CSS

CSS Lists

- Unordered lists : represented with bullets
- Ordered lists: with numbers and alphabets

- **list-style-type:** for ul: circle, square
  for ol: upper-roman,lower-alpha

- **list-style-position:** outside(default) , inside

- **To remove default setting:**

```
list-style-type: none;
  margin: 0;
  padding: 0;
```

CSS Tables

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

- **Collapse border:**

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

- **Table width and height:**

```
table {
  width: 100%;
}
```

- **Horizontal Alignment** : horizontal alignment (like left, right, or center) of the content in <th> or <td>.

By default, <th> aligned center and <td> aligned left.

- **Vertical Alignment**: vertical alignment (like top, bottom or middle) of the content in <th> or <td>
- By default it is middle for both <th> , <td>

CSS Tables

- **Table Padding**

- **Horizontal Dividers** : <th> and <td> for horizontal dividers.
- th, td {
  border-bottom: 1px solid #ddd;
  }

- **Hoverable Table** : tr:hover {background-color: #f5f5f5;}

- **Striped Tables**: tr:nth-child(even) {background-color: #f2f2f2;}

- **Table Color**:
- th {
  background-color: #4CAF50;
  color: white;
  }

CSS Display

- Most important css property to control layout.
- Depending on what type of html element it is, every element has default display value. Which is in most elements is **block or inline.**

- **Block level Elements:** always start on new line and take the full width on left and right.

- List of block level elements:
-
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

- **Inline level Elements**: are those who takes only space required and not start from new line.
-
- <a>
- <img>

Cont.

- **Display: none;** mostly used for hide and show. Used a lot in JS.
- The <script> element uses display: none; as default.

- **Override The Default Display Value:**
- li {
  display: inline;
  }
- Will over ride the list display and show horizontal.

- **display:none or visibility:hidden** : elements can be hidden using one of thsese properties.
- Diplay:none will hide the element and page will be shown as there is no element.
- Visibility:hidden will hide the element but it will affect layout.

CSS width and max-width

- Setting the width of block level will prevent it from stretching. By setting width and adding margin auto, the block will take the specific space and rest of the space will be divided in margin equally.

- But width will add a scrollbar on the smaller screen horizontally.

- And **max-width** will handle this properly.

- ```
  div {
    max-width: 500px;
    margin: auto;
    border: 5px solid #73AD21;
  }
  ```
  ```

CSS Layout - The position Property

- position Property : specifies the type of positioning method used for an element.

- position can be :

- 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;

- By default, html elements are positioned **static** and are not affected by the top, bottom, left, and right properties.

- Not positioned in specific way, just with the flow of page.

- div.class {
- position: static;
- border: 3px solid red;
- }

position: relative;

- When you set the position *relative to an element*, without adding any other positioning attributes (top, bottom, right, left) **nothing will happen**. When you add an additional position, such as left: 20px the element will move 20px to the right from its normal position.

- you can see that **this element is relative to itself**. When the element moves, no other element on the layout will be affected.

position: absolute;

- This type of positioning allows you to **place your element precisely where you want it**.
- The positioning is done **relative to the first relatively (or absolutely) positioned parent element**. In the case when there is no positioned parent element, it will be positioned related **directly to the HTML element (the page itself)**.

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.**

Position: sticky;

- An element with position: sticky; is positioned based on the user's scroll position.
- Some browsers do not support it.

- div.class {
- position: sticky;
- top: 0;
- padding: 10px;
- background-color: yellow;
- border: 2px solid #4CAF50;
- }

Overlapping Elements

- Elements can overlap on another element when use positioning.
- Z-index specifies how to display. Front or behind others.
- An element can have a positive or negative stack order.

- Z-index: -1 ;

Overflow

- Kind of problem, when content overflow from fixed height. If content is more than fixed height.
- We can control overflow by using following values:

- **visible** - Default. The overflow is not clipped. The content renders outside the element's box
- **hidden** - The overflow is clipped, and the rest of the content will be invisible
- **scroll** - The overflow is clipped, and a scrollbar is added to see the rest of the content
- **auto** - Similar to scroll, but it adds scrollbars only when necessary

- **Note: The overflow property only works for block elements with a specified height.**

overflow-x and overflow-y

- The overflow-x and overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both).
- overflow-x specifies what to do with the left/right edges of the content.
- overflow-y specifies what to do with the top/bottom edges of the content.

```
div {
        height: 200px;
        width: 200px;
        background-color: lightgrey;
        overflow-y: scroll;
  }
```

float Property

- Used for positioning and formatting content.
- Left
- Right
- None
- Inherit

- Css clear : used after float.
- Left
- Right
- Both : most used

clearfix Hack
in float

- If there are two divs inside a div and on applying left and right float to both divs, the outer div collapse. To overcome this, clearfix can be used.

Can also fix the problem if one div is larger than other inside a div.

```
.clearfix::after {
 content: "";
 clear: both;
 display: table;
}
```

.clearfix will be replaced by any class name and instead of table it can also be block.

Example of adding 3 images side by side

- *** {**
  box-sizing: border-box;
  }
- It gives best result. Does not include the padding , margin, border in the width.

```
* {
  box-sizing: border-box;
}

.img-container {
  float: left;
  width: 33.33%;
  padding: 5px;
}

.clearfix::after {
  content: "";
  clear: both;
  display: table;
}
```

display: inline-block

- Inline-block allow to set width and height of the element and respect the top and bottom margins/paddings , which is not possible with display:inline.

- **Display: block** adds a new line but **not inline-block.**
- Inline-block used a lot to make navigation list.

CSS Layout Align

- Align div center: using margin:auto; div must have width to align center.
- Align image: img {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 40%;
  }

- Left and Right Align - Using position absolute; following example align right.

- .right {
  position: absolute;
  right: 0px;
  width: 300px;
  border: 3px solid #73AD21;
  padding: 10px;
  }
- **Note:** Absolute positioned elements are removed from the normal flow, and can overlap elements.

- Align vertically using padding and text center using text align:
- .center {
  padding: 60px 0;
  border: 3px solid red;
  text-align: center;
  }

CSS Combinators

- There can be more than one selectors in css and combinators explains the relationship between them.

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

- Descendant Selector : matches all elements that are descendants of a specified element. div p { } select all the p in div

- Child selector: matches all the elements that are child of a specified element. div > p { }

Combinators cont.

- Adjacent Sibling Selector: ul + p {color:red }
- It will select the first p that comes after ul tag.

- General sibling selector (~) : ul ~ p {color:red }
- selects all elements that are siblings of a specified element.

- Refence: www.w3schools.com