Creating Accessible Data and Layout Tables

For more information please view the site, direct information has been copied for coding and other questions received from the Section 508 Accessibility Testing and Training Center Office. We do not support one company over another; this is a source of extra information on creating accessible tables.

1. Creating Accessible Tables - http://www.webaim.org/techniques/tables/data.php

Designate Row and Column Headers Using the <th> Tag

The very first step toward creating an accessible data table is to designate row and/or column headers. This is easy enough to do. Most authoring tools provide a method of changing data cells into header cells. In the markup, the <td> tag is used for table data cells and the <th> tag is used for table header cells. Going back to our original data table example, the column headers for this table are *Name*, *Age*, and *Birthday*. The row headers are *Jackie* and *Beth*.

Shelly's Daughters		
Name	**Age**	**Birthday**
Jackie	5	April 5
Beth	8	January 14

Associate the Data Cells with the Appropriate Headers

Now that we've created headers, we need to associate the cells with the appropriate headers. There are two ways to do associate data cells with their headers.

The scope attribute

The scope attribute should be used on simple data tables such as the one in this example. Here is the markup for the table, using the scope attribute:

```
<table border="1" align="center">
<caption>Shelly's Daughters</caption>

<tr>
```

```
<th scope="col">Name</th>
<th scope="col">Age</th>
<th scope="col">Birthday</th>
</tr>

<tr>
<th scope="row">Jackie</th>
<td>5</td>
<td>April 5</td>
</tr>

<tr>
<th scope="row">Beth</th>
<td>8</td>
<td>January 14</td>
</tr>

</table>
```

The scope attribute tells the browser and screen reader that everything under the column is related to the header at the top, and everything to the right of the row header is related to that header. It's a straightforward concept.

The headers and id attributes

Another way to accomplish the same purpose is to use the headers and id attributes. This method is NOT recommended for simple tables such as the first example. The headers and id method should only be used when there is more than one logical level in a table, and when it is necessary to link more than two headers with a data cell. If we extend our original example, we can create a table that fits this criterion. In the table below, data have three headers each, so it is appropriate to use a more complex technique.

Shelly's Daughters

	Name	Age	Birthday
by birth	Jackie	5	April 5
	Beth	8	January 14
by marriage	Jenny	12	Feb 12

The markup looks like this:

```
<table border="1">
<caption>Shelly's Daughters</caption>

<tr>
<td> </td>
<th id="name">Name</th>
<th id="age">Age</th>
<th id="birthday">Birthday</th>
</tr>

<tr>
<th rowspan="2" id="birth">by birth</th>
<th id="jackie">Jackie</th>
<td headers="birth jackie age">5</td>
<td headers="birth jackie birthday">April 5</td>
</tr>

<tr>
<th id="beth">Beth</th>
<td headers="birth beth age">8</td>
<td headers="birth beth birthday">January 14</td>
</tr>

<tr>
<th id="step">by marriage</th>
<th id="jenny">Jenny</th>
<td headers ="step jenny age">12</td>
<td headers="step jenny birthday">Feb 12</td>
</tr>

</table>
```

Again, it should be emphasized that this method is more complex. It should be used with tables of a more complex nature, where the scope attribute will not work.

Another caveat: spanned rows and columns are not handled well by the JAWS screen reader, which is the most popular brand of screen reader. If at all possible, avoid complex data tables, or represent the data in a way that is less complex, preferably with no more than two headings applying to a single data cell.

Provide Names or Titles for Data Tables Using the <caption> Tag

Tables ought to have some sort of title or caption to them. This is properly done by using the caption tag, right after the opening <table> tag, like this:

```
<table border="1" align="center">
<caption>Shelly's Daughters</caption>
...
```

It is not absolutely necessary to have caption tags on every data table for the sake of accessibility, but it is still a good practice.

Avoid Spanned Rows or Columns

Important

The following information pertains to older versions of the JAWS screen reader. Newer versions of JAWS do not have the flaw explained below.

One of the most popular screen readers on the market, JAWS, does not read tables with spanned cells well at all, even when correct markup is used to associate data cells with their corresponding headers. As it turns out, JAWS handles spanned rows in tables very poorly. Let's look at an example of a table that spans more than one row:

View table – (click here to see in HTML)

Example of a Table with Spanned Rows

Dept. Code	Class #	Section	Max Enrollment	Current Enrollment	Rm. #	Days	Start Time	End Time	Inst.
BIO	100	1	15	13	5	Mon,Wed,Fri	10:00	11:00	Magde
	100	2	15	7	5	Tue,Thu	11:00	12:30	Indge
	205	1	15	9	6	Tue,Thu	09:00	10:30	Magde
	315	1	12	3	6	Mon,Wed,Fri	13:00	14:00	Indge
BUS	150	1	15	15	13	Mon,Wed,Fri	09:00	10:00	Roberts
	210	1	10	9	13	Mon,Wed,Fri	08:00	09:00	Rasid

The above table uses spanned rows in two instances: for the two cells on the left which contain the department codes BIO and BUS (which stand for biology and business). JAWS gets very confused with these spanned rows, and actually associates the **WRONG** headers with the data cells below. This means that the user will hear such nonsensical things as "Room number: Tue, Thu" and "Days: 11:00."

Don't despair. There are things that can be done to avoid totally confusing the users of screen readers, even if they are only work-arounds to make up for the programming deficits in JAWS.

Work-around #1: Make duplicate cells within the table, rather than one single row-spanned cell. See what work-around #1 (click here to see in HTML) would look like.

Work-around Number One

Dept. Code	Class Number	Section	Max Enrollment	Current Enrollment	Rm. #	Days	Start Time	End Time	Inst.
BIO	100	1	15	13	5	Mon,Wed,Fri	10:00	11:00	Magde
BIO	100	2	15	7	5	Tue,Thu	11:00	12:30	Indge
BIO	205	1	15	9	6	Tue,Thu	09:00	10:30	Magde
BIO	315	1	12	3	6	Mon,Wed,Fri	13:00	14:00	Indge
BUS	150	1	15	15	13	Mon,Wed,Fri	09:00	10:00	Roberts
BUS	210	1	10	9	13	Mon,Wed,Fri	08:00	09:00	Rasid

Work-around #2: We could also combine the first two columns into one. See what work-around #2 (click here to see in HTML) would look like.

Work-around Number Two

Dept. Code/ Class Number	Section	Max Enrollment	Current Enrollment	Rm #	Days	Start Time	End Time	Inst.
BIO 100	1	15	13	5	Mon,Wed,Fri	10:00	11:00	Magde
BIO 100	2	15	7	5	Tue,Thu	11:00	12:30	Indge
BIO 205	1	15	9	6	Tue,Thu	09:00	10:30	Magde
BIO 315	1	12	3	6	Mon,Wed,Fri	13:00	14:00	Indge
BUS 150	1	15	15	13	Mon,Wed,Fri	09:00	10:00	Roberts
BUS 210	1	10	9	13	Mon,Wed,Fri	08:00	09:00	Rasid

2. Accessible Data Tables - http://www.usability.com.au/resources/tables.cfm

The following table for "plum and pear" prices uses <th> for the headings.

