification of the previous table:

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

<!-- cell_span.html
     An example to illustrate rowspan and colspan
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Rowspan and colspan </title>
  </head>
  <body>
    <table border = "border">
      <caption> Fruit Juice Drinks and Meals </caption>
      <tr>
        <td rowspan = "2"> </td>
        <th colspan = "3"> Fruit Juice Drinks </th>
      </tr>
      <tr>
        <th> Apple </th>
        <th> Orange </th>
        <th> Screwdriver </th>
      </tr>
      <tr>
        <th> Breakfast </th>
        <td> 0 </td>
        <td> 1 </td>
        <td> 0 </td>
      </tr>
      <tr>
        <th> Lunch </th>
        <td> 1 </td>
        <td> 0 </td>
        <td> 0 </td>
      </tr>
      <tr>
        <th> Dinner </th>
        <td> 0 </td>
        <td> 0 </td>
        <td> 1 </td>
      </tr>
    </table>
  </body>
</html>
```

Figure 19 shows a browser display of cell_span.html.



Fruit Juice Drinks and Meals

	Fruit Juice Drinks		
	Apple	**Orange**	**Screwdriver**
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

Figure 19 Display of cell_span.html: multiple-labeled columns and labeled rows

6.3- The align and valign Attributes

The placement of the content within a table cell can be specified with the align and valign attributes in the <tr>, <th>, and <td> tags. The align attribute has the possible values left, right, and center, with the obvious meanings for horizontal placement of the content within a cell. The default alignment for the cells is center; for td cells, it is left. If align is specified in a <tr> tag, it applies to all of the cells
in the row. If it is included in a <th> or <td> tag, it applies only to that cell.
The valign attribute of the <th> and <td> tags has the possible values top and bottom. The default vertical alignment for both headings and data is center. Because valign applies only to a single cell, there is never any point in specifying center.The following example illustrates the align and valign attributes:

```
<!-- cell_align.html
     An example to illustrate align and valign
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Alignment in cells </title>
  </head>
  <body>
    <table border = "border">
      <caption> The align and valign attributes </caption>
      <tr align = "center">
        <th> </th>
        <th> Column Label </th>
        <th> Another One </th>
        <th> Still Another One </th>
      </tr>
      <tr>
        <th> align </th>
        <td align = "left"> Left </td>
        <td align = "center"> Center </td>
        <td align = "right"> Right </td>
      </tr>
      <tr>
        <th> <br /> valign <br /> <br /> </th>
        <td> Default </td>
        <td valign = "top"> Top </td>
        <td valign = "bottom"> Bottom </td>
      </tr>
    </table>
  </body>
</html>
```

Figure **20 shows a browser display of cell_align.html.**

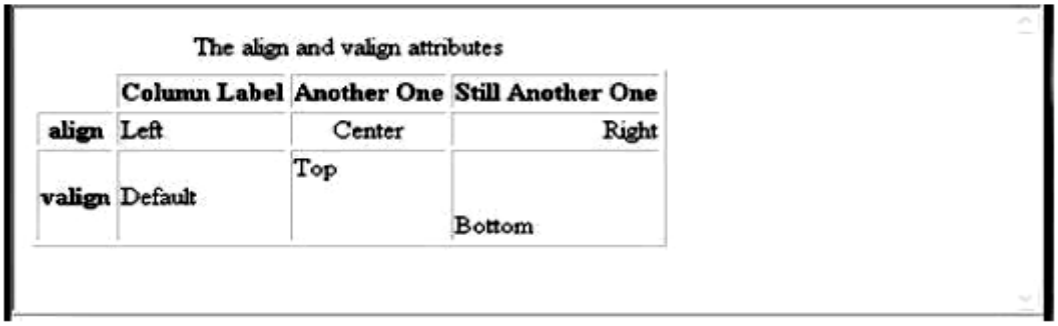


Figure 20 Display of cell_align.html: the align and valign attributes

6.4 The cellpadding and cellspacing Attributes

The table tag has two attributes that can respectively be used to specify the spacing between the content of a table cell and the cell's edge and the spacing between adjacent cells. The cellpadding attribute is used to specify the spacing between the content of a cell and the inner walls of the cell—often, to prevent text in a cell from being too close to the edge of the cell. The cellspacing attribute is used to specify the distance between cells in a table. The following document, space_pad.html, illustrates the cellpadding and cellspacing attributes:

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

<!-- space_pad.html
     An example that illustrates the cellspacing and
     cellpadding table attributes
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Cell spacing and cell padding </title>
  </head>
  <body>
    <b>Table 1 (space = 10, pad = 30) </b><br /><br />
    <table border = "5"  cellspacing = "10"  cellpadding = "30">
      <tr>
        <td> Small spacing, </td>
        <td> large padding </td>
      </tr>
    </table>
    <br /><br /><br /><br />
    <b>Table 2 (space = 30, pad = 10) </b><br /><br />
    <table border = "5"  cellspacing = "30"  cellpadding = "10">
      <tr>
        <td> Large spacing, </td>
        <td> small padding </td>
      </tr>
    </table>
  </body>
</html>
```

