

## **Table.pm – EASIER CREATION OF HTML TABLES**

**Dean Calahan\***

### **ABSTRACT**

Mars Society Members are in good company with activists worldwide, maintaining hundreds or even thousands of Web pages. Although many helpful Web authoring tools exist, some authors code simpler pages directly in HTML. One common HTML idiom is using the HTML <table> tag to position text and graphics. Although many Web authoring tools use this idiom, it is probably fair to say that hand-editing HTML composed of complex tables is so tedious and fraught with error, that hardly anybody does it. Authoring tools are simply the most convenient way to obtain the desired formatting. Table.pm, a Perl (see [www.perl.org](http://www.perl.org)) module available from [http://www.baloney.com/dean/table\\_pm/](http://www.baloney.com/dean/table_pm/), is not quite an authoring tool, but affords the HTML hand-coder the ability to more easily produce complex tables, and thus more sophisticated Web pages.

### **INTRODUCTION**

Table.pm converts an easily derived text string defining a table, to a piece of HTML code that embodies that table, containing replaceable elements that make it easy for the author to insert content. Although easy to create, the finished table is no more easily hand-edited than any other HTML table, so to “modify” pages after they are first created, simply recreate the page with new content (rebuilding or reusing the Table.pm generated HTML). In fact, although this process can be done manually with little effort, Table.pm is even more useful in an automated HTML generating environment.

### **1: CREATING COMPLEX TABLES**

There are four steps for creating the desired HTML table. First, write down a string representation of the table. Second, using Table.pm, generate the HTML table code with replaceable elements. Third, replace those elements with the desired content. Forth, incorporate the table into the HTML of the page being designed. The following sections describe how to do steps one, two, and three (the intended audience for Table.pm is unlikely to need instructions for step four).

#### **1.1: Defining the table string**

The string representation of a table uses a sequence of numerals separated by commas to define a row, and a sequence of rows separated by semicolons to define the table. In the examples, consecutive numbers are used, but as long as the numerals are unique for each cell or

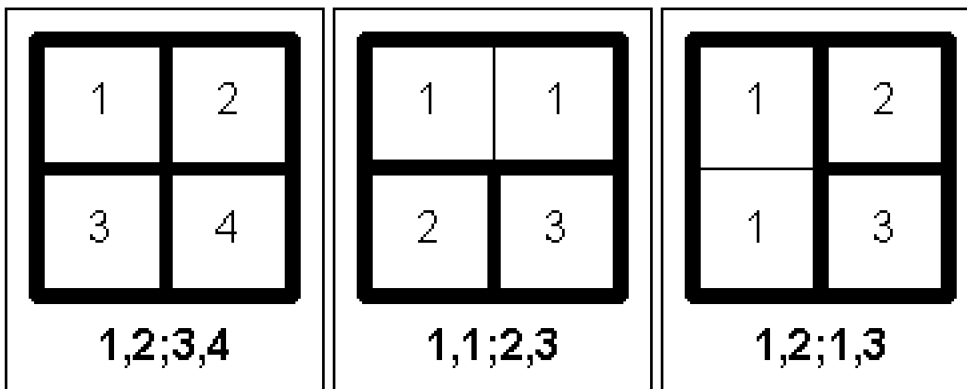
---

\* email: [dean@baloney.com](mailto:dean@baloney.com)

span (randomly generated characters might even be used, so long as each is unique), Table.pm will generate correct HTML for the defined table.

Four steps are used to create a string that matches the geometry of an HTML table. First, sketch the desired table geometry using firm bold strokes. Second, if there are colspans or rowspans, lightly sketch in the cells hidden by the span. Third, give each individual cell (including the each cell that constitutes a span) a numeral, using the same numeral for each cell of a particular span, but a unique numeral for each individual cell or span.

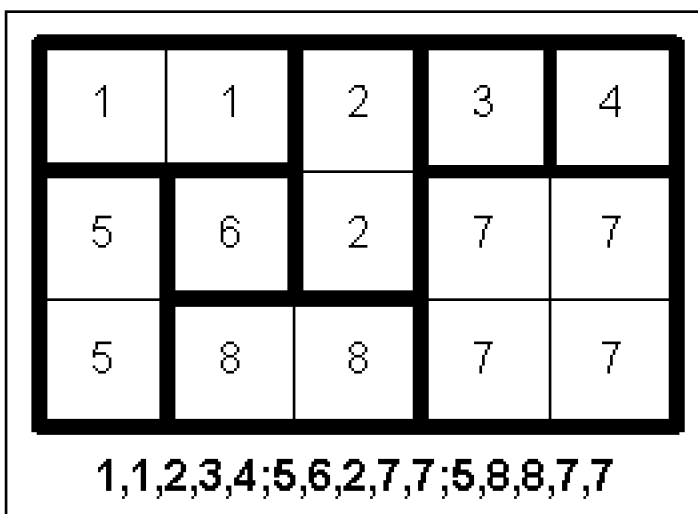
This is perhaps best understood through a few examples. Figure 1 shows how the string “1,2;3,4” is obtained for a simple 4 by 4 table with no spans. Figures 2 and 3 show how colspan and rowspan attributes are interpreted. Figure 4 shows how a much more complex table, using both colspans and rowspans, easily produces a string representation.



*Figure 1*

*Figure 2*

*Figure 3*



*Figure 4*

## **1.2: Generating HTML with table.pl**

### **1.2.1: The command line**

Table.pl (the Perl script distributed with the Table.pm module) is designed to function on the command line. The Perl savvy among the audience will find it trivial to make use of Table.pm in their own scripts by a quick glance at table.pl, whereas the rest of the audience may find it easier to create batch files or shell scripts that call table.pl.

