❖ CSS Font:

The CSS font properties define the **font family**, **boldness**, **size**, and the **style** of a text.

• Font Family:

- The font family of a text is set with the font-family property.
- The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

Ex:

```
p {
    font-family: "Times New Roman", Times, serif;
}
```

• Font Style:

- The font-style property is mostly used to specify italic text.

Ex:

```
p.normal {
    font-style: normal;
}

p.italic {
    font-style: italic;
}
```

• Font Size:

- The font-size property sets the size of the text.
- Set font size with pixels.

Ex:

```
p {
    font-size: 14px;
}
```

- **Font Weight:**

➢ The font-weight property specifies the weight of a font:

Ex:

```
p.normal {
    font-weight: normal;
}

p.thick {
    font-weight: bold;
}
```

❖ CSS Links:

➢ With CSS, links can be styled in different ways.

- **Styling Links:**

➢ Links can be styled with any CSS property (e.g. color, font family, background, etc.).

Ex:

```
a {
    color: hotpink;
}
```

➢ In addition, links can be styled differently depending on what **state** they are in. The four links states are:

- a:link - a normal, unvisited link.
- a:visited - a link the user has visited.
- a:hover - a link when the user mouses over it.
- a:active - a link the moment it is clicked.

Ex:

```
/* unvisited link */
a:link {
    color: red;
}
/* visited link */
a:visited {
    color: green;
}
/* mouse over link */
a:hover {
    color: hotpink;
}
```

```
/* selected link */
a:active {
    color: blue;
}
```

Note: When setting the style for several link states, there are some order rules:

- ✓ a:hover **MUST** come after a:link and a:visited
- ✓ a:active **MUST** come after a:hover

- **Advanced - Link Buttons:**

➢ This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

Ex:

```
a:link, a:visited {
    background-color: #f44336;
    color: white;
    padding: 14px 25px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
}

a:hover, a:active {
    background-color: red;
}
```

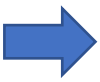
❖ CSS Tables:

➢ The look of an HTML table can be greatly improved with CSS:

- **Table Borders:**
  - ➢ The example below specifies a black border for <table>, <th>, and <td> elements:

Ex:

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



Firstname	Lastname
Peter	Griffin
Lois	Griffin

- **Table Borders:**
  - ➢ The border-collapse property sets whether the table borders should be collapsed into a single border:

Ex:

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

Firstname	Lastname
Peter	Griffin
Lois	Griffin

- **Table Horizontal Alignment:**

- The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.
- By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned.

  Ex:

```
th {
    text-align: left;
}
```

- **Hoverable Table:**

- Use the :hover selector on <tr> to highlight table rows on mouse over:

  Ex:

```
tr:hover {background-color: #f5f5f5;}
```

First Name	Last Name
Peter	Griffin
Lois	Griffin
Joe	Swanson

- **Striped Tables:**

- For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows:

  Ex:

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

First Name	Last Name
Peter	Griffin
Lois	Griffin
Joe	Swanson

- **Responsive Table:**

➢ A responsive table will display a horizontal scroll bar if the screen is too small to display the full content:

➢ Add a container element (like) with overflow-x:auto around the <table> element to make it responsive:

Ex:

```
<div style="overflow-x:auto;">
   <table>
    ... table content ...
   </table>
</div>
```

❖ CSS Position:

➢ The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;

➢ HTML elements are positioned static by default.

➢ Static positioned elements are not affected by the top, bottom, left, and right properties.

➢ An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

Ex:

```
div.static {
    position: static;
    border: 3px solid #73AD21;
}
```

position: relative;

- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

  Ex:

```
div.relative {
    position: relative;
    left: 30px;
    border: 3px solid #73AD21;
}
```

position: fixed;

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
- A fixed element does not leave a gap in the page where it would normally have been located.

  Ex:

```
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;
}
```

position: absolute;

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- **Note:** A "positioned" element is one whose position is anything except static.

Ex:

```
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
}

div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;
}
```

position: sticky;

- An element with position: sticky; is positioned based on the user's scroll position.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).
- In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

Ex:

```
div.sticky {
    position: -webkit-sticky; /* Safari */
    position: sticky;
    top: 0;
    background-color: green;
    border: 2px solid #4CAF50;
}
```

End of CSS Lectures