CHAPTER 15

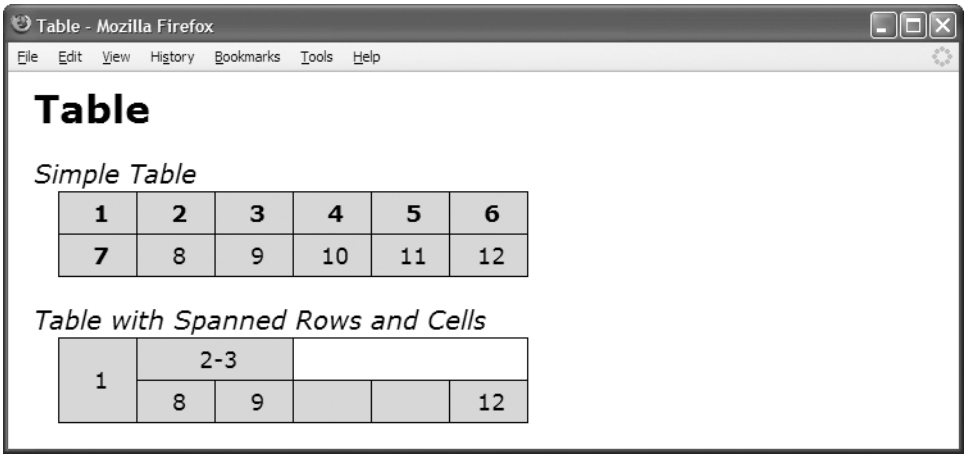
Tables

Tables are one of the most useful and complex structures in HTML. This is the first of two chapters on tables. This chapter explores the HTML structure of tables and how you can style them. The next chapter explores the many ways you can automatically lay out columns in tables. The purpose of tables is to identify and style tabular data.

Chapter Outline

- **Table** shows how to create and style the fundamental structure of a table.
- **Row and Column Groups** shows how to create and style row headers, row footers, row groups, column groups, and columns.
- **Table Selectors** shows how to select cells from columns, rows, and row groups.
- **Separated Borders** shows how to separate table borders from cell borders.
- **Collapsed Borders** shows how to combine table and cell borders.
- **Styled Collapsed Borders** shows how to style collapsed borders.
- **Hidden and Removed Cells** shows how to hide or remove cells.
- **Removed and Hidden Rows and Columns** shows how to remove or hide rows, row groups, and columns of cells.
- **Vertical-aligned Data** shows how to vertically align data to the top, middle, bottom, or baseline of a cell.
- **Striped Tables** shows how to assign alternating backgrounds to rows.
- **Accessible Tables** shows how to create a table that is friendly to nonsighted users.
- **Tabled, Rowed, and Celled** shows how to turn any element into a table, row, or cell.
- **Table Layout** shows how to create the four types of tables: *shrinkwrapped*, *sized, stretched*, and *fixed*.

Table



HTML

```
<h1>Table</h1>

<h2>Simple Table</h2>
<table>
  <tr> <th>1</th> <th>2</th> <th>3</th> <th>4 </th> <th>5 </th> <th>6 </th> </tr>
  <tr> <th>7</th> <td>8</td> <td>9</td> <td>10</td> <td>11</td> <td>12</td> </tr>
</table>

<h2>Table with Spanned Rows and Cells</h2>
<table>
  <tr> <td rowspan="2">1</td> <td colspan="2">2-3</td>                    </tr>
  <tr> <td>8</td> <td>9</td>  <td> </td> <td> </td> <td>12</td> </tr>
</table>
```

CSS

```
table { width:auto; height:1px; table-layout:auto; border-collapse:collapse;
  margin-left:20px; border:1px solid black; }

td, th { width:50px; height:1px; overflow:hidden; visibility:visible;
  border:1px solid black; padding:5px; background:gold;
  text-align:center; vertical-align:middle; text-indent:5px; }
```

Table

Problem You want to create a table to present data in rows and columns.

Solution At its simplest, a table consists of a `<table>` element containing one or more row `<tr>` elements, which contain one or more cells. Cells can be **header cells**, `<th>`, or **data cells**, `<td>`.

Header cells contain text describing the purpose of the columns and rows that they head. You may have zero or more rows of header cells to describe each column. You may have zero or more columns of header cells in each row to describe each row. Header cells and data cells may contain any content including nested tables, blocks, text, and objects. It is a common practice to restrict data cells to tabular data and header cells to text.

You can add the `colspan` and `rowspan` attributes to a cell to have it span one or more columns and/or one or more rows. To prevent missing cells, you need to use the same number of cells in each row or to use `colspan` to span cells across multiple columns. In the second table of the example, the first cell spans two rows, the second cell spans two columns, and the first row is missing three cells.

The major browsers apply box model properties in limited ways to tables, cells, rows, row groups, columns, and column groups. `background` is the only property that applies to all these elements. `margin` applies only to tables. `border` applies only to tables and cells. `padding`, `overflow`, and `vertical-align` apply to cells. `text-indent`, `text-align`, and other text-styling properties apply only to cells but can be inherited from row, row group, and table elements. `width` applies to tables, cells, and columns. `width` is important enough for the next chapter to be devoted to showing how it creates column layouts.

`height` applies to tables, rows, and cells, and specifies the *minimum* height of a table, row, or cell. It is a minimum height because content can always expand the height of a cell, row, or table. Contrast this with block elements where content overflows a fixed-height block instead of expanding it. A percentage-height block assigned to a table is a percentage of the height of the table's container. A percentage-height block is ignored when assigned to rows and cells. In the example, `height:1px` is applied to cells, but is overridden by the height of cell content and padding.

There are several unique table properties including `border-collapse` and `table-layout`. `border-collapse` is discussed in this chapter. `table-layout` is discussed in the next chapter. Additional unique table properties exist, but are implemented inconsistently by the major browsers: `table-layout`, `border-collapse`, `border-spacing`, `caption-side`, and `empty-cells`.

Pattern

HTML

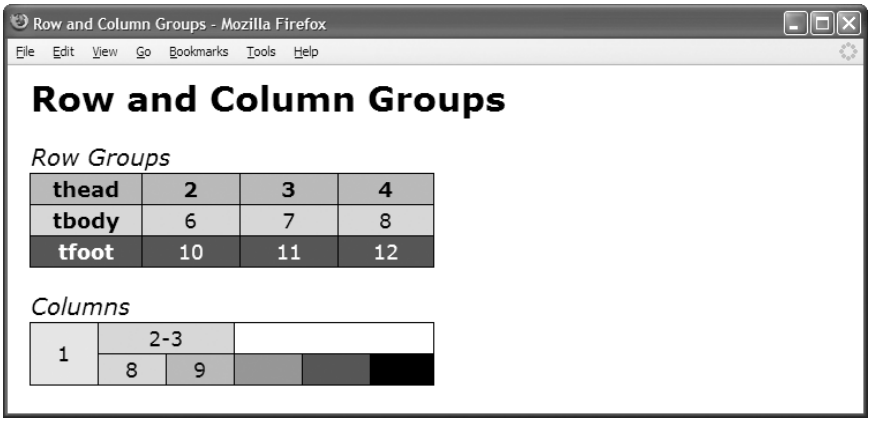
```
<table>
  <tr>
    <td colspan="NUMBER" rowspan="NUMBER"> CONTENT </td>
  </tr>
</table>
```

Location Tables can be used anywhere blocks can be used.