Prices for black plums and bosca pears in Sydney

	Black Plums	Bosca Pears
Wholesale	$1.00	$1.50
Retail	$2.00	$2.50

```
<table border="1" summary="Black plums and bosca pears table with one level of row and column headers">
  <caption>Prices for black plums and bosca pears in Sydney</caption>
  <tr>
    <td></td>
    <th>Lemons</th>
    <th>Pears</th>
  </tr>
  <tr>
    <th>Wholesale</th>
    <td>$1.00</td>
    <td>$1.50</td>
  </tr>
  <tr>
    <th>Retail</th>
    <td>$2.00</td>
    <td>$2.50</td>
  </tr>
</table>
```

A screen reader like JAWS 5 will read the 'wholesale' row like this:

"wholesale, dollar one point OO, dollar one point five O"

If the user wants to know the wholesale price of pears, JAWS will voice the selected data cell like this:

"column three, row two, pears wholesale, dollar one point five O"

Abbreviation

The 'abbr' attribute can be used to provide an abbreviation for long headers so that the entire header is not read out every time. In the example of a simple data table below, 'abbr' is included in the <th> tags for the column headings. Some screen readers will then only read the full headings "Black Plums" and "Bosca Pears" the first time they are encountered, and the abbreviations "plums" and "pears" will be read on all the other occasions.

```
<table border="1" summary="Black plums and bosca pears table with one level of row and column headers">
  <caption>Prices for black plums and bosca pears in Sydney</caption>
  <tr>
    <td></td>
    <th abbr="plums">Black Plums</th>
    <th abbr="pears">Bosca Pears</th>
  </tr>
  <tr>
    <th>Wholesale</th>
    <td>$1.00</td>
    <td>$1.50</td>
  </tr>
  <tr>
    <th>Retail</th>
    <td>$2.00</td>
    <td>$2.50</td>
  </tr>
</table>
```

ID and Headers

HTML 4 introduced the 'headers' attribute for table cells <td>. This attribute is used in conjunction with the id attribute within a table heading <th> to allow any cell or cells to be associated with a heading or headings.

The following table, which is from the table test page, for orange and apple prices uses id and headers.

Imported and domestic orange and apple prices in Sydney and Melbourne

	Imported		Domestic	
	Oranges	**Apples**	**Oranges**	**Apples**
Sydney				
Wholesale	$1.00	$1.25	$1.20	$1.00
Retail	$2.00	$3.00	$1.80	$1.60
Melbourne				
Wholesale	$1.20	$1.30	$1.00	$0.80
Retail	$1.60	$2.00	$2.00	$1.50

QUESTION: what is the wholesale price of imported apples in Sydney?

The use of id and headers means that most people who rely on screen readers will be able to obtain the answer to this question. The source code for part of this table with the relevant id and headers highlighted follows:

```
<table border="1" summary="Wholesale and retail prices of imported and domestic oranges and apples in Sydney and Melbourne. There are two levels of column headings.">
  <caption>
    Imported and domestic orange and apple prices in Sydney and Melbourne
  </caption>
  <thead>
    <tr>
      <td></td>
      <th colspan="2" id="imported">Imported</th>
      <th colspan="2" id="domestic">Domestic</th>
    </tr>
    <tr>
      <td></td>
      <th id="oranges-imp">Oranges</th>
      <th id="apples-imp">Apples</th>
      <th id="oranges-dom">Oranges</th>
      <th id="apples-dom">Apples</th>
    </tr>
  </thead>
  <tbody>
    <tr>
```

```
    <th id="sydney" colspan="5">Sydney</th>
  </tr>
  <tr>
    <th headers="sydney" id="wholesale-sydney">Wholesale</th>
    <td headers="imported oranges-imp sydney wholesale-sydney">$1.00</td>
    <td headers="imported apples-imp sydney wholesale-sydney">$1.25</td>
    <td headers="domestic oranges-dom sydney wholesale-sydney">$1.20</td>
    <td headers="domestic apples-dom sydney wholesale-sydney">$1.00</td>
  </tr>
  <tr>
    <th headers="sydney" id="retail-sydney">Retail</th>
...THE REST OF THE TABLE CODE ...
```

Very Complex Data Tables

The table test page contains the following data table about cherry and apricot prices, which has three levels of column headings and two levels of row headings that use id and headers.

Imported and domestic cherry and apricot prices in Perth and Adelaide

	Imported			Domestic		
	Apricots	Cherries		Apricots	Cherries	
		A Grade	B Grade		A Grade	B Grade
Perth						
Wholesale	$1.00	$9.00	$6.00	$1.20	$13.00	$9.00
Retail	$2.00	$12.00	$8.00	$1.80	$16.00	$12.50
Adelaide						
Wholesale	$1.20	N/A	$7.00	$1.00	$11.00	$6.00
Retail	$1.60	N/A	$11.00	$2.00	$13.00	$10.00

QUESTION: What is the wholesale price of imported A-grade cherries in Perth?

Sections of code for the cherry and apricot table follows with the relevant id and headers highlighted.

```
<thead>
  <tr>
    <td></td>
    <th id="imp" colspan="3">Imported</th>
    <th id="dom" colspan="3">Domestic</th>
  </tr>
  <tr>
    <td></td>
    <th headers="imp" id="imp-apr">Apricots</th>
    <th headers="imp" id="imp-che" colspan="2">Cherries</th>
    <th headers="dom" id="dom-apr">Apricots</th>
    <th headers="dom" id="dom-che" colspan="2">Cherries</th>
  </tr>
  <tr>
    <td></td>
    <td></td>
    <th headers="imp imp-che" id="imp-che-agrade">A Grade</th>
    <th headers="imp imp-che" id="imp-che-bgrade">B Grade</th>
    <td></td>
    <th headers="dom dom-che" id="dom-che-agrade">A Grade</th>
    <th headers="dom dom-che" id="dom-che-bgrade">B Grade</th>
  </tr>
</thead>
... MORE CODE ...

<tbody>
  <tr>
    <th id="perth" colspan="7">Perth</th>
    </tr><tr>
    <th headers="perth" id="perth-wholesale">Wholesale</th>
    <td headers="imp imp-apr perth perth-wholesale">$1.00</td>
    <td headers="imp imp-che imp-che-agrade perth perth-wholesale">$9.00</td>
    <td headers="imp imp-che imp-che-bgrade perth perth-wholesale"> $6.00</td>
    <td headers="dom dom-apr perth perth-wholesale">$1.20</td>
... THE REST OF THE TABLE CODE ...
```

Although the heading structure of this table is complex, by using a combination of keyboard commands, most JAWS users will be able to locate the cell that is likely to contain the information they require.

With the focus on the appropriate cell ($9.00), JAWS will "say the current cell" like this:

"Row five, column three, imported Perth wholesale cherries A grade, dollar nine point O O"

3. Data and Layout Tables - http://www.doit.wisc.edu/accessibility/online-course/standards/tables.htm

Explanation - Standard (h) - Complex Data Tables

What is a complex data table? A complex data table includes more than two levels of headings. Complex data tables are not common on the web. If the complexity of the concepts presented in this section overwhelm you, don't panic. Chances are high that you will not be working with tables that are this complex.

The Enrollment Comparison example that follows illustrates a data table with three levels of headings. (Data in this table is fictional.)

Enrollment Comparison			
	2000	**2001**	**2002**
University of Wisconsin-Stevens Point			
Freshman	2200	2150	2000
Sophomore	2000	2050	2010
Junior	1992	1987	2000
Senior	1875	1990	1900
Graduate	500	475	510
Total	8566	8652	8420
University of Wisconsin-Platteville			
Freshman	2250	2260	2100
Sophomore	2075	2000	2040
Junior	1900	2000	1900
Senior	1775	1850	1950
Graduate	425	450	400
Total	8425	8560	8390

HTML code from the Enrollment Comparison table.

