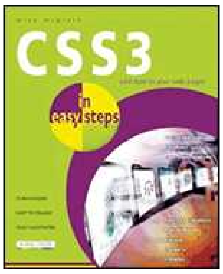


CSS3

by Mike McGrath

Chapter 6: Arranging Data

This chapter demonstrates how to arrange data in tables and lists.

Setting Table Columns

Although web page authors are now discouraged from using HTML tables for page layout (in favor of CSS) tables remain an invaluable format for the presentation of information within the content of a page.

When displaying an HTML table the browser will, by default, automatically create a table layout sized to accommodate its content. This invariably produces a table with columns of varying width where each column width is determined by the widest content of any cell in that column. This process requires the browser to examine the table content in some detail before it can compute the optimum table layout and, especially for large tables, can take some time before the browser is able to draw the table.

Don't Forget The **caption-side** property can suggest where a caption might appear but the actual treatment of captions is browser-specific.

CSS provides an alternative that allows the browser to quickly compute a suitable table layout without examining the content of the entire table – a fixed-layout can be specified to the **table-layout** property of a table element with the **fixed** keyword.

In a **fixed** layout the browser need only consider the **width** value of the table itself and the **width** value of the columns and cells on its first row to determine the table layout like this:

- The overall table width will be its specified **width** value or the sum of its column **width** values – whichever is the greater
- A specified column **width** value sets the width for that column
- When there is no specified column **width** value a specified cell **width** value sets the width for that column
- Any columns that have no specified **width** values, for either column or cell, will be sized equally within the table width

Hot Tip Specify the first column width and a fixed layout rule to create a first column of custom width and other columns of equal width to each other.

Alternatively a style rule can explicitly specify that the default table layout scheme should be used, in which the browser computes the column widths according to their content, by assigning an **auto** value to the **table-layout** property.

Where tables include a caption element the position of the caption can be suggested by specifying keywords of **top** or **bottom** to the table element's CSS **caption-side** property.

1. Create an HTML document containing two tables with similar content

```
<table><caption>Auto Layout</caption>
<tr><td>Text content</td>
<td>Text content wider than 130px</td>
<td>Text content</td></tr></table>

<table class = "fix"><caption>Fixed Layout</caption>
<tr><td>Text content</td>
<td>Text content wider than 130px</td>
<td>Text content</td></tr></table>
```

table.html

2. Save the HTML document then create a linked style sheet containing a rule to specify table width and its features

```
table { width : 500px; border : 3px dashed blue;
caption-side : top; text-align : center; margin : 0 0 20px }
```

table.css

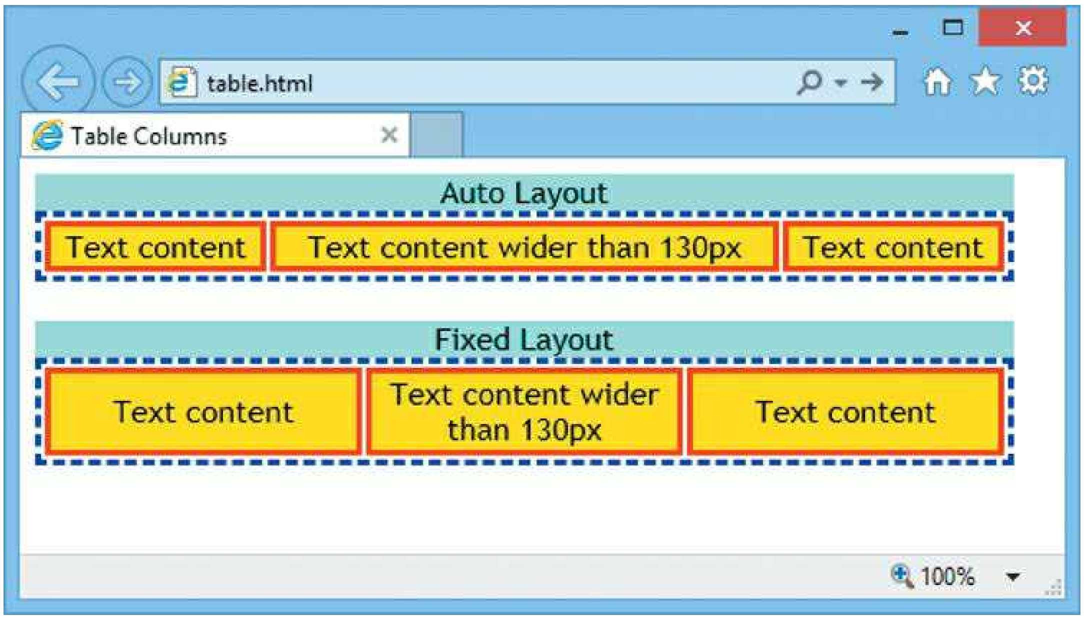
3. Next add style rules to color each table cell and caption

```
td { border : 3px solid red; background : yellow }
caption { background : aqua }
```