Related to Structural Block Elements, Terminal Block Elements (Chapter 2); Display, Table Box (Chapter 4); Width, Height, Sized, Shrinkwrapped, Stretched (Chapter 5); Margin, Border, Padding (Chapter 6); Atomic (Chapter 7); Offset or Indented Static Table, Aligned and Offset Static Table (Chapter 8); Structural Meaning, Visual Structure, Inlined (Chapter 13); all design patterns in Chapters 15 and 16

See also `www.cssdesignpatterns.com/table`

Row and Column Groups



HTML

```
<h1>Row and Column Groups</h1>

<h2>Row Groups</h2>
<table class="example1">
  <thead> <tr> <th>thead</th> <th>2 </th> <th>3 </th> <th>4 </th> </tr> </thead>
  <tfoot> <tr> <th>tfoot</th> <td>10</td> <td>11</td> <td>12</td> </tr> </tfoot>
  <tbody> <tr> <th>tbody</th> <td>6 </td> <td>7 </td> <td>8 </td> </tr> </tbody>
</table>

<h2>Columns</h2>
<table class="example2">
  <colgroup><col class="col1" /><col class="col2" /><col class="col3" />
            <col class="col4" /><col class="col5" /><col class="col6" /></colgroup>

  <tr> <td rowspan="2">1</td> <td colspan="2">2-3</td>              </tr>
  <tr> <td>8</td> <td>9</td>  <td> </td> <td> </td> <td>12</td> </tr>
</table>
```

CSS

```
table.example1 thead { background:orange; color:black; }
table.example1 tbody { background:gold; color:black; }
table.example1 tfoot { background:firebrick; color:white; }
*.col1 { background:wheat; }
*.col2 { background:gold; }
*.col3 { background:orange; }
*.col4 { background:tomato; }
*.col5 { background:firebrick; }
*.col6 { background:black; color:white; }

/* Nonessential styles are not shown */
```

Row and Column Groups

Problem You want to group together rows and columns to make it easy to style groups of rows and columns.

Solution You can optionally use the following elements to group together rows and columns: `<thead>` (table header row group), `<tfoot>` (table footer row group), `<tbody>` (table body row group), `<colgroup>` (column group), and `<col>` (column).

Row groups are useful for styling groups of rows and cells with `background`, `visibility`, `display:none`, and text properties. You can also use descendant selectors to select rows and cells in row groups. On the other hand, column groups and columns are limited to styling with `background` and `width`.

Row groups may surround any number of rows. You can use data cells or header cells in any row of any row group. You may include any number of `<tbody>` elements in a table, but you should only include at most one `<thead>` and one `<tfoot>`. This is because a browser renders table header and footer groups once per table. Table header groups are placed at the beginning of the table, and the footer groups are placed at the end (even though footer rows are placed before body rows in *HTML code*). When a document is printed, table headers and footers are supposed to be repeated at the top and bottom of each page, but only Firefox 2 does this. Because of this, `<tfoot>` is unsuitable for containing summary data.

Because of inheritance, cells inherit text styles assigned to tables, row groups, and rows. Cells cannot inherit from column groups and columns. `visibility:hidden` and `display:none` apply to tables, rows, row groups, and cells, but not to column groups and columns. `background` applies to all.

Table backgrounds are layered from back to front as follows: table, column groups, columns, row groups, rows, and cells. Since there is no padding between these elements, you can only see the background of an element when its children have a transparent background. For example, to see a row group's background, its rows and cells must have a transparent background.

A table may contain one or more column groups (`<colgroup>`), which may contain one or more columns (`<col>`). Browsers can reliably style column groups and columns with only two properties: `background` and `width`. This is a problem and a severe limitation. In the second table of the example, I select column elements to apply different background colors to each column. Notice how you cannot see the text in cell 12, for it is black on black because browsers apply `background:black` to column elements but not `color:white`.

Pattern

HTML

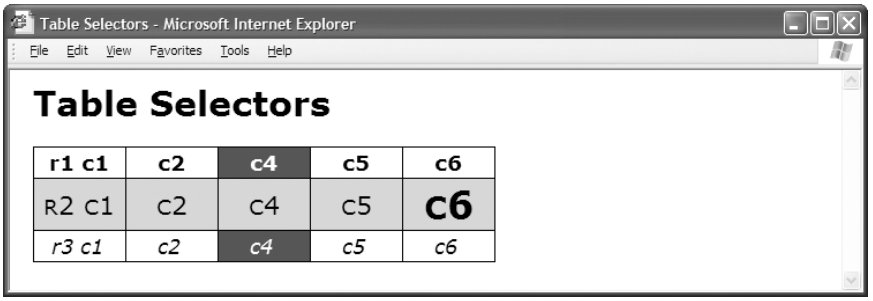
```
<table>
  <colgroup> <col /> </colgroup>
  <thead> <tr> <th> CONTENT </th> </tr> </thead>
  <tfoot> <tr> <th> CONTENT </th> </tr> </tfoot>
  <tbody> <tr> <td> CONTENT </td> </tr> </tbody>
</table>
```

Location This pattern applies to tables.

Related to Table

See also `www.cssdesignpatterns.com/row-groups`
`www.cssdesignpatterns.com/column-groups`

Table Selectors



HTML

```
<h1>Table Selectors</h1>
<table id="t1">
  <thead>
    <tr class="r1"> <td class="c1">r1 c1</td> <td class="c2">c2</td>
                    <td class="c3">c3</td>    <td class="c4">c4</td>
                    <td class="c5">c5</td>    <td class="c6">c6</td> </tr></thead>
  <tfoot>
    <tr class="r3"> <td class="c1">r3 c1</td> <td class="c2">c2</td>
                    <td class="c3">c3</td>    <td class="c4">c4</td>
                    <td class="c5">c5</td>    <td class="c6">c6</td> </tr></tfoot>
  <tbody class="b1">
    <tr class="r2"> <td class="c1">r2 c1</td> <td class="c2">c2</td>
                    <td class="c3">c3</td>    <td class="c4">c4</td>
                    <td class="c5">c5</td>    <td class="c6">c6</td> </tr></tbody>
</table>
```

CSS

```
table,td,th { border:1px solid black; }        /* Selecting all tables and cells */
td,th { background-color:white; }              /* Selecting all cells */

#t1 { border-collapse:collapse; }              /* Selecting table */
#t1 thead td { font-weight:bold; }             /* Selecting cells in head */
#t1 tfoot td { font-style:italic; }            /* Selecting cells in foot */
#t1 tbody td { font-variant:small-caps; }      /* Selecting cells in body */
#t1 *.b1  td { font-size:1.2em; }              /* Selecting cells in body */
#t1 *.c3 { display:none; }                     /* Selecting cells in column */
#t1 *.c4 { background-color:firebrick; color:white; }
#t1 *.r1 { background-color:gold; color:black; }     /* Selecting row-no effect*/
#t1 *.r2 td { background-color:gold; color:black; }  /* Selecting cells in row */
#t1 *.r2 *.c6 { font-size:1.8em; font-weight:bold; } /* Selecting cell */

/* Nonessential styles are not shown */
```

Table Selectors

Problem You want a simple, flexible, and generic way to select a column, a row, or a cell for styling.