Be aware that, as arguments to table.pl are separated by spaces, care must be taken to insure that all argument strings be free of spaces.

### **1.2.2: Example**

To generate a 2x2 table with no spans, use Perl table.pl 1,2;3,4 (see Figure 1). Used this way, table.pl will print the HTML for the table directly to the screen, Perl table.pl 1,2;3,4 > table.txt will, on many platforms, create a file named table.txt containing the HTML for the table. Other options for directing the output stream exist, including piping it to another process. It is up to the user to determine which option works best in their environment.

### **1.2.3: Attributes**

In addition to the table string, which is required, further arguments can be supplied to Table.pm that may specify any <table> tag attribute, as well as any attributes for the <td> tags within the table.

Arguments for the table attributes are specified on the command line in a colon separated format. For example, to specify a border in the example, use Perl table.pl 1,2;3,4 border:1.

Arguments to specify format of <td> tags are specified on the command line in a colon-comma-separated format. The plural of the desired attribute is followed by a colon, followed by the value for each <td> separated by a comma. For example, to specify the align attribute of each cell in the example, the following command would work: Perl table.pl 1,2;3,4 aligns:left, center, right, center.

The single exceptions to the <td> formatting syntax is the specification of cell widths. This is still expressed in a colon-comma separated format, but only the widths of each column are specified, not the width and height of each cell. To use the example, Perl table.pl 1,2;3,4 widths:200,300. Note: for the sake of completeness, it might be desirable that cell heights be specifiable in a row-by-row manner exactly similar to the specification of column widths. At press time, Table.pm does not support this function, and in practice the author has not found it necessary. This does not mean that cell heights cannot be specified, just that they must be specified in a cell-by cell manner rather than row-by-row. However, future revisions of the module may include this and other functions that may seem appropriate.

### 1.3: Inserting content into the table

Each cell or span is represented in Table.pm generated HTML as a table tag (optionally with attributes):“<td>%n%</td>”, where n is an integer surrounded by percent sign characters, starting at zero and ascending incrementally, numbering left to right, top to bottom. Two basic ways exist for inserting the content – manually open the created file in an editing program, search for each integer pattern, and type or paste in the content, or store each piece of content in a separate file and use an automated process to insert the content. Again, it is up to the user to decide which approach is best.

## CONCLUSION

For the HTML-savvy Web author comfortable with shell scripts, batch files, or other automation tools, creating pages with sophisticated layout does not require use of a sophisticated Web authoring tool. Table.pm defines an easy-to-use notation that can be combined with manual or elementary automation techniques to generate tables at any level of complexity.

## APPENDIX

HTML code generated by Table.pm for each of the example tables in figures 1-4, as well as a variant of figure 4 with <table> and <td> attributes (including widths) follows. It should be noted in 5.5 that apparent redundancies exist in the width attributes – not only does each cell have a width specified, there is also an extra row of blank cells with widths specified. The state of web browser technology is still such that incompatibilities exist between versions of browsers available. The author has discovered circumstances in which specifying only the cell width in spanned cells can produce incorrect output in one browser, and other circumstances in which specifying only the column widths in an extra row of blank cells can produce incorrect output in a different browser. The only way to resolve this inconsistency is to redundantly specify widths. The author would cynically welcome evidence that this solution also produces output inconsistent between browser versions.

Additionally, be aware that web browsers can be presented with inconsistent HTML. For example, if a table cell has both width and height specified, yet the content for that cell exceeds either of those limits, it is certain that the output will not be as expected, being wider or higher than specified. It is often easiest to specify only those widths or heights that are most important, as inspecting HTML to detect overspecified, or inconsistently specified, values is often tedious and confusing.

### 5.1: Figure 1

```
<table>
<tr>
<td>%0%</td>
<td>%1%</td>
</tr>
<tr>
```

```
<td>%2%</td>
<td>%3%</td>
</tr>
</table>
```

## 5.2: Figure 2

```
<table>
<tr>
<td colspan="2">%0%</td>
</tr>
<tr>
<td>%1%</td>
<td>%2%</td>
</tr>
</table>
```

## 5.3: Figure 3

```
<table>
<tr>
<td rowspan="2">%0%</td>
<td>%1%</td>
</tr>
<tr>
<td>%2%</td>
</tr>
</table>
```

## 5.4: Figure 4

```
<table>
<tr>
<td colspan="2">%0%</td>
<td rowspan="2">%1%</td>
<td>%2%</td>
<td>%3%</td>
</tr>
<tr>
<td rowspan="2">%4%</td>
<td>%5%</td>
<td colspan="2" rowspan="2">%6%</td>
</tr>
<tr>
<td colspan="2">%7%</td>
</tr>
</table>
```

## 5.5: Figure 4'

```
<table border="1">
<tr>
<td width="40" align="left" height="50" colspan="2">%0%</td>
<td width="20" align="right" height="50" rowspan="2">%1%</td>
<td width="40" align="center" height="*">%2%</td>
<td width="40" align="left" height="50">%3%</td>
</tr>
<tr>
<td width="20" align="right" height="50" rowspan="2">%4%</td>
<td width="20" align="center" height="*">%5%</td>
<td width="80" align="left" height="100" colspan="2"
rowspan="2">%6%</td>
</tr>
<tr>
<td width="40" align="right" height="*" colspan="2">%7%</td>
</tr>
<tr>
<td width="20" height="0"></td>
<td width="20" height="0"></td>
<td width="20" height="0"></td>
<td width="40" height="0"></td>
<td width="40" height="0"></td>
</tr>
</table>
```