<table width="50%" border="1" cellspacing="0" cellpadding="0" summary="Comparison of Enrollment between the University of Wisconsin-Stevens Point and the University of Wisconsin-Platteville. Enrollment numbers are presented vertically by the calendar years of 2000, 2001, and 2002 and horizontally as Freshman, Sophomore, Junion, Senior, Graduate, and Total. Stevens Point is presented first, Platteville second.">

```
<caption>
Enrollment Comparison
</caption>
<tr>
<th> </th>
<th id="r1_c2">2000</th>
<th id="r1_c3">2001</th>
<th id="r1_c4">2002</th>
</tr>
<tr>
<th id="r2_c1">University of Wisconsin-Stevens Point</th>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<th class="thspecialheader" id="r3_c1">Freshman</th>
<td headers="r2_c1 r1_c2 r3_c1">2200</td>
<td headers="r2_c1 r1_c3 r3_c1">2150</td>
<td headers="r2_c1 r1_c4 r3_c1">2000</td>
</tr>
<tr>
<th class="thspecialheader" id="r4_c1">Sophomore</th>
<td headers="r2_c1 r1_c2 r4_c1">2000</td>
<td headers="r2_c1 r1_c3 r4_c1">2050</td>
<td headers="r2_c1 r1_c4 r4_c1">2010</td>
</tr>
<tr>
<th class="thspecialheader" id="r5_c1">Junior</th>
<td headers="r2_c1 r1_c2 r5_c1">1992</td>
<td headers="r2_c1 r1_c3 r5_c1">1987</td>
<td headers="r2_c1 r1_c4 r5_c1">2000</td>
</tr>
<tr>
<th class="thspecialheader" id="r6_c1">Senior</td>
<td headers="r2_c1 r1_c2 r6_c1">1875</td>
<td headers="r2_c1 r1_c3 r6_c1">1990</td>
<td headers="r2_c1 r1_c4 r6_c1">1900</td>
</tr>
<tr>
<th class="thspecialheader" id="r7_c1">Graduate</td>
<td headers="r2_c1 r1_c2 r7_c1">500</td>
<td headers="r2_c1 r1_c3 r7_c1">475</td>
<td headers="r2_c1 r1_c4 r7_c1">510</td>
</tr>
<tr>
```

```
<th class="thspecialheader" id="r8_c1">Total</td>
<td headers="r2_c1 r1_c2 r8_c1">8566</td>
<td headers="r2_c1 r1_c3 r8_c1">8652</td>
<td headers="r2_c1 r1_c4 r8_c1">8420</td>
</tr>
<tr>
<th id="r9_c1">University of Wisconsin-Platteville</th>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<th class="thspecialheader" id="r10_c1">Freshman</th>
<td headers="r9_c1 r1_c2 r10_c1">2250</td>
<td headers="r9_c1 r1_c3 r10_c1">2260</td>
<td headers="r9_c1 r1_c4 r10_c1">2100</td>
</tr>
<tr>
<th class="thspecialheader" id="r11_c1">Sophomore</th>
<td headers="r9_c1 r1_c2 r11_c1">2075</td>
<td headers="r9_c1 r1_c3 r11_c1">2000</td>
<td headers="r9_c1 r1_c4 r11_c1">2040</td>
</tr>
<tr>
<th class="thspecialheader" id="r12_c1">Junior</th>
<td headers="r9_c1 r1_c2 r12_c1">1900</td>
<td headers="r9_c1 r1_c3 r12_c1">2000</td>
<td headers="r9_c1 r1_c4 r12_c1">1900</td>
</tr>
<tr>
<th class="thspecialheader" id="r13_c1">Senior</th>
<td headers="r9_c1 r1_c2 r13_c1">1775</td>
<td headers="r9_c1 r1_c3 r13_c1">1850</td>
<td headers="r9_c1 r1_c4 r13_c1">1950</td>
</tr>
<tr>
<th class="thspecialheader" id="r14_c1">Graduate</th>
<td headers="r9_c1 r1_c2 r14_c1">425</td>
<td headers="r9_c1 r1_c3 r14_c1">450</td>
<td headers="r9_c1 r1_c4 r14_c1">400</td>
</tr>
<tr>
<th class="thspecialheader" id="r15_c1">Total</th>
<td headers="r9_c1 r1_c2 r15_c1">8425</td>
<td headers="r9_c1 r1_c3 r15_c1">8560</td>
<td headers="r9_c1 r1_c4 r15_c1">8390</td>
```

</tr>
</table>

To test your knowledge take a quick 4 question quiz at:
http://www.doit.wisc.edu/accessibility/online-course/quiz.asp?quiz=7

4. Table Manners: Creating Accessible Tables for both Layout and Data - http://www.mcu.org.uk/articles/tables.html

Accessible Data Tables

The first step towards making data tables accessible is to use existing tags such as TH, TD and new HTML 4 tags such as CAPTION and SUMMARY correctly. When used as intended these tags will provide valuable meta information about your table. You should use the following tags to indicate the type of content contained with a table cell:

TH for headers

TD for data.

CAPTION to give your table a short title - indicating the tables contents.

The **summary** attribute can also be used to provide a verbose description of the purpose of the table - to help someone who is using a voice or Braille browser. It can also be used to help explain the information within a table if it is of a particularly complex nature. I have seen examples of the summary attribute being used to re-state the entire contents of a table - as an alternative to providing a link to a single column version on another page.

Here is a simple example:

Jim, Pat and John's favourite things

	Jim	**Pat**	**John**
Favourite colour	Green	Turquoise	Yellow
Favourite Pastime	Photography	Swimming	Badminton

Here is what the HTML looks like:

<TABLE summary="Example table showing Jim, Pat and John's favourite colours." border="1" cellpadding="15">
<CAPTION>Jim, Pat and John's favourite THings</CAPTION>

```
<TR>
<TH></TH>
<TH>Jim</TH>
<TH>Pat</TH>
<TH>John</TH>
</TR>
<TR>
<TH>Favourite colour</TH>
<TD>Green</TD>
<TD>Turquoise</TD>
<TD>yellow</TD>
</TR>
<TR>
<TH>Favourite pastime</TH>
<TD>photography</TD>
<TD>swimming</TD>
<TD>badmington</TD>
</TR>
</TABLE>
```

The summary attribute is not visible to those using standard GUI (Graphical User Interface) Web browsers - but can be used by assistive technologies such as screen and braille readers.

work-arounds for CSS bugs with Internet Explorer 5x, (W=Win '95 M=Mac)

CSS Property/ Problem	Description	Workaround	Notes/Footnotes
absolute/relative positioning	absolute child loses position	set the width property of the parent element	W demo
Classes set as number (i.e. P.1)	recognized due to error-recovery	Illegal syntax; don't use	W95, NT (6)
color	f0f0f0 rendered	illegal syntax; include the required '#'	W95,NT
width on body is ignored	ignored	use width on DIV instead	W95, NT

ID and HEADERS

By giving each table heading a unique label (using the 'id' attribute) - and then associating each of the data cells with that label, using the 'header' attribute, new Web browsers can give appropriate feedback for those using assistive technologies.