Solution You can assign a unique ID to each table, such as `t1`. This allows you to select each table individually. You can label each row with a class that is unique within the table, such as `r1`, `r2`, and so on. You can label each cell with a class that is unique within each row, such as `c1`, `c2`, and so on. Because each table has a unique ID, you can reuse the same class names for rows and columns. By using the table ID with descendant selectors, you can select the table, any row in the table, any cell in any row, and any cell in any column.

You can also enclose rows within `<thead>`, `<tfoot>`, and `<tbody>` elements. If you have multiple `<tbody>` elements, you can also label each one with a unique class, such as `b1`, `b2`, and so on. You can use descendant selectors following the table's ID to select and style the cells in a table header, footer, or one of the row groups defined by `<tbody>`. This makes it easy to style cells in groups of rows.

Selecting a row, table header, table footer, or table body is of little use because you can only style its background and even then you cannot see the background unless cell backgrounds are transparent. In the example, I style all cells with a white background. I also style the first *row* element with a gold background, but you cannot see its gold background because it is covered by the white cell backgrounds. On the other hand, I style *cells* in the second row with a gold background, which you can see because the selector styles cells, not the row. Thus, selecting cells within a row or row group is very useful. All of the following selector design patterns select cells.

Patterns

All Table and Cells Selector
```
table,td,th { STYLES }
```

All Cells Selector
```
td,th { STYLES }
```

Table Selector
```
#tx { STYLES }
```

Column Cells Selector
```
#tx *.cx { STYLES }
```

Row Cells Selector
```
#tx *.rx td { STYLES }   or   #tx *.rx th { STYLES }
```

Cell Selector
```
#tx *.rx *.cx { STYLES }
```

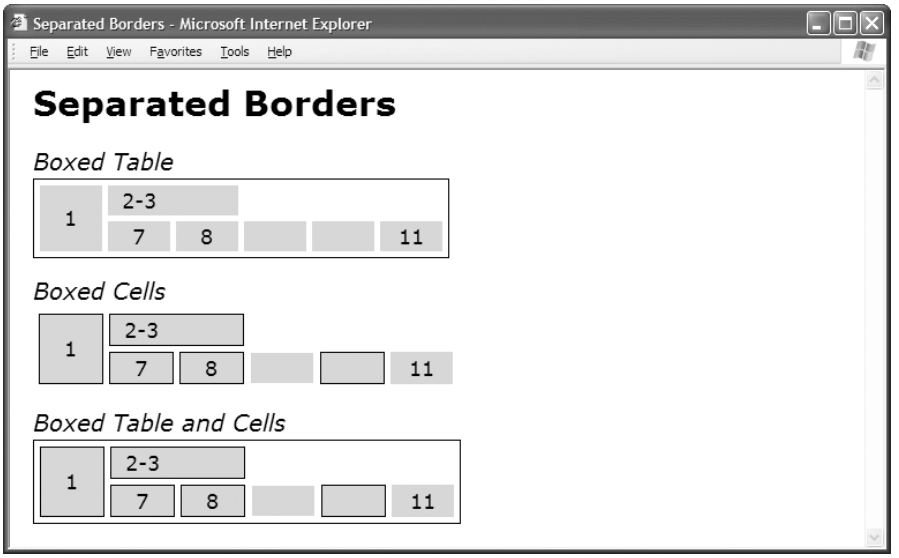
Row Group Selector
```
#tx thead td { STYLES }   or   #tx thead th { STYLES }
```

Location This pattern applies to cells, rows, row groups, and tables.

Related to Table

See also `www.cssdesignpatterns.com/table-selectors`

Separated Borders



HTML

```
<h1>Separated Borders</h1>

<h2>Boxed Table</h2>
<table class="boxed-table" cellspacing="5">
 <tr><td rowspan="2">1</td><td colspan="2">2-3</td></tr>
 <tr><td>7</td><td>8</td><td> </td><td> </td><td class="x">11</td></tr></table>

<h2>Boxed Cells</h2>
<table class="boxed-cells" cellspacing="5">
 <tr><td rowspan="2">1</td><td colspan="2">2-3</td></tr>
 <tr><td>7</td><td>8</td><td> </td><td> </td><td class="x">11</td></tr></table>

<h2>Boxed Table and Cells</h2>
<table class="boxed-table boxed-cells" cellspacing="5">
 <tr><td rowspan="2">1</td><td colspan="2">2-3</td></tr>
 <tr><td>7</td><td>8</td><td> </td><td> </td><td class="x">11</td></tr></table>
```

CSS

```
table { border-collapse:separate; }
*.boxed-table { border:1px solid black; }
*.boxed-cells td { border:1px solid black; }
*.boxed-cells td.x { border:none; }

/* Nonessential styles are not shown */
```

Separated Borders

Problem You want to put independent borders around tables and cells.

Solution You can apply the `border-collapse:separate` property to a table to separate table borders from cell borders. You can use the `border` property to put a border around a table or around a cell. When borders are separate, borders around tables are distinct from borders around cells. You can use the `cellspacing` attribute to control the amount of spacing around cell borders.

Pattern

HTML

```
<table cellspacing="WIDTH">
  <tr> <td> CONTENT </td> </tr>
</table>
```

CSS

```
TABLE_SELECTOR { border-collapse:separate;
  border:WIDTH STYLE COLOR; }

CELL_SELECTOR { border:WIDTH STYLE COLOR; }
```

Location This pattern applies to tables and cells.

Limitations Internet Explorer 7 does not render a border around empty cells. An empty cell does not contain content. Whitespace is not content. In the example, cell 9 has no border because it is empty. In contrast, cell 10 has a border because it contains a nonbreaking space—even though it looks empty. You can prevent this problem by always putting a nonbreaking space in empty cells.

No major browser renders borders or backgrounds for missing cells. Missing cells occur when a row has fewer cells than the table has columns and existing cells do not span enough columns to compensate. In the example, cells 4, 5, and 6 are missing.

Browsers ignore borders applied to rows, columns, column groups, and row groups. This means the only way to put borders around columns or rows is to put them around each cell in the column or row.

Advantages Unlike collapsed borders, separated borders do not have border conflicts between adjacent cells and between the table and its cells.

Disadvantages Separated borders require an HTML attribute, `cellspacing`, to control the distance between cells because Internet Explorer 7 and earlier versions do not implement the `border-spacing` property.

