# Coding Tables

# Basic Tables

# Table element

When creating a table you want to use the **<table>** element to initialize. It has an opening and closing tag and it will wrap all the **table rows** and **table cells** inside of it.

Tables are made up of rows of information that go across the page. A <tr> element is used to create a table row.

There is no element for the table column as the columns depend on how many table cells are inside each row.

# Continued

A **<td>** element is used to create a table cell. So basically the number of <td> elements you will add inside the **<tr>** element is the exact same number of columns you will get inside your table row.

**<table>:** The table element represents data in a series of rows and columns. Tables should only be used for displaying tabular data, and never for page layout.

**<tr>:** The table row element defines a row of cells in a table. Table rows can be filled with table cells and table header cells.

**<td>:** The table cell element contains data and represents a single table cell. Each table cell should be inside a table row.

# Table Header & Table Body Element

The **table header** element on each of these three columns to tell the browser, search engine crawlers, and screen readers that these are actually headers and not just regular data.

**<thead>**

**<tbody>**

The table body element defines one or more rows that makeup the primary content of ( or “body”) of a table/

# Table foot element

In general, a t**able footer** element should contain a summary of the table. This might be some final cells containing sums, totals and averages for each column. It might also contain some meta information like copyright information or the source of data within a table.

You would think that the **table footer** would go at the **bottom** of the table. However, it actually should go right after the **table head** element and just before the table body element.

**<tfoot>**

***but HTML5 has provided leeway here. These elements may now occur in any order so long as they are never parent elements of one another.***

# Caption Element

This element is basically a title for the table, and it should come immediately after the opening table tag. This is nice to add because it quickly summarizes what a table might contain.

A caption will help users identify what the table pertains to and what data they can expect to find within it

**<caption>**

# Combining Multiple cells

The **colspan** attribute is used to span a single cell across multiple columns within a table, while the rowspan attribute is used to span a single cell across multiple rows. Each attribute accepts an integer value that indicates the number of cells to span across, with 1 being the default value.

```
<table>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
  <tr>
    <td colspan="2">Sum: $180</td>
  </tr>
</table>

</body>
</html>
```

| Month | Savings |
| --- | --- |
| January | $100 |
| February | $80 |
| Sum: $180 | |

# Table Border, Table Collapse & Border Spacing

To specify table borders in CSS, use the **border** property.

The **border-collapse** property sets whether the table borders should be collapsed into a single border.

The **border-spacing** property can determine how much space, if any, appears between the borders.

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

}
```

| Firstname | Lastname |
|---|---|
| Peter | Griffin |
| Lois | Griffin |

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

}
```

| Firstname | Lastname |
|---|---|
| Peter | Griffin |
| Lois | Griffin |

```
table, th, td {
  border: 1px solid black;
  border-collapse: seperate;
  border-spacing: 4px;
 }
```

| Firstname | Lastname |
|---|---|
| Peter | Griffin |
| Lois | Griffin |

# Adding Borders to Rows

Within a table, borders cannot be applied to **<tr>** elements or table structural elements, so when we want to put a border between rows some thought is required.

Example:

making sure the table's border-collapse property value is set to collapse, add a bottom border to each table cell, regardless of whether it's a **<th>** or **<td>** element. If we wish, we can remove the bottom border from the cells within the last row of the table by using the **:last-child** pseudo-class selector to select the last **<tr>** element within the table and target the **<td>** elements within that row. Additionally, if a table is using the structural elements, we'll want to make sure to prequalify the last row of the table as being within the **<tfoot>** element.

See This Example Code

# Table Striping

In the effort to make tables more legible, one common design practice is to “stripe” table rows with alternating background colors. This makes the rows clearer and provides a visual cue for scanning information. One way to do this is to place a class on every other **<tr>** element and set a background color to that class. Another, easier way is to use the **:nth-child** pseudo-class selector with an **even** or **odd** argument to select every other **<tr>** element.

Table Striping Code

# Text Aligning

In addition to table borders and striping, the alignment of text within cells, both horizontal and vertical, plays an integral role in table formatting. Names, descriptions, and so forth are commonly flush left, while numbers and other figures are flush right. Other information, depending on its context, may be centered.

The **vertical-align** property works only with inline and table-cell elements—it won't work for block, inline-block, or any other element levels.

The **vertical-align** property accepts a handful of different values; the most popular values are **top**, **middle**, and **bottom**. These values vertically position text in relation to the table cell, for table-cell elements, or to the closest parent element, for inline-level elements.

Text aligning example code

# Let’s Practice Making a table together!