An example should make this clearer: Here is how the HTML looks for the above table using the 'id' attribute in the headers and the 'header' attribute in each of the data cells.

```
<TABLE border="1" summary="Workarounds for CSS bugs with Internet Explorer 5x, (W=Win '95 M=Mac)" cellpadding="10">
<CAPTION style="font-size: 120%; font-style: bold">Workarounds for CSS bugs with Internet Explorer 5x, (W=Win '95 M=Mac)</CAPTION>
<TR>
<TH id="header1">CSS Property/ Problem</TH>
<TH id="header2">Description</TH>
<TH id="header3">Workaround</TH>
<TH id="header4">Notes/Footnotes</TH>
</TR>
<TR>
<TD headers="header1">absolute/relative positioning</TH>
<TD headers="header2"> absolute child loses position</TD>
<TD headers="header3"> set the width property of the parent element</TD>
<TD headers="header4">W demo<</TD>
</TR>
<TR>
<TD headers="header1">Classes set as number (i.e. P.1)</TD>
<TD headers="header2">recognized due to error-recovery</TD>
<TD headers="header3">Illegal syntax; don't use</TD>
<TD headers="header4">W95, NT (6)</TD>
</TR>
<TR>
<TD headers="header1">color</TD>
<TD headers="header2">f0f0f0 rendered</TD>
<TD headers="header3">illegal syntax; include the required '#'</TD>
<TD headers="header4">W95,NT</TD>
</TR>
<TR>
<TD headers="header1">width on body is ignored</TD>
<TD headers="header2"> ignored</TD>
<TD headers="header3">use width on DIV instead</TD>
<TD headers="header4"> W95, NT</TD>
</TR>
</TABLE>
```

A screen reader will read out the first data row as follows:

CSS Property/ Problem: absolute/relative,

Description: absolute child loses position,

Workaround: set the width property of the parent element,

Notes/Footnotes: W demo

And the second row:

CSS Property/ Problem: Classes set as number (i.e. P.1),

Description: recognized due to error-recovery,

Workaround: Illegal syntax; don't use,

Notes/Footnotes: W95, NT (6)

And so on.

SCOPE

The 'headers' attribute works even when table headings are not aligned logically with the data cells they apply to. i.e. when headers and the corresponding data are not part of the same column or row.

For simple tables, where the column or row headers do correspond, the id/headers pair can be replaced with the 'scope' attribute. The scope attribute can have a value of 'col', 'row', colgroup or rowgroup. When using SCOPE with the value of 'col' (i.e scope ="col") - you are saying to the browser (or relevant user agent) ' everything in this column relates to this heading. An example should make this clear:

Using the first two data rows from the previous table, 'scope' is used as follows:

```
<TABLE border="1" summary="Workarounds for CSS bugs with Internet Explorer 5x, (W=Win '95 M=Mac)" cellpadding="10">
<CAPTION style="font-size: 120%; font-style: bold">Workarounds for CSS bugs with Internet Explorer 5x, (W=Win '95 M=Mac)</CAPTION>
<TR>
<TH scope="col">CSS Property/ Problem</TH>
<TH scope="col">Description</TH>
<TH scope="col">Workaround</TH>
<TH scope="col">Notes/Footnotes</TH>
</TR>
<TR>
<TD>absolute/relative positioning</TH>
<TD> absolute child loses position</TD>
<TD> set the width property of the parent element</TD>
<TD>W demo<</TD>
</TR>
<TR>
<TD>Classes set as number (i.e. P.1)</TD>
<TD>recognized due to error-recovery</TD>
<TD>Illegal syntax; don't use</TD>
<TD>W95, NT (6)</TD>
</TR>
</TABLE>
```

More advanced uses of the scope attribute can be found on the World Wide Web Consortiums Website at: http://www.w3.org/TR/REC-html40/struct/tables.html#h-11.4

5. Layout Tables - http://www.w3.org/TR/html4/struct/tables.html

Here's a simple table that illustrates some of the features of the HTML table model. The following table definition:

```
<TABLE border="1"
       summary="This table gives some statistics about fruit
                flies: average height and weight, and percentage
                with red eyes (for both males and females).">
<CAPTION><EM>A test table with merged cells</EM></CAPTION>
<TR><TH rowspan="2"><TH colspan="2">Average
   <TH rowspan="2">Red<BR>eyes
<TR><TH>height<TH>weight
<TR><TH>Males<TD>1.9<TD>0.003<TD>40%
<TR><TH>Females<TD>1.7<TD>0.002<TD>43%
</TABLE>
```

might be rendered something like this on a tty device:

```
      A test table with merged cells
   /---------------------------------------\
   |        |     Average     |   Red    |
   |        |-----------------|  eyes    |
   |        | height | weight |          |
   |----------------------------------------|
   | Males  | 1.9    | 0.003  |   40%    |
   |----------------------------------------|
   | Females| 1.7    | 0.002  |   43%    |
   \---------------------------------------/
```

or like this by a graphical user agent:

A test table with merged cells

	Average		Red eyes
	height	weight	
Males	1.9	0.003	40%
Females	1.7	0.002	43%

11.2.1 The TABLE element

```
<!ELEMENT TABLE - -
   (CAPTION?, (COL*|COLGROUP*), THEAD?, TFOOT?, TBODY+)>
<!ATTLIST TABLE                    -- table element --
 %attrs;                    -- %coreattrs, %i18n, %events --
 summary    %Text;       #IMPLIED  -- purpose/structure for speech output--
 width      %Length;     #IMPLIED  -- table width --
```

```
border      %Pixels;      #IMPLIED  -- controls frame width around table --
frame       %TFrame;      #IMPLIED  -- which parts of frame to render --
rules       %TRules;      #IMPLIED  -- rulings between rows and cols --
cellspacing %Length;      #IMPLIED  -- spacing between cells --
cellpadding %Length;      #IMPLIED  -- spacing within cells --
>
```

Start tag: ***required****, End tag:* ***required***

To specify a right-to-left table, set the dir attribute as follows:

```
<TABLE dir="RTL">
...the rest of the table...
</TABLE>
```

11.2.2 Table Captions: The CAPTION element

```
<!ELEMENT CAPTION  - - (%inline;)*     -- table caption -->

<!ATTLIST CAPTION
  %attrs;                  -- %coreattrs, %i18n, %events --
  >
```

Start tag: ***required****, End tag:* ***required***

When present, the CAPTION element's text should describe the nature of the table. The CAPTION element is only permitted immediately after the TABLE start tag. A TABLE element may only contain one CAPTION element.

11.2.3 Row groups: the THEAD, TFOOT, and TBODY elements

```
<!ELEMENT THEAD    - O (TR)+          -- table header -->
<!ELEMENT TFOOT    - O (TR)+          -- table footer -->
```

Start tag: ***required****, End tag:* ***optional***

```
<!ELEMENT TBODY    O O (TR)+          -- table body -->
```

Start tag: ***optional****, End tag:* ***optional***

```
<!ATTLIST (THEAD|TBODY|TFOOT)          -- table section --
  %attrs;                  -- %coreattrs, %i18n, %events --
  %cellhalign;               -- horizontal alignment in cells --
  %cellvalign;               -- vertical alignment in cells --
  >
```

Table rows may be grouped into a table head, table foot, and one or more table body sections, using the THEAD, TFOOT and TBODY elements, respectively. This division enables user agents to support scrolling of table bodies independently of the table head

and foot. When long tables are printed, the table head and foot information may be repeated on each page that contains table data.

The table head and table foot should contain information about the table's columns. The table body should contain rows of table data.

This example illustrates the order and structure of table heads, feet, and bodies.

```
<TABLE>
<THEAD>
   <TR> ...header information...
</THEAD>
<TFOOT>
   <TR> ...footer information...
</TFOOT>
<TBODY>
   <TR> ...first row of block one data...
   <TR> ...second row of block one data...
</TBODY>
<TBODY>
   <TR> ...first row of block two data...
   <TR> ...second row of block two data...
   <TR> ...third row of block two data...
</TBODY>
</TABLE>
```

TFOOT must appear before TBODY within a TABLE definition so that user agents can render the foot before receiving all of the (potentially numerous) rows of data. The following summarizes which tags are required and which may be omitted:

- The TBODY start tag is always required except when the table contains only one table body and no table head or foot sections. The TBODY end tag may always be safely omitted.
- The start tags for THEAD and TFOOT are required when the table head and foot sections are present respectively, but the corresponding end tags may always be safely omitted.