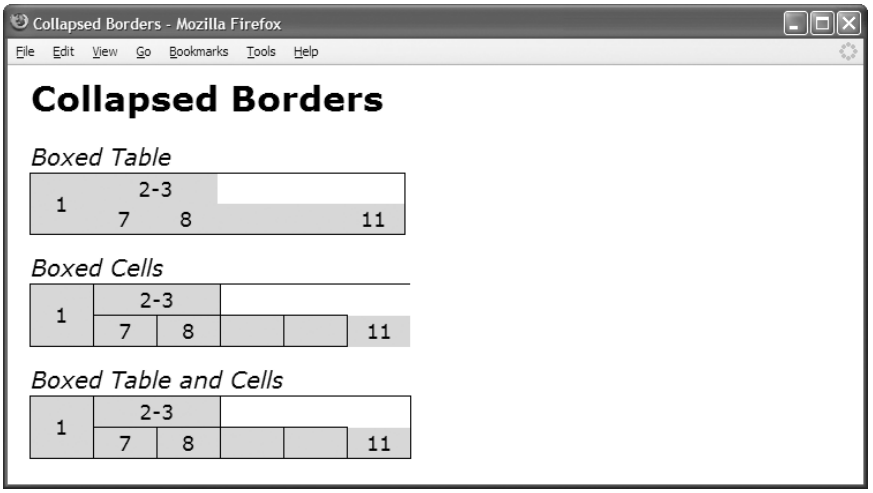
Tips You can use `border:none` to remove a border applied by another rule. Notice in the example how `border:none` removes the border from cell 11.

You can use `border-left`, `border-right`, `border-top`, and `border-bottom` to apply borders independently to each side of a cell or table. In other words, any side of a table or cell can have a different border width, style, and color.

Related to Collapsed Borders; Border (Chapter 6)

See also `www.cssdesignpatterns.com/separated-borders`

Collapsed Borders



HTML

```
<h1>Collapsed Borders</h1>

<h2>Boxed Table</h2>
<table class="boxed-table" cellspacing="0">
 <tr><td rowspan="2">1</td><td colspan="2">2-3</td> </tr>
 <tr><td>7</td><td>8</td><td> </td><td> </td><td class="x">11</td></tr></table>

<h2>Boxed Cells</h2>
<table class="boxed-cells" cellspacing="0">
 <tr><td rowspan="2">1</td><td colspan="2">2-3</td> </tr>
 <tr><td>7</td><td>8</td><td> </td><td> </td><td class="x">11</td></tr></table>

<h2>Boxed Table and Cells</h2>
<table class="boxed-table boxed-cells" cellspacing="0">
 <tr><td rowspan="2">1</td><td colspan="2">2-3</td> </tr>
 <tr><td>7</td><td>8</td><td> </td><td> </td><td class="x">11</td></tr></table>
```

CSS

```
table { border-collapse:collapse; }
*.boxed-table { border:1px solid black; }
*.boxed-cells td { border:1px solid black; }
*.boxed-cells td.x { border:none; }

/* Nonessential styles are not shown */
```

Collapsed Borders

Problem You want to merge table and cell borders.

Solution You can apply the `border-collapse:collapse` property to a table to merge its borders with its cell borders. You can use the `border` property to put borders around a table and its cells. When borders are collapsed, you must omit the `cellspacing` attribute from the table element or set it to `0` to avoid problems in Internet Explorer 7 and earlier versions.

Pattern

HTML

```
<table cellspacing="0">
  <tr> <td> CONTENT </td> </tr>
</table>
```

CSS

```
TABLE_SELECTOR { border-collapse:collapse;
  border:WIDTH STYLE COLOR; }

CELL_SELECTOR { border:WIDTH STYLE COLOR; }
```

Location This pattern applies to tables and cells.

Limitations Internet Explorer 7 (and earlier versions) does not apply borders to rows, columns, column groups, and row groups.

Internet Explorer 7 does not implement `border:hidden`. This is unfortunate because a hidden border has the ability to override and hide a visible merged border. `border:none` cannot override merged borders. Notice in the example how cell 11's border is set to `border:none`, but the left and bottom merged borders are visible because they override `border:none`.

Advantages In contrast to separated borders, all major browsers render collapsed borders around empty cells. Notice in the example how cell 9 is empty and has a border; in the Separated Borders design pattern, it does not have a border.

Disadvantages Unlike separated borders, collapsed borders have border conflicts between adjacent cells and between the table and its cells.

Tips When assigning collapsed borders, it is important to set both table and cell borders. Firefox 2 and Opera 9 in particular have bugs that render extra and incomplete borders when table borders are not set. Notice in the second table in the example how Firefox 2 adds an extra border above the missing cells. This is an error because the second table has no border.

If adjacent borders have different styles, width, or color, the most visible border wins. Wider borders override narrower ones. Border styles override each other in the following order from most prominent to least: `double`, `solid`, `dashed`, `dotted`, `ridge`, `outset`, `groove`, and `inset`. When colors conflict, cell border color overrides table border color. Also, left border color overrides right, and top overrides bottom.

Related to Separated Borders; Border (Chapter 6)

See also `www.cssdesignpatterns.com/collapsed-borders`

Styled Collapsed Borders

HTML

```
<h1>Styled Collapsed Borders</h1>

<table id="t1">
  <tr class="r1"> <td class="c1">1</td> <td class="c2">2</td> </tr>
  <tr class="r2"> <td class="c1">1</td> <td class="c2">2</td> </tr> </table>
```

CSS

```
table { border-collapse:collapse; }             /* Table and cells borders */
table,td,th { border:5px solid red; }

#t1 { border-left:1px solid black; }            /* Left table border */
#t1 *.c1 { border-left:1px solid black; }

#t1 { border-right:2px solid black; }           /* Right table border */
#t1 *.c2 { border-right:2px solid black; }

#t1 *.c1 { border-right:1px dotted black; }     /* Interior column border */
#t1 *.c2 { border-left:1px dotted black; }

#t1 { border-top:1px solid black; }             /* Top table border */
#t1 *.r1 td { border-top:1px solid black; }

#t1 { border-bottom:2px solid black; }          /* Bottom table border */
#t1 *.r2 td { border-bottom:2px solid black; }

#t1 *.r1 td { border-bottom:1px dotted black; } /* Interior row border */
#t1 *.r2 td { border-top:1px dotted black; }

/* Nonessential styles are not shown */
```

Styled Collapsed Borders

Problem You want to assign borders to rows and columns in a table with collapsed borders. The problem is that the table shares borders with its cells, and cells share borders with each other. Thus, each visible border is actually two borders that have been merged, such as the left table border and the left border of each cell in the first column. If you do not style merged borders the same, a browser decides which of the merged borders to display, which may not be the border you want.

Solution You can use the Table Selectors design pattern to mark up the table to make it easy to select columns and rows of cells.

A table with collapsed borders has six types of borders: left table border, interior column border, right table border, top border, interior row border, and bottom border. The design patterns that follow show how to style these six types of merged borders.