4. Now add a style rule to specify a fixed size column scheme for the second table

```
table.fix { table-layout : fixed }
```

5. Save the style sheet alongside the HTML document then open the web page in a browser to see tables drawn with both automatic and fixed layout schemes



Auto Layout		
Text content	Text content wider than 130px	Text content

Fixed Layout		
Text content	Text content wider than 130px	Text content

Spacing Table Cells

The distance between table cell borders can be specified as a unit value to the CSS **border-spacing** property. This easily allows cells to be spread some distance apart throughout a table.

A single specified **border-spacing** value will be applied uniformly to all cell separations – in much the same way as with the HTML **cellspacing** attribute.

CSS provides greater flexibility, however, by allowing two values to be specified to the **border-spacing** property as a space-separated list. The first will be applied to the horizontal spacing, at the left and right of each table cell, and the second will be applied to the vertical spacing at the top and bottom of each cell. This means that different distances can be specified for the horizontal and vertical spacing throughout a table.

Another possibility offered by CSS is the ability to hide table cells that contain no content. These frequently occur due to the grid format of tables, which does not always conveniently match the number of cells required. For example, displaying nine content items in a table of five rows and two columns.

Creating a style rule with the CSS **empty-cells** property specifying a **hide** value will cause the browser to not display the border and background of any cell that contains absolutely no content. Cells that contain any content at all, even if it's simply a **&nbsp;** (nonbreaking space entity) will still be visible.

Conversely a style rule can explicitly ensure that empty cells are displayed by specifying a **show** value to the **empty-cells** property.

Beware Older versions of Internet Explorer, prior to IE8, do not support the **border-spacing** property.

Empty cells that are hidden do continue to have a presence in the table layout inasmuch as their **border-spacing** values are preserved. For example, where the **border-spacing** property is set to **20px**, and the **empty-cells** property specifies a **hide** value, a single empty cell is not displayed but the surrounding cells remain 40 pixels apart – rather than just a distance of 20 pixels that would exist if the hidden cell did not exist.

1. Create an HTML document containing two tables with similar content

```
<table>
<tr><td>1</td><td></td><td>3</td></tr>
</table>

<table class = "space">
<tr><td>1</td><td></td><td>3</td></tr>
</table>
```

hide.html

2. Save the HTML document then create a linked style sheet containing a rule to specify table width and its features

```
table { width : 500px;
                  margin : 20px; border : 3px dotted red }
```

hide.css

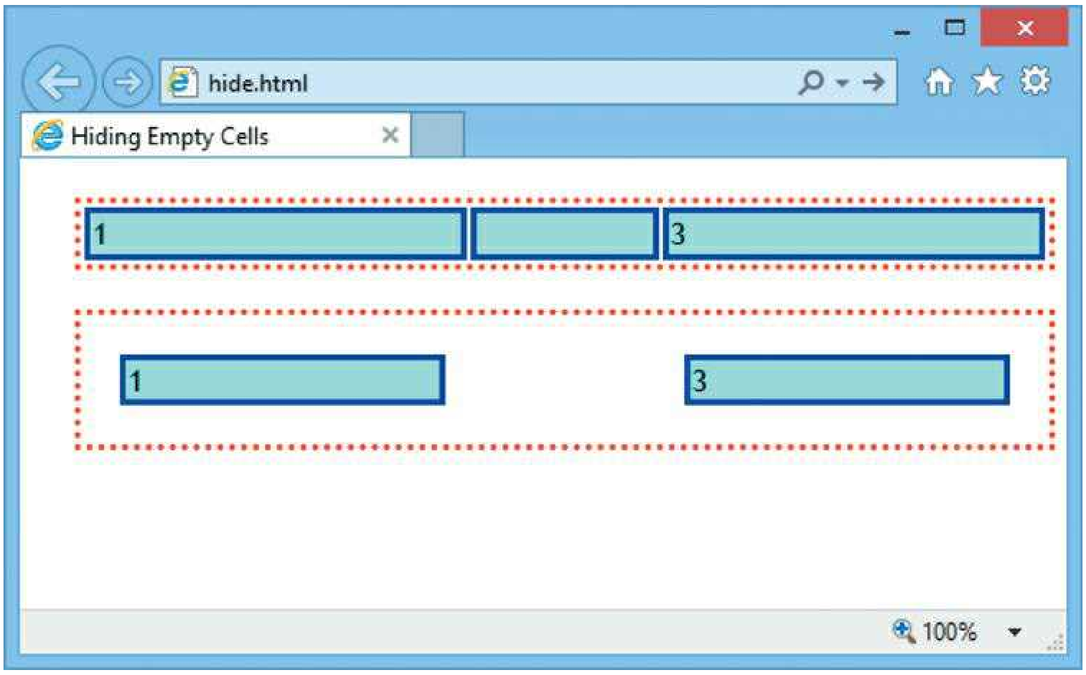
3. Next add a style rule to color each table cell and border

```
td { border : 3px solid blue; background : aqua }
```