The table of the previous example could be shortened by removing certain end tags, as in:

```
<TABLE>
<THEAD>
   <TR> ...header information...
<TFOOT>
   <TR> ...footer information...
<TBODY>
   <TR> ...first row of block one data...
   <TR> ...second row of block one data...
<TBODY>
   <TR> ...first row of block two data...
   <TR> ...second row of block two data...
   <TR> ...third row of block two data...
```

```
</TABLE>
```

The THEAD, TFOOT, and TBODY sections must contain the same number of columns.

The COLGROUP element

```
<!ELEMENT COLGROUP - O (COL)*          -- table column group -->
<!ATTLIST COLGROUP
  %attrs;                    -- %coreattrs, %i18n, %events --
  span       NUMBER       1       -- default number of columns in group --
  width      %MultiLength; #IMPLIED  -- default width for enclosed COLs --
  %cellhalign;                 -- horizontal alignment in cells --
  %cellvalign;                 -- vertical alignment in cells --
  >
```

Start tag: ***required****, End tag:* ***optional***

The COLGROUP element creates an explicit column group. The number of columns in the column group may be specified in two, mutually exclusive ways:

1. The element's span attribute (default value 1) specifies the number of columns in the group.
2. Each COL element in the COLGROUP represents one or more columns in the group.

The advantage of using the span attribute is that authors may group together information about column widths. Thus, if a table contains forty columns, all of which have a width of 20 pixels, it is easier to write:

```
<COLGROUP span="40" width="20">
</COLGROUP>
```

than:

```
<COLGROUP>
  <COL width="20">
  <COL width="20">
  ...a total of forty COL elements...
</COLGROUP>
```

When it is necessary to single out a column (e.g., for style information, to specify width information, etc.) within a group, authors must identify that column with a COL element. Thus, to apply special style information to the last column of the previous table, we single it out as follows:

```
<COLGROUP width="20">
  <COL span="39">
  <COL id="format-me-specially">
</COLGROUP>
```

The width attribute of the COLGROUP element is inherited by all 40 columns. The first COL element refers to the first 39 columns (doing nothing special to them) and the second one assigns an id value to the fortieth column so that style sheets may refer to it.

The table in the following example contains two column groups. The first column group contains 10 columns and the second contains 5 columns. The default width for each column in the first column group is 50 pixels. The width of each column in the second column group will be the minimum required for that column.

```
<TABLE>
<COLGROUP span="10" width="50">
<COLGROUP span="5" width="0*">
<THEAD>
<TR><TD> ...
</TABLE>
```

The COL element

```
<!ELEMENT COL     - O EMPTY          -- table column -->
<!ATTLIST COL                        -- column groups and properties --
  %attrs;                      -- %coreattrs, %i18n, %events --
  span       NUMBER       1        -- COL attributes affect N columns --
  width      %MultiLength;  #IMPLIED  -- column width specification --
  %cellhalign;                 -- horizontal alignment in cells --
  %cellvalign;                 -- vertical alignment in cells --
  >
```

Start tag: ***required****, End tag:* ***forbidden***

The COL element allows authors to group together attribute specifications for table columns. The COL does **not** group columns together structurally -- that is the role of the COLGROUP element. COL elements are empty and serve only as a support for attributes. They may appear inside or outside an explicit column group (i.e., COLGROUP element).

The width attribute for COL refers to the width of each column in the element's span.

Calculating the number of columns in a table

For example, for each of the following tables, the two column calculation methods should result in three columns. The first three tables may be rendered incrementally.

```
<TABLE>
<COLGROUP span="3"></COLGROUP>
<TR><TD> ...
...rows...
</TABLE>

<TABLE>
<COLGROUP>
<COL>
```

```
<COL span="2">
</COLGROUP>
<TR><TD> ...
...rows...
</TABLE>

<TABLE>
<COLGROUP>
<COL>
</COLGROUP>
<COLGROUP span="2">
<TR><TD> ...
...rows...
</TABLE>

<TABLE>
<TR>
  <TD><TD><TD>
</TR>
</TABLE>
```

Once the (visual) user agent has received the table's data: the available horizontal space will be alloted by the user agent as follows: First the user agent will allot 30 pixels to columns one and two. Then, the minimal space required for the third column will be reserved. The remaining horizontal space will be divided into six equal portions (since 2* + 1* + 3* = 6 portions). Column four (2*) will receive two of these portions, column five (1*) will receive one, and column six (3*) will receive three.

```
<TABLE>
<COLGROUP>
   <COL width="30">
<COLGROUP>
   <COL width="30">
   <COL width="0*">
   <COL width="2*">
<COLGROUP align="center">
   <COL width="1*">
   <COL width="3*" align="char" char=":">
<THEAD>
<TR><TD> ...
...rows...
</TABLE>
```

We have set the value of the align attribute in the third column group to "center". All cells in every column in this group will inherit this value, but may override it. In fact, the final COL does just that, by specifying that every cell in the column it governs will be aligned along the ":" character.

In the following table, the column width specifications allow the user agent to format the table incrementally:

```
<TABLE width="200">
<COLGROUP span="10" width="15">
<COLGROUP width="*">
  <COL id="penultimate-column">
  <COL id="last-column">
<THEAD>
<TR><TD> ...
...rows...
</TABLE>
```

The first ten columns will be 15 pixels wide each. The last two columns will each receive half of the remaining 50 pixels. Note that the COL elements appear only so that an id value may be specified for the last two columns.

Note. *Although the width attribute on the TABLE element is not deprecated, authors are encouraged to use style sheets to specify table widths.*

11.2.5 Table rows: The TR element

```
<!ELEMENT TR      - O (TH|TD)+        -- table row -->
<!ATTLIST TR                          -- table row --
  %attrs;                     -- %coreattrs, %i18n, %events --
  %cellhalign;                  -- horizontal alignment in cells --
  %cellvalign;                  -- vertical alignment in cells --
  >
```

Start tag: ***required****, End tag:* ***optional***

The TR elements acts as a container for a row of table cells. The end tag may be omitted.

This sample table contains three rows, each begun by the TR element:

```
<TABLE summary="This table charts the number of cups
                of coffee consumed by each senator, the type
                of coffee (decaf or regular), and whether
                taken with sugar.">
<CAPTION>Cups of coffee consumed by each senator</CAPTION>
<TR> ...A header row...
<TR> ...First row of data...
<TR> ...Second row of data...
...the rest of the table...
</TABLE>
```

11.2.6 Table cells: The TH and TD elements

```
<!ELEMENT (TH|TD)  - O (%flow;)*       -- table header cell, table data cell-->

<!-- Scope is simpler than headers attribute for common tables -->
<!ENTITY % Scope "(row|col|rowgroup|colgroup)">

<!-- TH is for headers, TD for data, but for cells acting as both use TD -->
```

```
<!ATTLIST (TH|TD)                   -- header or data cell --
  %attrs;                           -- %coreattrs, %i18n, %events --
  abbr        %Text;        #IMPLIED  -- abbreviation for header cell --
  axis        CDATA         #IMPLIED  -- comma-separated list of related headers--
  headers     IDREFS        #IMPLIED  -- list of id's for header cells --
  scope       %Scope;       #IMPLIED  -- scope covered by header cells --
  rowspan     NUMBER        1         -- number of rows spanned by cell --
  colspan     NUMBER        1         -- number of cols spanned by cell --
  %cellhalign;                        -- horizontal alignment in cells --
  %cellvalign;                        -- vertical alignment in cells --
  >
```

Start tag: ***required****, End tag:* ***optional***

Table cells may contain two types of information: header information and data. This distinction enables user agents to render header and data cells distinctly, even in the absence of style sheets. For example, visual user agents may present header cell text with a bold font. Speech synthesizers may render header information with a distinct voice inflection.