Patterns

Left Table Border

```
#t1 { border-left: WIDTH_1 STYLE_1 COLOR_1; }
#t1 *.cx_FIRST { border-left: WIDTH_1 STYLE_1 COLOR_1; }
```

Right Table Border

```
#t1 { border-right: WIDTH_2 STYLE_2 COLOR_2; }
#t1 *.cx_LAST { border-right: WIDTH_2 STYLE_2 COLOR_2; }
```

Interior Column Border

```
#t1 *.cx { border-right: WIDTH_3 STYLE_3 COLOR_3; }
#t1 *.cx+1 { border-left: WIDTH_3 STYLE_3 COLOR_3; }
```

Top Table Border

```
#t1 { border-top: WIDTH_4 STYLE_4 COLOR_4; }
#t1 *.rx_FIRST td { border-top: WIDTH_4 STYLE_4 COLOR_4; }
```

Bottom Table Border

```
#t1 { border-bottom: WIDTH_5 STYLE_5 COLOR_5; }
#t1 *.rx_LAST td { border-bottom: WIDTH_5 STYLE_5 COLOR_5; }
```

Interior Row Border

```
#t1 *.rx td { border-bottom: WIDTH_6 STYLE_6 COLOR_6; }
#t1 *.rx+1 td { border-top: WIDTH_6 STYLE_6 COLOR_6; }
```

Location This pattern applies to cells and tables.

`<colgroup>` and `<col />` cannot be used to style borders.

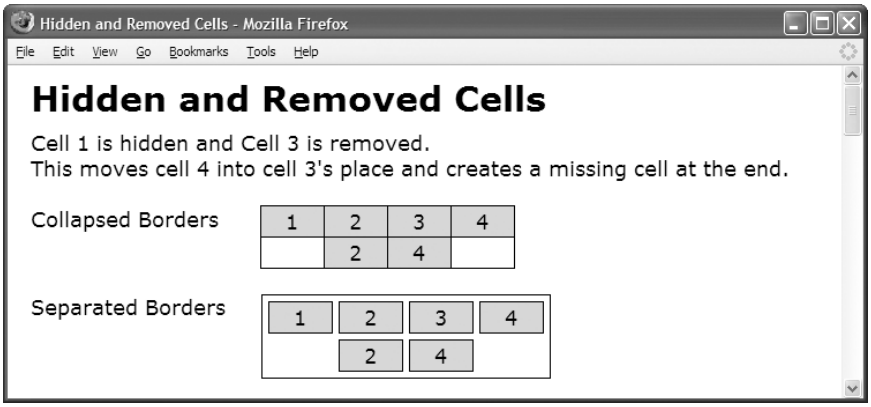
Tip When a table uses separated borders, you do not need this design pattern because separated borders are not shared.

Example In the example, I use the `table,td,td {}` selector to set all table and cell borders to be 5 pixels wide and solid red. If you want all borders to be the same, this selector is all you need. The example overrides these red borders with a variety of smaller black borders assigned to each row and column.

Related to Table Selectors, Collapsed Borders; Border (Chapter 6)

See also `www.cssdesignpatterns.com/styled-collapsed-borders`

Hidden and Removed Cells



HTML

```
<h1>Hidden and Removed Cells</h1>

<h3>Cell 1 is hidden and Cell 3 is removed. <br /> This moves cell 4
  into cell 3's place and creates a missing cell at the end.</h3>

<br /><div>Collapsed Borders</div>
<table class="collapsed" cellspacing="0">
 <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr>
 <tr><td class="h">1</td><td>2</td><td class="x">3</td><td>4</td></tr></table>

<br /><div>Separated Borders</div>
<table class="separated" cellspacing="5">
 <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr>
 <tr><td class="h">1</td><td>2</td><td class="x">3</td><td>4</td></tr></table>

<!-- Many additional examples are not shown -->
```

CSS

```
table, td, th { border:1px solid black; }

*.separated { border-collapse:separate; }
*.collapsed { border-collapse:collapse; }

*.x { display:none; }
*.h { visibility:hidden; }

/* Nonessential styles are not shown */
```

Hidden and Removed Cells

Problem You want to hide or remove one or more cells.

Solution You can use `visibility:hidden` to hide cells. Hidden cells are not rendered, but their location and the space they would have occupied is preserved. This is the most common way to hide a cell because it keeps cells in their proper locations. Notice in the example how the first cell in the second row is hidden without changing the location of the following cells.

When a table has collapsed borders, the borders around hidden cells are still rendered. Thus, when you hide a cell in a table with collapsed borders, its contents are hidden, but its borders are not. Notice in the first table of the example how borders surround the hidden cell in the first column of the second row. On the other hand, borders are not rendered around hidden cells in a table with separate borders. In the second table in the example, there are no borders around the hidden cell in the first column of the second row.

You can use `display:none` to remove cells. Removed cells are not rendered. It is as if they never existed. This means that cells to the right of removed cells slide over to take the place of removed cells! In the example, cell 3 is removed. Notice how cell 4 slides into its place. Because cell 3 is removed, there are fewer cells in the second row than in the first row, which creates a missing cell at the end. Thus, if you do not want cells to be shuffled around, you should hide cells instead of removing them. On the other hand, it is common to remove columns, rows, row groups, and tables because you typically do not want these items to leave behind empty space. This is explored further in the Removed and Hidden Rows and Columns design pattern.

Pattern

Hidden Tables, Rows, and Cells

```
SELECTOR { visibility:hidden; }
```

Removed Tables, Rows, and Cells

```
SELECTOR { display:none; }
```

Location This pattern applies to cells.

Tip When you hide a table with collapsed borders, the table's outer borders are hidden and its contents are hidden, but its internal borders remain visible. To completely hide the table, you can assign `visibility:hidden` to the table and `border:none` to its cells. This is not necessary for tables with separate borders.

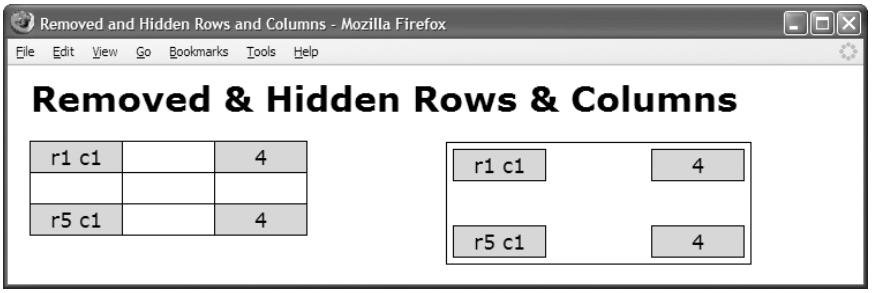
Example The code and the screenshot shown here is a small part of the full example, which includes many more examples of hidden columns, hidden rows, hidden row groups, and hidden tables.