4. Now add a style rule to specify the border spacing and hide empty cells in the second table

```
table.space { border-spacing : 20px; empty-cells : hide }
```

5. Save the style sheet alongside the HTML document then open the web page in a browser to see tables drawn with both visible and hidden empty cells



Collapsing Table Borders

The borders of adjacent table borders, and table cell borders, can be made to "collapse" into a single border by specifying the **collapse** keyword to the CSS **border-collapse** property. This requires the browser to perform a series of evaluations, comparing the existing borders, to determine how the collapsed border should appear:

- **Visibility Evaluation:** where one of the borders to be collapsed has a **border-style** value of **hidden**, that value takes precedence – so the collapsed border at that location will be hidden
- **Width Evaluation:** where two visible borders with different **border-width** values are to be collapsed, the highest value takes precedence – so the collapsed border will be the greater width
- **Style Evaluation:** where two visible borders of equal width are to be collapsed, their **border-style** value sets the precedence in the descending status order of **double, solid, dashed, dotted, ridge, outset, groove, inset** – so the collapsed border at that location will be in the style of highest status. For example, a **double** style wins out over a **solid** style
- **Color Evaluation:** where two visible borders of equal width and identical style are to be collapsed, the **border-color** value is determined in the descending status order of **cell, row, row group, column, column group, table** – so that collapsed border will be in the color of highest status. For example, the cell **border-color** wins out over the table **border-color** value

Don't Forget The **separate** keyword can also be specified to the **border-collapse** property – to explicitly prevent collapsing borders.

The effect of collapsing borders where a table **border-width** of **2px** is compared to a cell **border-width** of **5px** means that the collapsed **border-width** will be 5 pixels – the greater width.

In comparing adjacent **border-style** values of **dotted** and **double**, the collapsed **border-style** will be double – the higher status.

Similarly, comparing adjacent **border-style** values of **dotted** and **solid**, the collapsed **border-style** will be solid – the higher status.

1. Create an HTML document containing two tables with similar content

```
<table><tr>
<td class = "twin">1</td>
<td class = "dots">2</td>
<td class = "full">3</td> </tr></table>

<table class = "fold"><tr>
<td class = "twin">1</td>
```

```
<td class = "dots">2</td>
<td class = "full">3</td> </tr></table>
```

collapse.html

2. Save the HTML document then create a linked style sheet containing a rule to specify table width and its features

```
table { width : 500px; height : 60px; margin : 20px }
```