For example, the following table contains four columns of data, each headed by a column description.

```
<TABLE summary="This table charts the number of cups
                of coffee consumed by each senator, the type
                of coffee (decaf or regular), and whether
                taken with sugar.">
<CAPTION>Cups of coffee consumed by each senator</CAPTION>
<TR>
  <TH>Name</TH>
  <TH>Cups</TH>
  <TH>Type of Coffee</TH>
  <TH>Sugar?</TH>
<TR>
  <TD>T. Sexton</TD>
  <TD>10</TD>
  <TD>Espresso</TD>
  <TD>No</TD>
<TR>
  <TD>J. Dinnen</TD>
  <TD>5</TD>
  <TD>Decaf</TD>
  <TD>Yes</TD>
</TABLE>
```

A user agent rendering to a tty device might display this as follows:

Name	**Cups**	**Type of Coffee**	**Sugar?**
T. Sexton	10	Espresso	No
J. Dinnen	5	Decaf	Yes

Cells that span several rows or columns

Cells may span several rows or columns. The number of rows or columns spanned by a cell is set by the rowspan and colspan attributes for the TH and TD elements.

In this table definition, we specify that the cell in row four, column two should span a total of three columns, including the current column.

```
<TABLE border="1">
<CAPTION>Cups of coffee consumed by each senator</CAPTION>
<TR><TH>Name<TH>Cups<TH>Type of Coffee<TH>Sugar?
<TR><TD>T. Sexton<TD>10<TD>Espresso<TD>No
<TR><TD>J. Dinnen<TD>5<TD>Decaf<TD>Yes
<TR><TD>A. Soria<TD colspan="3"><em>Not available</em>
</TABLE>
```

This table might be rendered on a tty device by a visual user agent as follows:

```
Cups of coffee consumed by each senator
 ---------------------------------------
|   Name  |Cups|Type of Coffee|Sugar?|
 ---------------------------------------
|T. Sexton|10  |Espresso      |No    |
 ---------------------------------------
|J. Dinnen|5   |Decaf         |Yes   |
 ---------------------------------------
|A. Soria |Not available              |
 ---------------------------------------
```

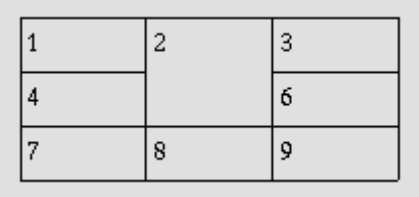
The next example illustrates (with the help of table borders) how cell definitions that span more than one row or column affect the definition of later cells. Consider the following table definition:

```
<TABLE border="1">
<TR><TD>1 <TD rowspan="2">2 <TD>3
<TR><TD>4 <TD>6
<TR><TD>7 <TD>8 <TD>9
</TABLE>
```

As cell "2" spans the first and second rows, the definition of the second row will take it into account. Thus, the second TD in row two actually defines the row's third cell. Visually, the table might be rendered to a tty device as:

```
-------------
| 1 | 2 | 3 |
----|   |----
| 4 |   | 6 |
----|---|----
| 7 | 8 | 9 |
-------------
```

while a graphical user agent might render this as:



Note that if the TD defining cell "6" had been omitted, an extra empty cell would have been added by the user agent to complete the row.

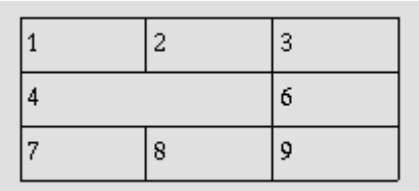
Similarly, in the following table definition:

```
<TABLE border="1">
<TR><TD>1 <TD>2 <TD>3
<TR><TD colspan="2">4 <TD>6
<TR><TD>7 <TD>8 <TD>9
</TABLE>
```

cell "4" spans two columns, so the second TD in the row actually defines the third cell ("6"):

```
-------------
| 1 | 2 | 3 |
--------|----
| 4     | 6 |
--------|----
| 7 | 8 | 9 |
-------------
```

A graphical user agent might render this as:



Defining overlapping cells is an error. User agents may vary in how they handle this error (e.g., rendering may vary).

The following illegal example illustrates how one might create overlapping cells. In this table, cell "5" spans two rows and cell "7" spans two columns, so there is overlap in the cell between "7" and "9":

```
<TABLE border="1">
<TR><TD>1 <TD>2 <TD>3
<TR><TD>4 <
```

11.3.1 Borders and rules

To help distinguish the cells of a table, we can set the border attribute of the TABLE element. Consider a previous example:

```
<TABLE border="1"
      summary="This table charts the number of cups
               of coffee consumed by each senator, the type
               of coffee (decaf or regular), and whether
               taken with sugar.">
<CAPTION>Cups of coffee consumed by each senator</CAPTION>
<TR>
  <TH>Name</TH>
  <TH>Cups</TH>
  <TH>Type of Coffee</TH>
  <TH>Sugar?</TH>
<TR>
  <TD>T. Sexton</TD>
  <TD>10</TD>
  <TD>Espresso</TD>
  <TD>No</TD>
<TR>
  <TD>J. Dinnen</TD>
  <TD>5</TD>
  <TD>Decaf</TD>
  <TD>Yes</TD>
</TABLE>
```

The following settings should be observed by user agents for backwards compatibility.

- Setting border="0" implies frame="void" and, unless otherwise specified, rules="none".
- Other values of border imply frame="border" and, unless otherwise specified, rules="all".
- The value "border" in the start tag of the TABLE element should be interpreted as the value of the frame attribute. It implies rules="all" and some default (non-zero) value for the border attribute.

For example, the following definitions are equivalent:

```
<TABLE border="2">
<TABLE border="2" frame="border" rules="all">
```

as are the following

```
<TABLE border>
<TABLE frame="border" rules="all">
```

11.3.2 Horizontal and vertical alignment

The following attributes may be set for different table elements (see their definitions).

```
<!-- horizontal alignment attributes for cell contents -->
<!ENTITY % cellhalign
  "align      (left|center|right|justify|char) #IMPLIED
   char       %Character;    #IMPLIED  -- alignment char, e.g. char=':' --
   charoff    %Length;       #IMPLIED  -- offset for alignment char --"
  >
<!-- vertical alignment attributes for cell contents -->
<!ENTITY % cellvalign
  "valign     (top|middle|bottom|baseline) #IMPLIED"
  >
```

Attribute definitions

The table in this example aligns a row of currency values along a decimal point. We set the alignment character to "." explicitly.

```
<TABLE border="1">
<COLGROUP>
<COL><COL align="char" char=".">
<THEAD>
<TR><TH>Vegetable <TH>Cost per kilo
<TBODY>
<TR><TD>Lettuce       <TD>$1
<TR><TD>Silver carrots <TD>$10.50
<TR><TD>Golden turnips <TD>$100.30
</TABLE>
```

The formatted table may resemble the following:

```
-----------------------------
|   Vegetable  |Cost per kilo|
|--------------|-------------|
|Lettuce       |       $1    |
|--------------|-------------|
|Silver carrots|      $10.50|
|--------------|-------------|
|Golden turnips|     $100.30|
-----------------------------
```

When the contents of a cell contain more than one instance of the alignment character specified by char and the contents wrap, user agent behavior is undefined. Authors should therefore be attentive in their use of char.