Related to Removed and Hidden Rows and Columns; Display (Chapter 4); Border, Visibility (Chapter 6)

See also

```
www.cssdesignpatterns.com/hidden-cells
www.cssdesignpatterns.com/removed-cells
```

Removed and Hidden Rows and Columns



HTML

```
<h1>Removed &amp; Hidden Rows &amp; Columns</h1>

<table id="t1">
  <tbody class="b1">
    <tr  class="r1"> <td class="c1">r1 c1</td>  <td class="c2">2</td>
                     <td class="c3">r1 c3</td>  <td class="c4">4</td> </tr>

    <tr  class="r2"> <td class="c1">r2 c1</td>  <td class="c2">2</td>
                     <td class="c3">r2 c3</td>  <td class="c4">4</td> </tr></tbody>

  <tbody class="b2">
    <tr  class="r3"> <td class="c1">r3 c1</td>  <td class="c2">2</td>
                     <td class="c3">r3 c3</td>  <td class="c4">4</td> </tr>

    <tr  class="r4"> <td class="c1">r4 c1</td>  <td class="c2">2</td>
                     <td class="c3">r4 c3</td>  <td class="c4">4</td> </tr></tbody>

  <tbody class="b3">
    <tr  class="r5"> <td class="c1">r5 c1</td>  <td class="c2">2</td>
                     <td class="c3">r5 c3</td>  <td class="c4">4</td> </tr></tbody>
</table>

<!-- Second identical table with separated borders is not shown -->
```

CSS

```
#t1 *.c2 { display:none; }       /* Removing column */
#t1 *.c3 { visibility:hidden; }  /* Hiding column */
#t1 *.r2 { visibility:hidden; }  /* Hiding row */
#t1 *.b2 { display:none; }       /* Removing row group */

/* Nonessential styles are not shown */
```

Removed and Hidden Rows and Columns

Problem

You want to remove a column, a row, or a group of rows so that following columns slide over and following rows slide up to take the place of the removed row or column. You want to hide a row or column when you want to leave behind empty space where the row, row group, or column would have been rendered.

Solution

You can use the Table Selectors design pattern to mark up a table to make it easy to select any row or column. You can use `display:none` to remove rows, row groups, and columns. To remove a column, you can assign `display:none` to each cell in the column. To remove a row or a row group, you can assign `display:none` to `<tr>`, `<thead>`, `<tfoot>`, or `<tbody>` elements. Removed elements are not rendered. It is as if they never existed. Columns on the right slide over into the place of removed columns. This causes a shrinkwrapped table to shrink because there is one less column. Rows slide up into the place of removed rows. This causes the height of a shrinkwrapped table to shrink. In the example, the cells in the second column are removed, which causes the third and fourth columns to slide over. Also, the third and fourth rows in the third row group are removed, which causes the fifth row to slide up into their place.

You can use `visibility:hidden` instead of `display:none` to hide rows and columns instead of removing them. This is less common than removing rows and columns because it leaves blank space behind. In the example, I hide the third column and the second row. The space where the rows and columns would have been rendered remains behind.

When columns and rows are *removed*, a browser does not render their borders. On the other hand, when columns and rows are *hidden*, a browser renders borders when borders are collapsed, but not when separated. In the first table of the example, borders are collapsed, and you can see the borders around hidden rows and columns. In the second table, borders are separated, and you cannot see the borders around the hidden rows and columns.

Patterns

Hidden Rows, Row Groups, and Cells

```
SELECTOR { visibility:hidden; }
```

Removed Rows, Row Groups, and Cells

```
SELECTOR { display:none; }
```

Location

This pattern applies to cells, rows, and row groups.

Limitations

You may be tempted to remove or hide columns using the two column elements: `<colgroup>` and `<col />`. Internet Explorer has a proprietary feature that allows this, but other major browsers do not. You may also want to apply `visibility:collapse` to these elements, but this does not work in Internet Explorer 7 or Opera 9. This design pattern is the best way to hide or remove columns.

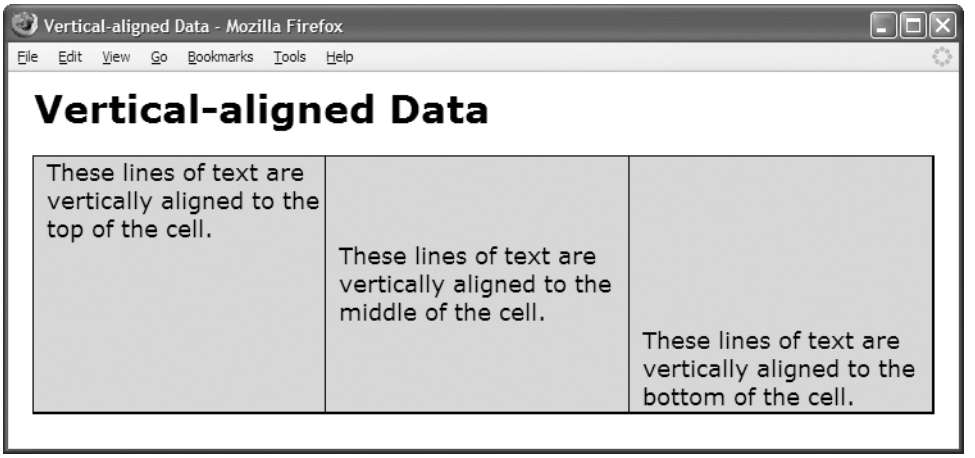
Related to

Hidden and Removed Cells; Display (Chapter 4); Border, Visibility (Chapter 6)

See also

`www.cssdesignpatterns.com/removed-rows-and-columns`
`www.cssdesignpatterns.com/hidden-rows-and-columns`

Vertical-aligned Data



HTML

```
<h1>Vertical-aligned Data</h1>

<table>
  <tr>
    <td class="align-top" >These lines of text are vertically aligned
      to the top of the cell.</td>

    <td class="align-middle">These lines of text are vertically aligned
      to the middle of the cell.</td>

    <td class="align-bottom">These lines of text are vertically aligned
      to the bottom of the cell.</td></tr></table>
```

CSS

```
*.align-top    { height:200px; vertical-align:top; }
*.align-middle { height:200px; vertical-align:middle; }
*.align-bottom { height:200px; vertical-align:bottom; }

/* Nonessential styles are not shown */
```

Vertical-aligned Data

Problem You want to align multiple lines of data as a group to the top, middle, or bottom of a cell.

Solution You can place multiple lines of data in a cell and use `vertical-align` to automatically align it to the top, middle, or bottom of the cell. For this to work, the cell needs to have a height greater than the height of the data; otherwise, there is no space for the data to move up or down within the cell.

`vertical-align` applies to cells and to inline elements. Just as you can use `vertical-align` to offset inline elements from the baseline, you can do the same to the contents of a cell.

There are three `vertical-align` settings that apply in unique ways to cells. These are `top`, `middle`, and `bottom`. `top` is the top of the cell, `middle` is the middle of the cell, and `bottom` is the bottom of the cell. When `top`, `middle`, and `bottom` are applied to inline elements, `top` is the top of the line, `bottom` is the bottom of the line, and `middle` is roughly the middle of the line.

What is unique and useful about `top`, `middle`, and `bottom` when applied to a cell is that they align *the entire contents of a cell* ***including multiple lines of content*** to the top, middle, or bottom of the cell. In contrast, when you apply `vertical-align` to an inline element, it aligns an inline element to another inline element *within a line*. In other words, `vertical-align` positions inline elements in relation to each other within a *single line*, whereas `vertical-align` applied to a cell vertically positions its content within the cell—including *multiple lines* of content.

There is no other mechanism in CSS and HTML that can vertically align multiple lines of *content*. The closest approximations are the absolute design patterns that vertically align an element (not its content) to the top, middle, or bottom of its closest positioned ancestor. These design patterns include Align Top, Align Middle, and Align Bottom. The main problem with absolute design patterns is that they remove elements from the flow. A cell can align its contents without leaving the normal flow.