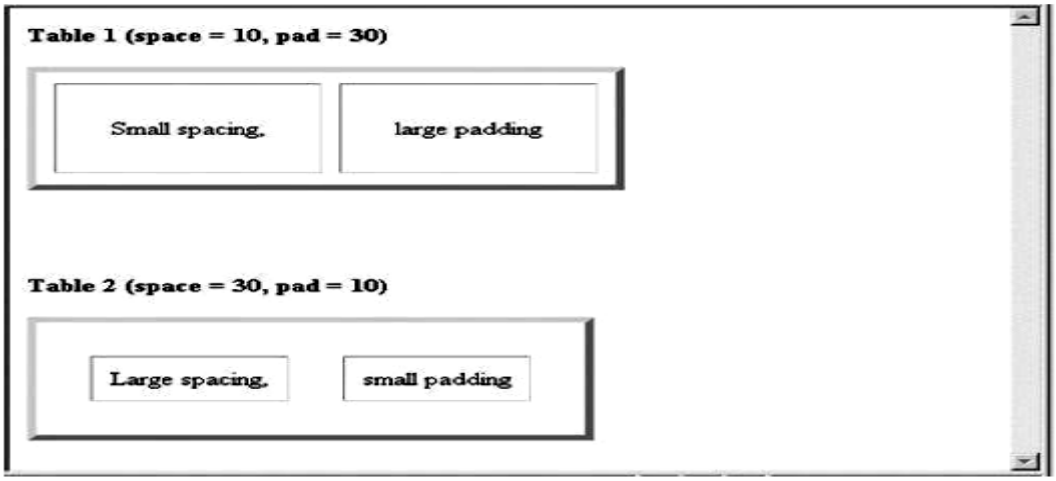


Figure 21 Display of space_pad.html

6.5 Table Sections

Tables naturally occur in two and sometimes three parts: header, body, and footer. (Not all tables have a natural footer.) These three parts can be respectively denoted in XHTML with the thead, tbody, and tfoot elements. The header includes the column labels, regardless of the number of levels in those labels. The body includes the data of the table, including the row labels. The footer, when it appears, sometimes has the column labels repeated after the body. In some tables, the footer contains totals for the columns of data above. A table can have multiple body sections, in which case the browser may delimit them with horizontal lines that are thicker than the rule lines within a body section.

7- Forms

The most common way for a user to communicate information from a Web browser to the server is through a form. Modeled on the paper forms that people frequently are required to fill out, forms can be described in XHTML and displayed by the browser. XHTML provides tags to generate the commonly used objects on a screen form. These objects are called controls or widgets. There are controls for single-line and multiple-line text collection, checkboxes, radio buttons, and menus, among others. All control tags are inline tags. Most controls are used to gather information from the user in the form of either text or button selections. Each control can have a value, usually given through user input. Together, the values of all of the controls (that have values) in a form are called the form data.

7.1 The <form> Tag

All of the controls of a form appear in the content of a <form> tag. A block tag, <form>, can have several different attributes, only one of which, action, is required. The action attribute specifies the URL of the application on the Web server that is to be called when the user clicks the Submit button. In this note, our examples of form elements will not have corresponding application programs, so the value of their action attributes will be the empty string ("").

The method attribute of <form> specifies one of the two techniques, get or post, used to pass the form data to the server. The default is get, so if no method attribute is given in the <form> tag, get will be used. The alternative technique is post. In both techniques, the form data is coded into a text string when the user clicks the Submit button. When the get method is used, the browser attaches the query string to the URL of the HTTP request, so the form data is transmitted to the server together with the URL. The browser inserts a question mark at the end of the actual URL just before the first character of the query string so that the server can easily find the beginning of the query string. The get method can also be used to pass parameters to the server when forms are not involved. (This cannot be done with post.)