collapse.css

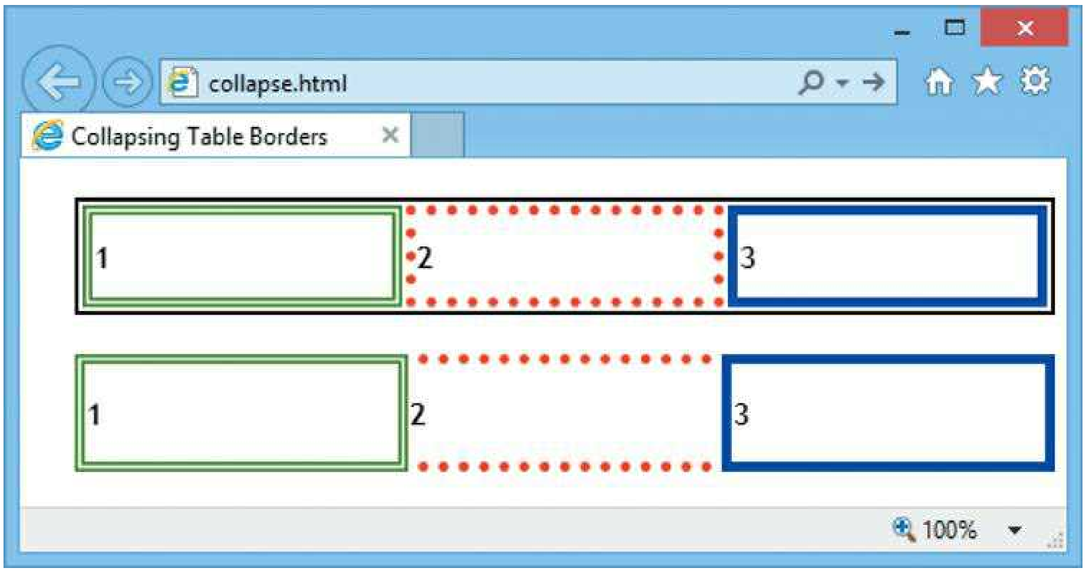
3. Next add style rules to specify the size and color of the table border and each table cell

```
table { border : 2px solid black }
td.twin { border : 5px double green }
td.dots { border : 5px dotted red }
td.full { border : 5px solid blue }
```

4. Now add a style rule to collapse the borders of the second table

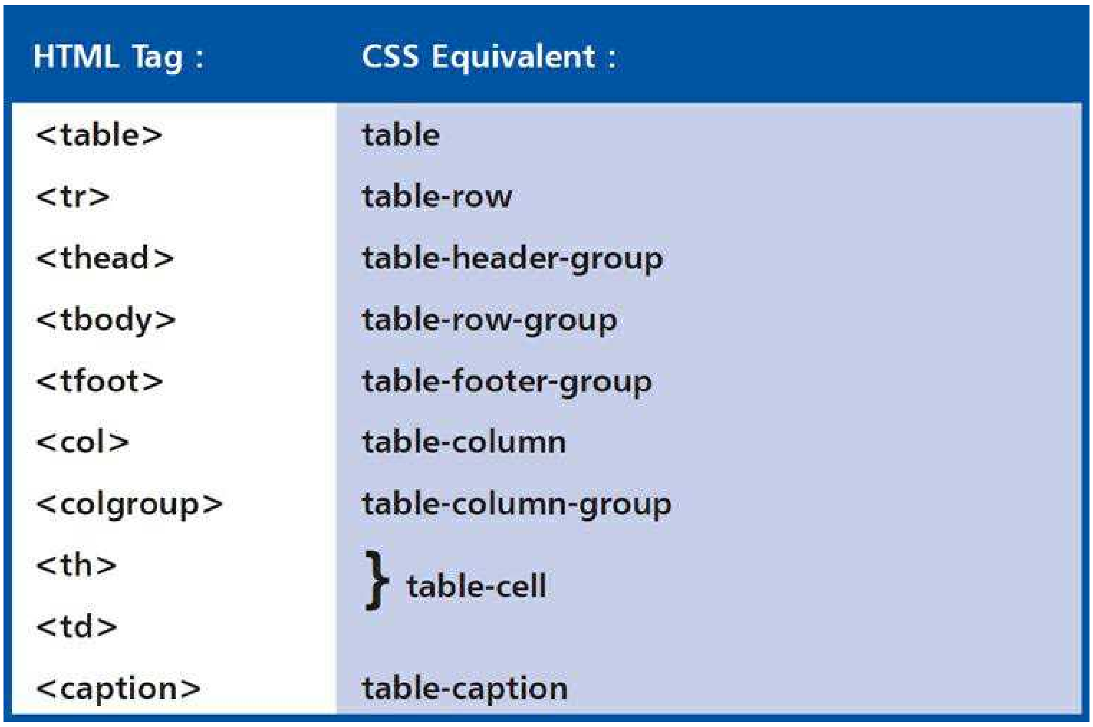
```
table.fold { border-collapse : collapse }
```

5. Save the style sheet alongside the HTML document then open the web page in a browser to see tables drawn with both regular and collapsed borders



Assigning Table Features

The CSS **display** property can accept a range of values to specify that a selected element should be treated as a table component – emulating the default behavior of HTML tags that a browser automatically applies to table components:



HTML Tag :	CSS Equivalent :
<table>	table
<tr>	table-row
<thead>	table-header-group
<tbody>	table-row-group
<tfoot>	table-footer-group
<col>	table-column
<colgroup>	table-column-group
<th>	table-cell
<td>	table-cell
<caption>	table-caption

The CSS values that can be specified to the **display** property are listed in the table above together with the HTML tag they most closely represent. These can be used to specify table features to elements of an XML document so a browser will display their content as if it was an HTML table.

xtable.xml

1. Create an XML document that nominates a CSS style sheet to format its element content

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="xtable.css" type="text/css"?>
<liga><cap>La Liga Top 3</cap>
 <hdrs>
  <lbl>Position</lbl> <lbl>Team</lbl> <lbl>Points</lbl>
 </hdrs>
 <rows>
  <team> <pos>1</pos> <name>Barcelona</name>
  <pts>84</pts> </team>
  <team> <pos>2</pos> <name>Real Madrid</name>
  <pts>80</pts> </team>
  <team> <pos>3</pos> <name>Villareal</name>
  <pts>65</pts> </team>
 </rows>
</liga>
```

2. Save the XML document then create a linked style sheet with rules that assign table characteristics to the XML tags

```
cap { display : table-caption }
liga { display : table }
hdrs { display: table-header-group }
rows { display: table-row-group }
team { display: table-row }
name, pos, pts, lbl { display: table-cell }
```

xtable.css

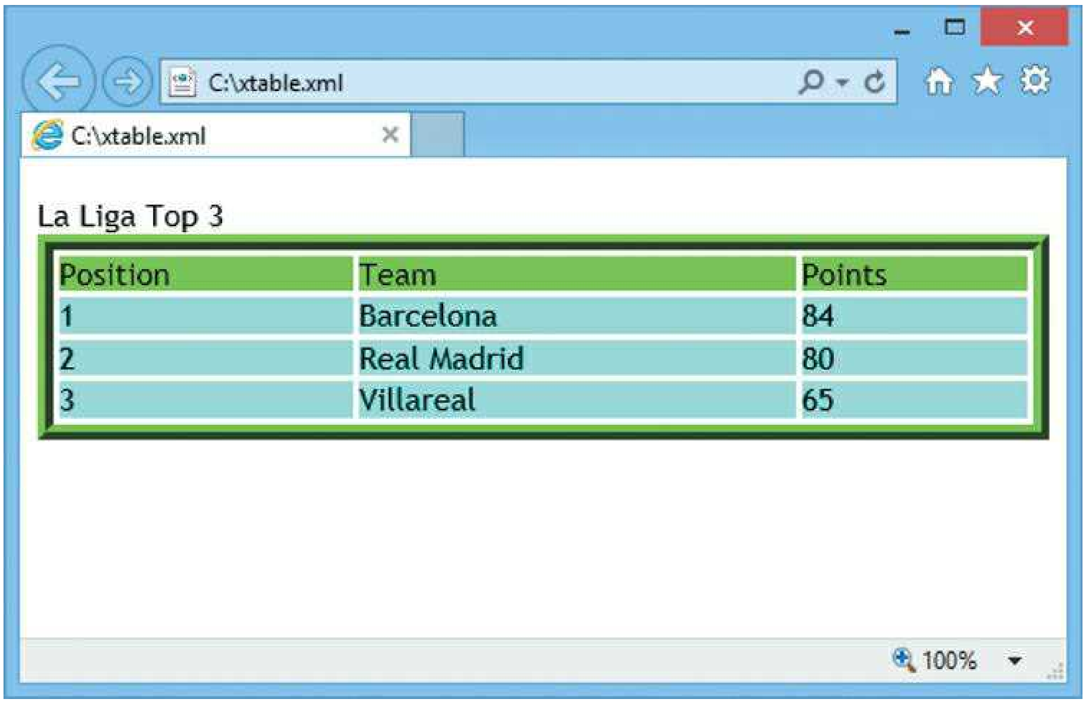
3. Next add a style rule that specifies the table features

```
liga { margin : auto; margin-top : 20px; width : 300px;
            border-spacing : 3px; border : 8px ridge lime }
```

4. Now add style rules to color the headers and row cells