Patterns

HTML

```
<table><tr><td class="ALIGNMENT"> CONTENT </td></tr></table>
```

CSS

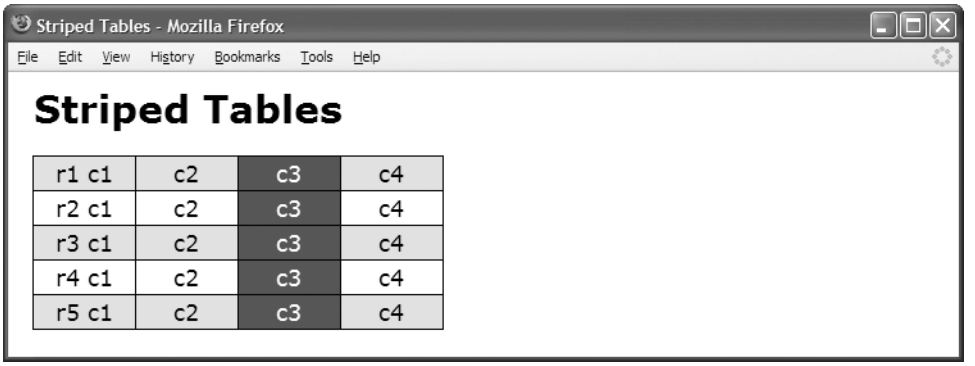
```
*.align-top    { height:+VALUE; vertical-align:top; }
*.align-middle { height:+VALUE; vertical-align:middle; }
*.align-bottom { height:+VALUE; vertical-align:bottom; }
```

Location This design pattern works on any cell.

Related to Vertical-aligned Content, Vertical-offset Content (Chapter 12)

See also `www.cssdesignpatterns.com/vertical-aligned-data`

Striped Tables



HTML

```
<h1>Striped Tables</h1>

<table id="t1">

  <tr class="r1 odd"> <td class="c1">r1 c1</td> <td class="c2">c2</td>
                      <td class="c3">   c3</td> <td class="c4">c4</td> </tr>

  <tr class="r2">     <td class="c1">r2 c1</td> <td class="c2">c2</td>
                      <td class="c3">   c3</td> <td class="c4">c4</td> </tr>

  <tr class="r3 odd"> <td class="c1">r3 c1</td> <td class="c2">c2</td>
                      <td class="c3">   c3</td> <td class="c4">c4</td> </tr>

  <tr class="r4">     <td class="c1">r4 c1</td> <td class="c2">c2</td>
                      <td class="c3">   c3</td> <td class="c4">c4</td> </tr>

  <tr class="r5 odd"> <td class="c1">r5 c1</td> <td class="c2">c2</td>
                      <td class="c3">   c3</td> <td class="c4">c4</td> </tr>

</table>
```

CSS

```
#ts td { background:white; }                    /* Background of all cells */
#t1 *.odd td { background:palegreen; }          /* Alternating Row Background */
#t1 td.c3 { background:darkgreen; color:white; } /* Column Background */

/* Nonessential styles are not shown */
```

Striped Tables

Aliases Greenbar, Zebra Stripes

Problem You want to style alternating rows with different background colors—much like reports printed on greenbar paper.

Solution You can optionally assign a standard background color to all cells or leave them all transparent. You can add a class to odd rows, even rows (or any arbitrary row for that matter), and you can use this class to select and style the `background` of cells in these rows. You can optionally style the backgrounds of cells in columns as well.

Pattern

HTML

```
<table><tr><td class="ALIGNMENT"> CONTENT </td></tr></table>
```

CSS

```
#TABLE_ID *.odd  td { background:COLOR; }
```

or

```
#TABLE_ID *.odd  th { background:COLOR; }
```

Location This pattern applies to cells in a row.

Advantages Styling alternate rows in alternating background colors makes it easier to read extra wide tables. It also promotes the user to read data in rows.

Disadvantages When styling the backgrounds of columns, it takes careful planning and color coordination to make the background of columns blend well with the alternating backgrounds of rows. Furthermore, if you want a column background to override an alternating row background, you need to make sure the column selector has a higher priority in the cascade order than the row selector. In the example, I made the column selector equal priority to the alternating selector by using `#t1 td.c3` instead of `#t1 *.c3`, and I made it a higher priority by placing it after the alternating row selector in the stylesheet.

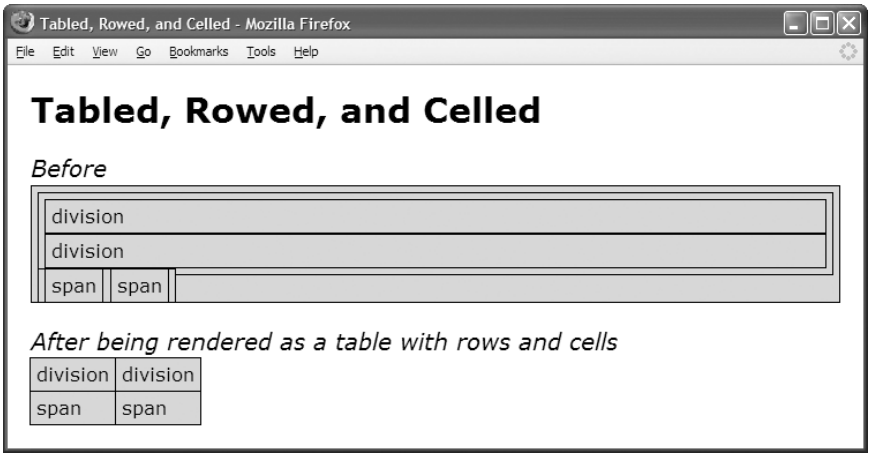
Tips The most important point of this simple design pattern is selecting and styling cells within rows. If you style the background of a row element, you will not see the background unless the background of each cell in the row is transparent. This is because the background of each cell overlays the background of its row. Even when you use separated borders, the spacing between cells does not reveal a row's background, it reveals the table's background. Thus, this design pattern uses the descendant operator to select and style the cells in a row rather than the row itself.

In addition to `background`, you may also want to style `border` and `padding` differently for alternating cells. You may also want to style text properties differently, such as `font-size`, `font-style`, `font-variant`, `font-weight`, `text-decoration`, `text-transform`, `line-height`, `letter-spacing`, and `word-spacing`.

Related to Border, Padding, Background (Chapter 6); Font (Chapter 10); Spacing (Chapter 11)

See also `www.cssdesignpatterns.com/striped-Tables`

Tabled, Rowed, and Celled



HTML

```
<h1>Tabled, Rowed, and Celled</h1>

<h2>Before</h2>
<div>
  <div>
    <div>division</div>
    <div>division</div></div>
  <span>
    <span>span</span>
    <span>span</span></span></div>

<h2>After being rendered as a table with rows and cells</h2>
<div class="tabled">
  <div class="rowed">
    <div class="celled">division</div>
    <div class="celled">division</div></div>
      <span class="rowed">
      <span class="celled">span</span>
        <span class="celled">span</span></span></div>
```

CSS

```
div,span { border:1px solid black; background-color:gold; padding:5px; }

*.tabled { display:table; border-collapse:collapse; }
*.rowed { display:table-row; }
*.celled { display:table-cell; }
```

Tabled, Rowed, and Celled

Problem You want to render ordinary inline and block elements as tables, rows, and cells.