One disadvantage of the get method is that some servers place a limit on the length of the URL string and truncate any characters past the limit.
So, if the form has more than a few controls, get is not a good choice. When the post method is used, the query string is passed by some other method to the form processing program. There is no length limitation for the query string with the post method, so, obviously, it is the better choice when there are more than a few controls in the form. There are also some security concerns with get that are not a problem with post.

7.2 The <input> Tag

Many of the commonly used controls are specified with the inline tag <input>, including those for text, passwords, checkboxes, radio buttons, and the action buttons Reset, Submit, and plain.
The one attribute of <input> that is required for all of the controls discussed in this section is type, which specifies the particular kind of control. The control's kind is its type name, such as checkbox. All of the previously listed controls except Reset and Submit also require a name attribute, which becomes the name of the value of the control within the form data. The controls for checkboxes and radio buttons require a value attribute, which initializes the value of the control. The values of these controls are placed in the form data that is sent to the server when the Submit button is clicked.
A text control, which we usually refer to as a text box, creates a horizontal box into which the user can type text. Text boxes are often used to gather information from the user, such as the user's name and address. The default size of a text box is often 20 characters. Because the default size can vary among browsers, it is a good idea to include a size on each text box. This is done with the size attribute of <input>. If the user types more characters than will fit in the box, the box is scrolled. If you do not want the box to be scrolled, you can include the maxlength attribute to specify the maximum number of characters that the browser will accept in the box. Any additional characters are ignored. As an example of a text box, consider the following:

```
<form action = "">
  <p>
    <input type = "text"  name = "Name"  size = "25" />
  </p>
</form>
```

Suppose the user typed the following line:

Alfred Paul von Frickenburger

The text box would collect the whole string, but the string would be scrolled to the right, leaving the following shown in the box:

ed Paul von Frickenburger

The left end of the line would be part of the value of Name, even though it does not appear in the box. The ends of the line can be viewed in the box by moving the cursor off the ends of the box. Notice that controls cannot appear directly in the form content—they must be placed in some block container, such as a paragraph. This is because <input> is an inline tag. Now consider a similar text box that includes a maxlength attribute:

```
<form action = "">
  <p>
    <input type = "text"  name = "Name"  size = "25"
           maxlength = "25" />
  </p>
</form>
```

If the user typed the same name as in the previous example, the resulting value of the Name text box would be as follows:

Alfred Paul von Frickenbu

No matter what was typed after the u in that person's last name, the value of Name would be as shown. If the contents of a text box should not be displayed when they are entered by the user, a password control can be used as follows:

```
<input type = "password"  name = "myPassword"
       size = "10" maxlength = "10" />
```

In this case, regardless of what characters are typed into the password control, only bullets or asterisks are displayed by the browser. There are no restrictions on the characters that can be typed into a text box. So, the string "?!34,:" could be entered into a text box meant for names. Therefore, the entered contents of text boxes nearly always must be validated, either on the browser or on the server to which the form data is passed for processing, or on both. Text boxes, as well as most other control elements, should be labeled. Labeling could be done simply by inserting text into the appropriate places in the form:

Phone: <input type = "text" name = "phone" />

This markup effectively labels the text box, but there are several ways the labeling could be better. For one thing, there is no connection between the label and the control. Therefore, they could become separated in maintenance changes to the document. A control and its label can be connected by putting the control and its label in the content of a label element, as in the following element: <label> Phone:

<input type = "text" name = "phone" />
</label>

Now the text box and its label are encapsulated together. There are several other benefits of this approach to labeling controls. First, browsers often render the text content of a label element differently to make it stand out. Second, if the text content of a label element is selected, the cursor is implicitly moved to the control in the content of the label. This feature is an aid to new Web users. Third, the text content of a label element can be rendered by a speech synthesizer on the client machine when the content of the label element is selected. This feature can be a great aid to a user with a visual disability. Checkbox and radio controls are used to collect multiple-choice input from the user. A checkbox control is a single button that is either on or off (checked or not). If a checkbox button is on, the value associated with the name of the button is the string assigned to its value attribute. A checkbox button does not contribute to the form data if it is off. Every checkbox button requires a name attribute and a value attribute in its <input> tag. For form processing on the server, the name identifies the button and the value is its value (if the button is checked). The attribute checked, which is assigned the value checked, specifies that the checkbox button is initially on. In many cases, checkboxes appear in lists, with every one in the list having the same name. Checkbox elements should appear in label elements, for the same reasons that text boxes should. The following example illustrates a checkbox: