**The position Property**

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

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.

**static**:- position: static;

**Static positioning**

Static positioning is the default that every element gets. It just means "put the element into its normal position in the document flow — nothing special to see here."

**Relative positioning**

positioning is the first position type we'll take a look at. This is very similar to static positioning, except that once the positioned element has taken its place in the normal flow, you can then modify its final position, including making it overlap other elements on the page. Go ahead and update the position declaration in your code:

**position: relative;**

**Introducing top, bottom, left, and right**

top, bottom, left, and right are used alongside position to specify exactly where to move the positioned element to. To try this out, add the following declarations to the .positioned rule in your CSS:

**CSS:-**

top: 30px;

left: 30px;

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.

**Sticky positioning**

There is another position value available called position: sticky, which is somewhat newer than the others. This is basically a hybrid between relative and fixed position. It allows a positioned element to act like it's relatively positioned until it's scrolled to a certain threshold (e.g., 10px from the top of the viewport), after which it becomes fixed.

**Basic example**

Sticky positioning can be used, for example, to cause a navigation bar to scroll with the page until a certain point and then stick to the top of the page.

.positioned {

position: sticky;

top: 30px;

left: 30px;

}

**Color Values**

**Introduction**:

Color values in CSS allow us to specify the colors we want to use in our web designs. Colors play a crucial role in creating visually appealing and engaging websites. CSS provides various ways to represent colors using different color value formats.

**Real-life example:**

Imagine you're designing a website for a clothing store. You want to use the store's brand color, which is a specific shade of blue, throughout the site. By using color values in CSS, you can precisely specify the exact shade of blue and apply it consistently across different elements. This consistency helps in building a cohesive and recognizable brand identity.

**Different color value formats:**

CSS provides multiple color value formats, including:

**Hexadecimal**: The hexadecimal color value format is represented by a six-digit code starting with a hash symbol (#). For example, #FF0000 represents the color red.

**RGB**: The RGB (Red, Green, Blue) color value format allows you to specify color intensity levels for each primary color. It is represented as rgb(red, green, blue), where each color component ranges from 0 to 255.

**RGBA**: The RGBA color value format is similar to RGB but includes an additional alpha value for transparency. It is represented as rgba(red, green, blue, alpha), where the alpha value ranges from 0 to 1.

**Color Names:** CSS also provides predefined color names, such as "red", "blue", "green", etc. These color names can be used directly to represent specific colors.

**Code-based example:**

Here's an example of how to use different color value formats in CSS:

.heading {

color: #336699;

}

.paragraph {

color: rgb(255, 0, 0);

}

background {

background-color: rgba(0, 128, 0, 0.5);

}

.button {

background-color: purple;

}

In this example, we have used different color value formats to style various elements. The .heading class uses a hexadecimal color value for the text color, the .paragraph class uses an RGB color value, the .background class uses an RGBA color value for the background color with transparency, and the .button class uses a color name directly.