```
hdrs { background : lime }
rows { background: aqua }
```

5. Save the style sheet alongside the XML document then open the document in a browser to see the table



La Liga Top 3

Position	Team	Points
1	Barcelona	84
2	Real Madrid	80
3	Villareal	65

Choosing List Markers

A list "marker" indicates the beginning of an item in a list – typically a bullet in an unordered <ul> list, or an incrementing number in an <ol> ordered list. The browser conducts an item count in each case but usually only uses this to number the items in an ordered list display.

The CSS **list-style-type** property can specify an alternative type of marker for any list – so unordered lists can have numbered markers and ordered lists can have bullet-points if so desired.

Keywords allow the bullet marker type to be specified as **disc, circle**, or **square**, and number marker types as **lower-roman, upper-roman, decimal**, or **decimal-leading-zero.**

Alphabetical marker types can be specified with the **lower-latin, upper-latin**, and **lower-greek** keywords. Additionally the CSS specification provides keywords for other alphabets such as **armenian** and **georgian** – but a suitable font is needed for the marker to be displayed correctly by the web browser.

The **list-style-type** property can also specify a **none** value to explicitly suppress the markers so they will not be displayed, although they do remain in the item count.

Optionally an image may be specified as a marker by stating its path in the parentheses of a **url()** value to the **list-style-image** property.

1. Create an HTML document containing three headings and several ordered lists

```
<h3>Alphabetical list marker types:</h3>
<ol id = "list-0"><li>lower-latin<li>...<li>...</ol>
<ol id = "list-1"><li>upper-latin<li>...<li>...</ol>
<ol id = "list-2"><li>lower-greek<li>...<li>...</ol>
<h3>Bullet list marker types:</h3>
<ol id = "list-3"><li>disc<li>...<li>...</ol>
<ol id = "list-4"><li>circle<li>...<li>...</ol>
<ol id = "list-5"><li>square<li>...<li>...</ol>
<ol id = "list-6"><li>image<li>...<li>...</ol>
<h3>Numerical list marker types:</h3>
<ol id = "list-7"><li>lower-roman<li>...<li>...</ol>
<ol id = "list-8"><li>upper-roman<li>...<li>...</ol>
<ol id = "list-9"><li>decimal<li>...<li>...</ol>
<ol id = "list-10"><li>decimal-leading-zero<li>...<li>...</ol>
```

markers.html

2. Save the HTML document then create a linked style sheet with rules to specify heading and list features

```
h3 { clear : left; margin : 0 }
ol {margin : 0; border : 1px solid black; float : left;
    background : aqua; padding : 0 0 0 10px }
li { margin : 0 0 0 20px; background : yellow }
```

markers.css

3. Next add style rules to specify alphabetical list markers

```
ol#list-0 { list-style-type: lower-latin }
ol#list-1 { list-style-type: upper-latin }
ol#list-2 { list-style-type: lower-greek }
```

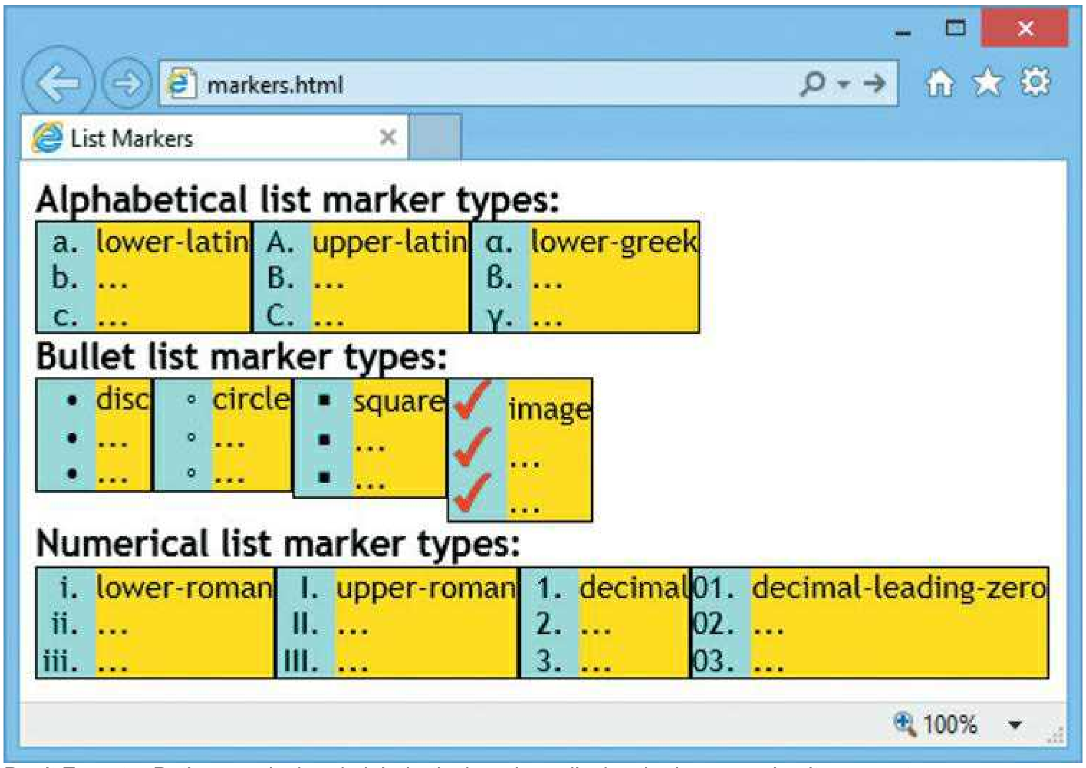
4. Now add style rules to specify bullet list markers