Note. *Visual user agents typically render TH elements vertically and horizontally centered within the cell and with a bold font weight.*

11.4.1 Associating header information with data cells

Non-visual user agents such as speech synthesizers and Braille-based devices may use the following TD and TH element attributes to render table cells more intuitively:

- For a given data cell, the headers attribute lists which cells provide pertinent header information. For this purpose, each header cell must be named using the id attribute. Note that it's not always possible to make a clean division of cells into headers or data. You should use the TD element for such cells together with the id or scope attributes as appropriate.
- For a given header cell, the scope attribute tells the user agent the data cells for which this header provides information. Authors may choose to use this attribute instead of headers according to which is more convenient; the two attributes fulfill the same function. The headers attribute is generally needed when headers are placed in irregular positions with respect to the data they apply to.
- The abbr attribute specifies an abbreviated header for header cells so that user agents may render header information more rapidly.

In the following example, we assign header information to cells by setting the headers attribute. Each cell in the same column refers to the same header cell (via the id attribute).

```
<TABLE border="1"
    summary="This table charts the number of cups
          of coffee consumed by each senator, the type
          of coffee (decaf or regular), and whether
          taken with sugar.">
<CAPTION>Cups of coffee consumed by each senator</CAPTION>
<TR>
  <TH id="t1">Name</TH>
  <TH id="t2">Cups</TH>
  <TH id="t3" abbr="Type">Type of Coffee</TH>
  <TH id="t4">Sugar?</TH>
<TR>
  <TD headers="t1">T. Sexton</TD>
  <TD headers="t2">10</TD>
  <TD headers="t3">Espresso</TD>
  <TD headers="t4">No</TD>
<TR>
  <TD headers="t1">J. Dinnen</TD>
  <TD headers="t2">5</TD>
  <TD headers="t3">Decaf</TD>
  <TD headers="t4">Yes</TD>
</TABLE>
```

A speech synthesizer might render this table as follows:

```
Caption: Cups of coffee consumed by each senator
Summary: This table charts the number of cups
      of coffee consumed by each senator, the type
      of coffee (decaf or regular), and whether
      taken with sugar.
Name: T. Sexton,   Cups: 10,   Type: Espresso,   Sugar: No
Name: J. Dinnen,   Cups: 5,    Type: Decaf,      Sugar: Yes
```

Note how the header "Type of Coffee" is abbreviated to "Type" using the abbr attribute.

Here is the same example substituting the scope attribute for the headers attribute. Note the value "col" for the scope attribute, meaning "all cells in the current column":

```
<TABLE border="1"
      summary="This table charts the number of cups
               of coffee consumed by each senator, the type
               of coffee (decaf or regular), and whether
               taken with sugar.">
<CAPTION>Cups of coffee consumed by each senator</CAPTION>
<TR>
  <TH scope="col">Name</TH>
  <TH scope="col">Cups</TH>
  <TH scope="col" abbr="Type">Type of Coffee</TH>
  <TH scope="col">Sugar?</TH>
<TR>
  <TD>T. Sexton</TD>
  <TD>10</TD>
  <TD>Espresso</TD>
  <TD>No</TD>
<TR>
  <TD>J. Dinnen</TD>
  <TD>5</TD>
  <TD>Decaf</TD>
  <TD>Yes</TD>
</TABLE>
```

Here's a somewhat more complex example illustrating other values for the scope attribute:

```
<TABLE border="1" cellpadding="5" cellspacing="2"
 summary="History courses offered in the community of
          Bath arranged by course name, tutor, summary,
          code, and fee">
 <TR>
  <TH colspan="5" scope="colgroup">Community Courses -- Bath Autumn 1997</TH>
 </TR>
 <TR>
  <TH scope="col" abbr="Name">Course Name</TH>
  <TH scope="col" abbr="Tutor">Course Tutor</TH>
  <TH scope="col">Summary</TH>
  <TH scope="col">Code</TH>
  <TH scope="col">Fee</TH>
 </TR>
 <TR>
  <TD scope="row">After the Civil War</TD>
  <TD>Dr. John Wroughton</TD>
  <TD>
    The course will examine the turbulent years in England
    after 1646. <EM>6 weekly meetings starting Monday 13th
   October.</EM>
  </TD>
  <TD>H27</TD>
```

```
  <TD>&pound;32</TD>
 </TR>
 <TR>
  <TD scope="row">An Introduction to Anglo-Saxon England</TD>
  <TD>Mark Cottle</TD>
  <TD>
    One day course introducing the early medieval
    period reconstruction the Anglo-Saxons and
    their society. <EM>Saturday 18th October.</EM>
  </TD>
  <TD>H28</TD>
  <TD>&pound;18</TD>
 </TR>
 <TR>
  <TD scope="row">The Glory that was Greece</TD>
  <TD>Valerie Lorenz</TD>
  <TD>
   Birthplace of democracy, philosophy, heartland of theater, home of
   argument. The Romans may have done it but the Greeks did it
   first. <EM>Saturday day school 25th October 1997</EM>
  </TD>
  <TD>H30</TD>
  <TD>&pound;18</TD>
 </TR>
</TABLE>
```

A graphical user agent might render this as:

Community Courses -- Bath Autumn 1997				
Course Name	**Course Tutor**	**Summary**	**Code**	**Fee**
After the Civil War	Dr. John Wroughton	The course will examine the turbulent years in England after 1646. *6 weekly meetings starting Monday 13th October.*	H27	£32
An Introduction to Anglo-Saxon England	Mark Cottle	One day course introducing the early medieval period reconstruction the Anglo-Saxons and their society. *Saturday 18th October.*	H28	£18
The Glory that was Greece	Valerie Lorenz	Birthplace of democracy, philosophy, heartland of theater, home of argument. The Romans may have done it but the Greeks did it first. *Saturday day school 25th October 1997*	H30	£18

Note the use of the scope attribute with the "row" value. Although the first cell in each row contains data, not header information, the scope attribute makes the data cell behave like a row header cell. This allows speech synthesizers to provide the relevant course name upon request or to state it immediately before each cell's content.

11.4.2 Categorizing cells

Travel Expense Report

	Meals	Hotels	Transport	subtotals
San Jose				
25-Aug-97	37.74	112.00	45.00	
26-Aug-97	27.28	112.00	45.00	
subtotals	65.02	224.00	90.00	379.02
Seattle				
27-Aug-97	96.25	109.00	36.00	
28-Aug-97	35.00	109.00	36.00	
subtotals	131.25	218.00	72.00	421.25
Totals	196.27	442.00	162.00	**800.27**

Authors categorize a header or data cell by setting the axis attribute for the cell. For instance, in the travel expense table, the cell containing the information "San Jose" could be placed in the "Location" category as follows:

```
<TH id="a6" axis="location">San Jose</TH>
```

Any cell containing information related to "San Jose" should refer to this header cell via either the headers or the scope attribute. Thus, meal expenses for 25-Aug-1997 should be marked up to refer to id attribute (whose value here is "a6") of the "San Jose" header cell:

```
<TD headers="a6">37.74</TD>
```

Each headers attribute provides a list of id references. Authors may thus categorize a given cell in any number of ways (or, along any number of "headers", hence the name).