Solution You can use the `display:table`, `display:table-row`, and `display:table-cell` rules to transform elements into tables, rows, and cells.

Typically you nest an element rendered as a cell within an element rendered as a row. In turn, you nest an element rendered as a row within an element that is rendered as a table. It does not matter what type of element is used as long as it is valid XHTML. A table can be created completely out of inline elements, block elements, or a mixture of both.

You can also render an element as a stand-alone cell, and a browser will automatically create a row box and table box. Since tables shrinkwrap by default and since blocks stretch by default, rendering a block as a cell is a good way to shrinkwrap it without having to leave the normal flow.

Patterns

HTML

```
<ELEMENT class="tabled">
  <ELEMENT class="rowed">
    <ELEMENT class="celled"> CONTENT </ELEMENT>
  <ELEMENT class="rowed">
</ELEMENT>
```

CSS

```
*.tabled { display:table; border-collapse:collapse; }
*.rowed { display:table-row; }
*.celled { display:table-cell; }
```

Location This pattern applies to block and inline elements.

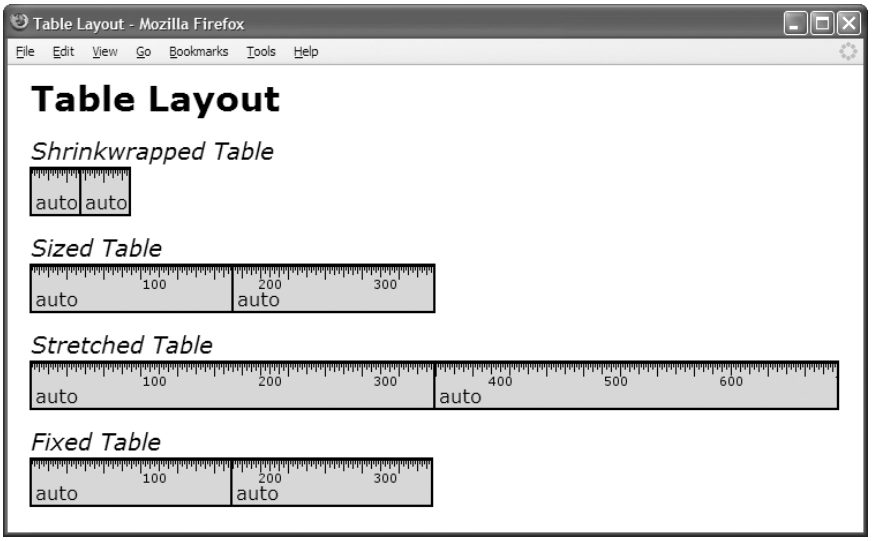
Limitations This pattern does not work in Internet Explorer 7 or earlier versions. This is unfortunate because this is a very useful design pattern. If Internet Explorer supported this part of the CSS standard, you could take advantage of all the unique features offered only by tables. For example, an element displayed as a table automatically shrinkwraps instead of stretches—without leaving the normal flow. This is very useful when you want to create shrinkwrapped buttons, menus, boxes around images, and so on. Displaying an element as a table also allows you to lay out its child elements using the many powerful and automatic layouts presented in Chapter 16. In short, you can take nontabular elements and lay them out in rows and columns for pure presentational pleasure without guilt.

Example In the example, I transform four divisions and three spans into a table with two rows and two columns. Notice how block elements and inline elements can be combined to create a table.

Related to Table; Display, Table Box (Chapter 4); Blocked (Chapter 11); Inlined (Chapter 13)

See also `www.cssdesignpatterns.com/tabled-rowed-celled`

Table Layout



HTML

```
<h1>Table Layout</h1>

<h2>Shrinkwrapped Table</h2>
<table class="auto-layout shrinkwrapped">
 <tr><td>auto</td><td>auto</td></tr></table>

<h2>Sized Table</h2>
<table class="auto-layout sized"> <tr><td>auto</td><td>auto</td></tr></table>

<h2>Stretched Table</h2>
<table class="auto-layout stretched"> <tr><td>auto</td><td>auto</td></tr></table>

<h2>Fixed Table</h2>
<table class="fixed-layout sized"> <tr><td>auto</td><td>auto</td></tr></table>
```

CSS

```
*.auto-layout { table-layout:auto; }
*.fixed-layout { table-layout:fixed; }
*.shrinkwrapped { width:auto; }
*.sized { width:350px; }
*.stretched { width:100%; }

/* Nonessential styles are not shown */
```

Table Layout

Problem You want to create shrinkwrapped, sized, stretched, or fixed tables.

Solution There are four types of tables: *shrinkwrapped*, *sized, stretched*, and *fixed*. Each has unique capabilities for laying out columns. These layouts are explored in detail in the next chapter.

A *shrinkwrapped table* shrinks to the width of its columns and will not expand beyond the width of its container. A *sized or stretched table* can lay out its columns in proportion to the table's width, and can expand beyond the width of its container. A *fixed table* is a variation of a sized or stretched table, except it ignores the width of its content when laying out columns. This greatly speeds the rendering and prevents content from expanding a column's width.

The following two properties assigned to a table determine the type of table: `table-layout` and `width`.

There are two values for `table-layout`: `auto` and `fixed`. The default value is `auto`. An *auto-layout table* lays out columns based on the minimum and maximum widths of cell contents and on the `width` assigned to its cells. A *fixed-layout table* ignores content and lays out columns based only on the `width` assigned to the cells in its first row.

The type of width assigned to the table determines whether a table is shrinkwrapped, sized, or stretched. There are three types of width: auto, fixed, and percentage. An auto width is created using `width:auto`. A fixed width is created using `width:VALUE`, such as `width:100px`. A percentage width is created using `width:PERCENT%`, such as `width:100%`.

A shrinkwrapped table is auto layout and auto width. A stretched table is auto layout and has a percentage width of 100%. A sized table is auto layout and fixed width, or has a percentage width other than 100%. A fixed table is fixed layout and has a fixed width or percentage width.

Patterns

Shrinkwrapped Table

```
TABLE_SELECTOR { table-layout:auto; width:auto; }
```

Sized Table

```
TABLE_SELECTOR { table-layout:auto; width:VALUE_OR_PERCENT; }
```

Stretched Table

```
TABLE_SELECTOR { table-layout:auto; width:100%; }
```

Fixed Table

```
TABLE_SELECTOR { table-layout:fixed; width:VALUE_OR_PERCENT; }
```

Location This pattern applies to table elements.

Tip A good way to set the width of columns is to assign `width` to each cell in the first row of the table. This works in fixed-layout and auto-layout tables, and it does not require `<colgroup>` and `<col>` elements.

Related to Table; Sized, Shrinkwrapped, Stretched (Chapter 5); Offset or Indented Static Table, Aligned and Offset Static Table (Chapter 8); all design patterns in Chapter 16

See also `www.cssdesignpatterns.com/table-layout`