```
ol#list-3 { list-style-type: disc}
ol#list-4 { list-style-type: circle }
ol#list-5 { list-style-type: square }
ol#list-6 { list-style-image : url(tick.png) }
```

5. Finally add style rules to specify numerical list markers

```
ol#list-7 { list-style-type: lower-roman }
ol#list-8 { list-style-type: upper-roman }
ol#list-9 { list-style-type: decimal }
ol#list-10 { list-style-type: decimal-leading-zero }
```

6. Save the style sheet alongside the HTML document then open the web page in a browser to see the list markers



Don't Forget Both numerical and alphabetical markers display the incrementing item count.

Positioning List Markers

Typically, to display a list the browser creates a block-level content box for the entire list and inline content boxes for each list item. Typically, a left margin insets the list item content boxes and each marker appears up against the right edge of this margin area – outside the list item content boxes.

The position of the marker may be explicitly specified to the **list-style-position** property using **inside** or **outside** keywords to determine whether the markers should appear inside the list item content boxes.

Rather than creating separate style rules for the **list-style-type, list-style-image**, and **list-style-position** properties it is simpler to use the CSS shorthand technique that may specify a value for each property as a space-separated list to the **list-style** property. The values may appear in any order and where any value is omitted the default value for that property will be assumed.

Lists of either type may be nested with their marker position and type specified independently.

1. Create an HTML document containing three lists plus one nested list

```
<ol class = "outside-markers">
 <li>List<li>Markers<li>Outside content box
</ol>

<ol class = "inside-markers">
 <li>List<li>Markers<li>Inside content box
</ol>

<ul>
 <li>List<li>Style
 <ol class = "inside-markers">
 <li>List<li>Markers<li>Inside content box
 </ol><li>Shorthand
</ul>
```

list.html

2. Save the HTML document then create a linked style sheet containing rules to show the list boundaries

```
li { background : yellow }
ol, ul { border : 2px solid red }
```

list.css

3. Next add a style rule to specify that some ordered list markers should appear outside the list item content boxes