Below we mark up the travel expense table with category information:

```
<TABLE border="1"
       summary="This table summarizes travel expenses
              incurred during August trips to
              San Jose and Seattle">
<CAPTION>
 Travel Expense Report
</CAPTION>
<TR>
 <TH></TH>
 <TH id="a2" axis="expenses">Meals</TH>
 <TH id="a3" axis="expenses">Hotels</TH>
 <TH id="a4" axis="expenses">Transport</TH>
 <TD>subtotals</TD>
</TR>
<TR>
 <TH id="a6" axis="location">San Jose</TH>
 <TH></TH>
 <TH></TH>
 <TH></TH>
 <TD></TD>
```

```
</TR>
<TR>
  <TD id="a7" axis="date">25-Aug-97</TD>
  <TD headers="a6 a7 a2">37.74</TD>
  <TD headers="a6 a7 a3">112.00</TD>
  <TD headers="a6 a7 a4">45.00</TD>
  <TD></TD>
</TR>
<TR>
  <TD id="a8" axis="date">26-Aug-97</TD>
  <TD headers="a6 a8 a2">27.28</TD>
  <TD headers="a6 a8 a3">112.00</TD>
  <TD headers="a6 a8 a4">45.00</TD>
  <TD></TD>
</TR>
<TR>
  <TD>subtotals</TD>
  <TD>65.02</TD>
  <TD>224.00</TD>
  <TD>90.00</TD>
  <TD>379.02</TD>
</TR>
<TR>
  <TH id="a10" axis="location">Seattle</TH>
  <TH></TH>
  <TH></TH>
  <TH></TH>
  <TD></TD>
</TR>
<TR>
  <TD id="a11" axis="date">27-Aug-97</TD>
  <TD headers="a10 a11 a2">96.25</TD>
  <TD headers="a10 a11 a3">109.00</TD>
  <TD headers="a10 a11 a4">36.00</TD>
  <TD></TD>
</TR>
<TR>
  <TD id="a12" axis="date">28-Aug-97</TD>
  <TD headers="a10 a12 a2">35.00</TD>
  <TD headers="a10 a12 a3">109.00</TD>
  <TD headers="a10 a12 a4">36.00</TD>
  <TD></TD>
</TR>
<TR>
  <TD>subtotals</TD>
  <TD>131.25</TD>
  <TD>218.00</TD>
  <TD>72.00</TD>
  <TD>421.25</TD>
</TR>
<TR>
  <TH>Totals</TH>
  <TD>196.27</TD>
  <TD>442.00</TD>
  <TD>162.00</TD>
  <TD>800.27</TD>
```

```
</TR>
</TABLE>
```

Note that marking up the table this way also allows user agents to avoid confusing the user with unwanted information. For instance, if a speech synthesizer were to speak all of the figures in the "Meals" column of this table in response to the query "What were all my meal expenses?", a user would not be able to distinguish a day's expenses from subtotals or totals. By carefully categorizing cell data, authors allow user agents to make important semantic distinctions when rendering.

However, user agents, particularly speech synthesizers, may want to factor out information common to several cells that are the result of a query. For instance, if the user asks "What did I spend for meals in San Jose?", the user agent would first determine the cells in question (25-Aug-1997: 37.74, 26-Aug-1997:27.28), then render this information. A user agent speaking this information might read it:

```
Location: San Jose. Date: 25-Aug-1997. Expenses, Meals: 37.74
Location: San Jose. Date: 26-Aug-1997. Expenses, Meals: 27.28
```

or, more compactly:

```
San Jose, 25-Aug-1997, Meals: 37.74
San Jose, 26-Aug-1997, Meals: 27.28
```

An even more economical rendering would factor the common information and reorder it:

```
San Jose, Meals, 25-Aug-1997: 37.74
                 26-Aug-1997: 27.28
```

User agents that support this type of rendering should allow user agents a means to customize rendering (e.g., through style sheets).

11.5 Sample table

This sample illustrates grouped rows and columns. The example is adapted from "Developing International Software", by Nadine Kano.

In "ascii art", the following table:

```
<TABLE border="2" frame="hsides" rules="groups"
       summary="Code page support in different versions
                of MS Windows.">
<CAPTION>CODE-PAGE SUPPORT IN MICROSOFT WINDOWS</CAPTION>
<COLGROUP align="center">
<COLGROUP align="left">
<COLGROUP align="center" span="2">
<COLGROUP align="center" span="3">
<THEAD valign="top">
```

```
<TR>
<TH>Code-Page<BR>ID
<TH>Name
<TH>ACP
<TH>OEMCP
<TH>Windows<BR>NT 3.1
<TH>Windows<BR>NT 3.51
<TH>Windows<BR>95
<TBODY>
<TR><TD>1200<TD>Unicode (BMP of ISO/IEC-10646)<TD><TD><TD>X<TD>X<TD>*
<TR><TD>1250<TD>Windows 3.1 Eastern European<TD>X<TD><TD>X<TD>X<TD>X
<TR><TD>1251<TD>Windows 3.1 Cyrillic<TD>X<TD><TD>X<TD>X<TD>X
<TR><TD>1252<TD>Windows 3.1 US (ANSI)<TD>X<TD><TD>X<TD>X<TD>X
<TR><TD>1253<TD>Windows 3.1 Greek<TD>X<TD><TD>X<TD>X<TD>X
<TR><TD>1254<TD>Windows 3.1 Turkish<TD>X<TD><TD>X<TD>X<TD>X
<TR><TD>1255<TD>Hebrew<TD>X<TD><TD><TD><TD>X
<TR><TD>1256<TD>Arabic<TD>X<TD><TD><TD><TD>X
<TR><TD>1257<TD>Baltic<TD>X<TD><TD><TD><TD>X
<TR><TD>1361<TD>Korean (Johab)<TD>X<TD><TD><TD>**<TD>X
<TBODY>
<TR><TD>437<TD>MS-DOS United States<TD><TD>X<TD>X<TD>X<TD>X
<TR><TD>708<TD>Arabic (ASMO 708)<TD><TD>X<TD><TD><TD>X
<TR><TD>709<TD>Arabic (ASMO 449+, BCON V4)<TD><TD>X<TD><TD><TD>X
<TR><TD>710<TD>Arabic (Transparent Arabic)<TD><TD>X<TD><TD><TD>X
<TR><TD>720<TD>Arabic (Transparent ASMO)<TD><TD>X<TD><TD><TD>X
</TABLE>
```

would be rendered something like this:

A graphical user agent might render this as:

CODE-PAGE SUPPORT IN MICROSOFT WINDOWS

Code-Page ID	Name	ACP	OEMCP	Windows NT 3.1	Windows NT 3.51	Windows 95
1200	Unicode (BMP of ISO/IEC-10646)			X	X	*
1250	Windows 3.1 Eastern European	X		X	X	X
1251	Windows 3.1 Cyrillic	X		X	X	X
1252	Windows 3.1 US (ANSI)	X		X	X	X
1253	Windows 3.1 Greek	X		X	X	X
1254	Windows 3.1 Turkish	X		X	X	X
1255	Hebrew	X				X
1256	Arabic	X				X
1257	Baltic	X				X
1361	Korean (Johab)	X			**	X
437	MS-DOS United States		X	X	X	X
708	Arabic (ASMO 708)		X			X
709	Arabic (ASMO 449+, BCON V4)		X			X
710	Arabic (Transparent Arabic)		X			X
720	Arabic (Transparent ASMO)		X			X

This example illustrates how COLGROUP can be used to group columns and set the default column alignment. Similarly, TBODY is used to group rows. The frame and rules attributes tell the user agent which borders and rules to render.