Working with Lists and Tables in CSS

Unit-III

SCS181 Web Technologies-I

Er. Anu Arora, Assistant Professor, GNA University

Working with Lists and Tables in CSS

- Lists and tables are essential HTML elements for organizing content on web pages.
- CSS provides various properties to style and control their appearance, allowing developers to create visually appealing, structured layouts.

1. Working with Lists

HTML lists come in two primary types:

- **Ordered Lists** (<ol>) – Displayed with numbered items.
- **Unordered Lists** (<ul>) – Displayed with bullet points.
- There are also **definition lists** (<dl>), used to create lists of terms and definitions.

Key Properties for Lists:

i. list-style-type

Specifies the type of marker (bullet or numbering style) used in lists.

Values:

- For unordered lists: disc (default), circle, square, none (removes bullets).
- For ordered lists: decimal (default), lower-alpha, upper-alpha, lower-roman, upper-roman.

Example:

ul { list-style-type: square;

/* Squares for bullets instead of discs */ }

ol { list-style-type: upper-roman;

/* Roman numerals in ordered list */ }

ii. list-style-position

Controls whether the marker appears inside or outside the list item's content.

Values:

- inside: The marker is placed inside the content area.
- outside: The marker is outside the content area (default).

Example:

ul { list-style-position: inside;

/* Moves the bullet points inside the content area */ }

Key Properties for Lists:

iii. list-style-image

Replaces the default marker (bullet or number) with a custom image.

Example:

```
ul {
list-style-image: url('custom-bullet.png’);
/* Replaces bullets with an image */
}
```

iv. list-style

A shorthand for setting all list style properties (list-style-type, list-style-position, and list-style-image) in one declaration.

Example:

```
ul {
list-style: square inside;
/* Squares for bullets, placed inside content */
}
```

Removing Default List Styles

Sometimes, you may want to remove the default list markers (bullets or numbers) entirely.

Example:

```
ul {
list-style-type: none;
padding-left: 0;
/* Removes default indentation */
 }
```

Customizing Spacing in Lists

You can control spacing between items and indentation using margins and padding.

Example:

```
ul {
padding-left: 20px;
/* Adds space between bullet and list content */
}
li {
margin-bottom: 10px;
/* Adds space between list items */
}
```

Working with Tables

HTML tables allow you to display tabular data in rows and columns. You can style tables using CSS to improve readability and make them more appealing.

a. **Basic HTML for Tables**

<table>: Wraps the entire table.

<thead>: Contains table headers.

<tbody>: Contains the body rows.

<tr>: Defines a row.

<th>: Defines a header cell (bold by default).

<td>: Defines a data cell.

Key Properties for Tables:

i. border

Sets borders around table cells and the table itself.

Example:

```
table, th, td {
border: 1px solid black;
border-collapse: collapse;
/* Combines adjacent borders */
}
```

ii. padding

Adds space inside table cells for better readability.

Example:

```
td, th {
padding: 10px;
}
```

Key Properties for Tables:

ii. text-align

Aligns text inside table cells. Typically, header cells (<th>) are centered, while data cells (<td>) are left-aligned.

Example:

th { text-align: center; }

td { text-align: left; }

iv. width

Controls the width of the entire table or individual columns.

Example:

table { width: 100%;

/* Makes the table full width */ }
th, td { width: 33%;

/* Each column takes up 33% of the table width */ }

Key Properties for Tables:

v. table-layout

Controls the algorithm used to lay out the table columns.

Values:

- auto: Default. Column widths are based on the widest content.
- fixed: Columns have equal widths, or widths are determined by the first row.

Example:

```
table {
table-layout: fixed;
width: 100%; }
```

Zebra Striping for Tables

You can use the nth-child() pseudo-class to create alternating row colors, which improves readability in tables with many rows.

Example:

```
tr: nth-child(even)
{ background-color: #f2f2f2;
/* Lighter background for even rows */ }
tr:nth-child(odd) {
background-color: #ffffff;
}
/*white background for odd rows */ }
```

d. Hover Effects for Tables

To highlight a row when the user hovers over it, you can use the :hover pseudo-class.

Example:

```
tr:hover {
background-color: #f1f1f1;
/* Change background color on hover */
}
```

e. Border Spacing and Collapse

i. border-spacing

Controls the space between borders of adjacent table cells.

Example:

```
table {
border-spacing: 15px;
}
```

ii. border-collapse

Merges adjacent borders of cells into a single border.

Values:

- collapse: Combines borders into a single border.
- separate: Keeps borders separate (default).

Example:

```
table {
border-collapse: collapse; }
```

f. Caption for Tables

You can add a table caption to give a title to the table.

Example:

```
<table>

<caption>Monthly Sales Data</caption>

<thead> <tr> <th>Month</th> <th>Sales</th> </tr> </thead>
<tbody> <tr> <td>January</td> <td>$10,000</td> </tr> </tbody>
</table>
```

Responsive Tables

Tables can be challenging to display on small screens. You can use CSS to make tables more responsive by enabling horizontal scrolling or converting them into a card layout for smaller screens.

a. Adding Horizontal Scroll for Tables

Use overflow-x to allow tables to scroll horizontally when they overflow the screen width.

Example:

```
.table-container { overflow-x: auto; }
table { width: 100%;
/* Ensures the table takes full width */
}
```

b. Stacking Table Cells for Small Screens

Using media queries, you can create a card-like layout where each table row becomes a block of information on small screens.

Example using Media Queries:

```
@media (max-width: 600px)
{
table, thead, tbody, th, td, tr {
display: block; }
th, td {
width: 100%; text-align: left; padding: 10px;
} }
```