```
ol.outside-markers { list-style-position : outside }
```

4. Now add a style rule to specify that other ordered list markers should appear inside the list item content boxes

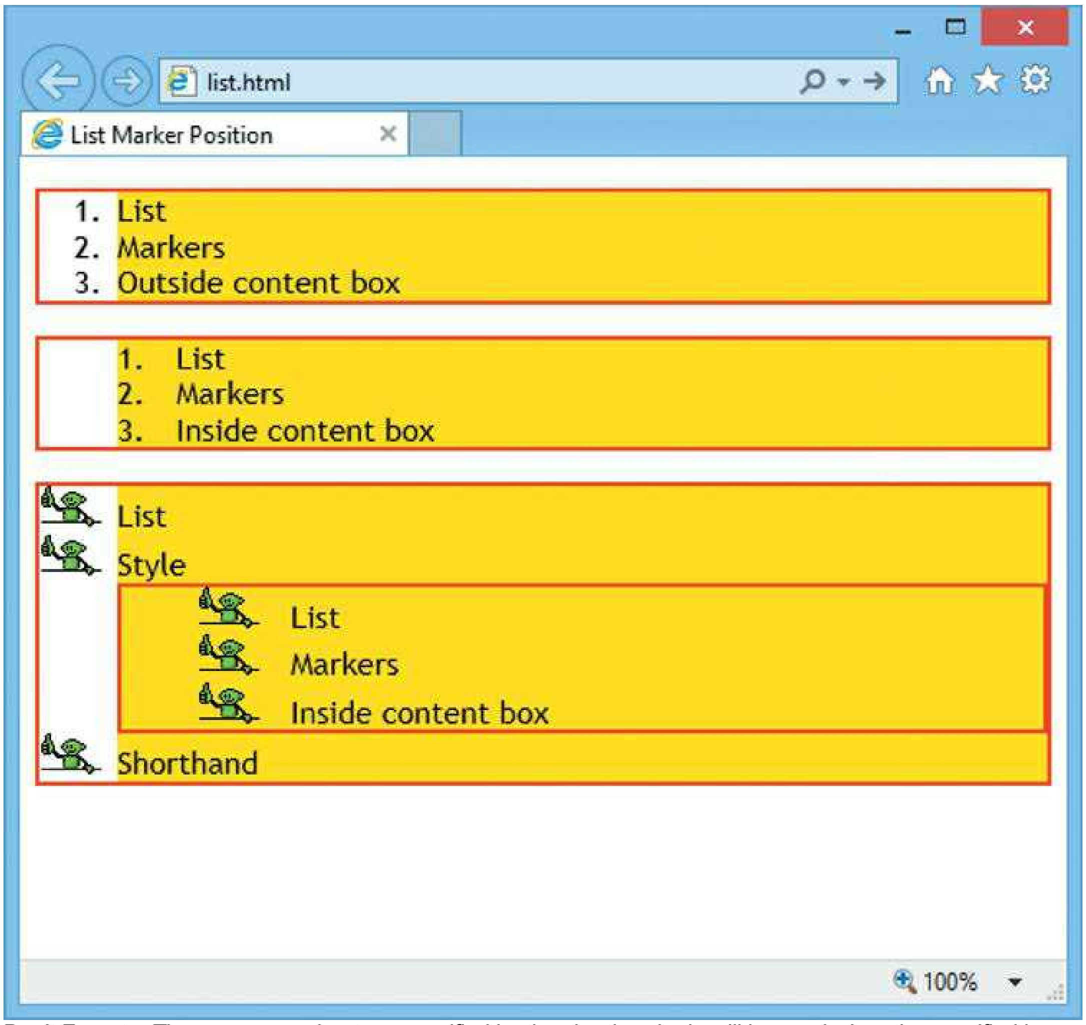
```
ol.inside-markers { list-style-position : inside }
```

Hot Tip Nested lists can specify they should adopt the **list-style** of the containing element using the **inherit** keyword or suppress markers with the **none** keyword.

5. Finally add a shorthand style rule that specifies the position, image, and bullet type for the unordered list

```
ul { list-style : url(lilguy.png) outside square }
```

6. Save the style sheet alongside the HTML document then open the web page in a browser to see the lists



1. List
2. Markers
3. Outside content box

1. List
2. Markers
3. Inside content box

List
Style
List
Markers
Inside content box
Shorthand

Don't Forget The **square** marker type specified by the shorthand rule will be used when the specified image is not available.

Summary

- When the **table-layout** property specifies a **fixed** layout the browser need only assess the **width** value of the table and the **width** of the cells on its first row so it can quickly draw the table
- The **caption-side** property can specify whether the table caption should appear at the **top** or **bottom** of the table
- A **border-spacing** property can specify the distance between table cell borders
- Cells containing absolutely no content can be hidden by specifying a **hide** value to the **empty-cells** property
- Adjacent borders of a table and its cells can be combined into a single border by specifying the **collapse** keyword to the **border-collapse** property
- The **display** property can specify that a selected element should be treated by the browser as a table component using **table, table-row, table-header-group, table-row-group, table-cell, table-footer-group, table-column**, or **table-caption** keywords
- A **list-style-type** property can specify the type of bullet marker to be used for list items with **disc, circle**, or **square** keywords
- The **list-style-type** property can specify that each list item should have numerical markers using the **lower-roman, upper-roman, decimal**, or **decimal-leading-zero** keywords
- Alphabetical list markers can be specified to the **list-style-type** property as **lower-latin, upper-latin**, or **lower-greek**
- An image can be used as a list marker by stating its path in the **url()** value specified to the **list-style-image** property

- Specifying a value of **inside** or **outside** to the **list-style-position** determines whether markers appear inside the content box
- The **list-style** property can specify the individual **list-style-type, list-style-image**, and **list-style-position** properties as CSS shorthand