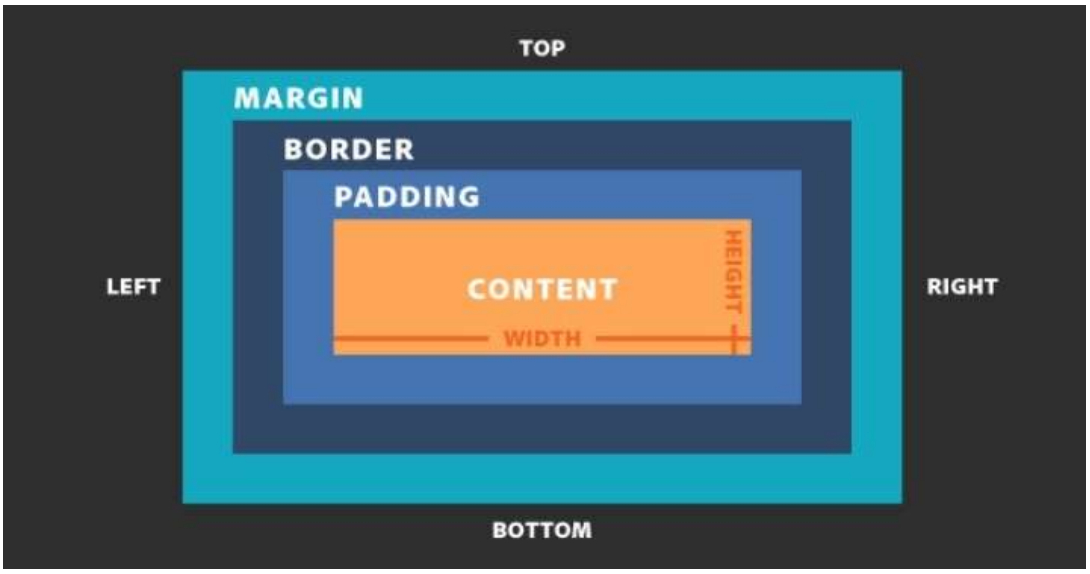
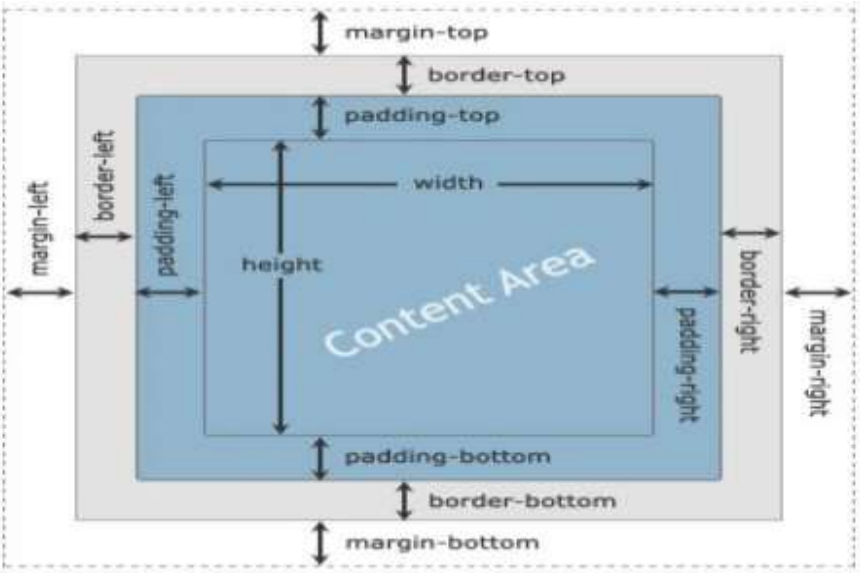
UNIT 1: HTML, CSS & Client Side Scripting

CSS box model

When laying out a document, the browser's rendering engine represents each element as a rectangular box according to the standard CSS basic box model. CSS determines the size, position, and properties (color, background, border size, etc.) of these boxes.



Every box is composed of four parts (or areas), defined by their respective edges: the content edge, padding edge, border edge, and margin edge. Every box has a content area and an optional surrounding margin, padding, and border.

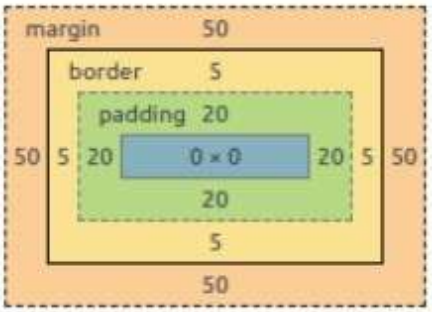


1. The innermost rectangle is the content box. The width and height of this depend on the element's content (text, images, videos, any child elements).
2. Then have the padding box (defined by the padding property). If there is no padding width defined, the padding edge is equal to the content edge.
3. Next, the border box (defined by the border property). If there is no border width defined, the border edge is equal to the padding edge.
4. The outermost rectangle is the margin box. If there is no margin width defined, the margin edge is equal to the border edge.

Consider the following example:

```
div{
 border: 5px solid;
 margin: 50px;
 padding: 20px;
}
```

This CSS styles all div elements to have a top, right, bottom and left border of 5px in width and a top, right, bottom and left margin of 50px; and a top, right, bottom, and left padding of 20px. Ignoring content, our generated box will look like this:



CSS Position

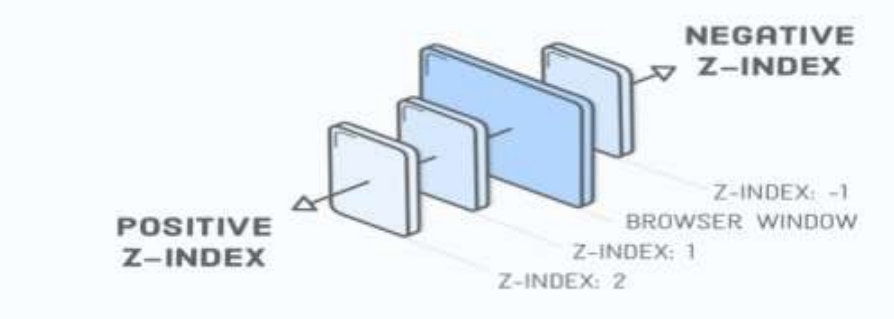
The position CSS property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements. There are 5 main values of the Position Property:

position: static , relative , absolute , fixed , sticky

and additional properties for setting the coordinates of an element : **top , right , bottom , left AND the z-index.**

What is this z-index?

Consider height and width (x, y) as 2 dimensions. Z is the 3rd dimension. An element in the webpage comes in front of other elements as its z-index value increases. Z-index doesn't work with position: static or without a declared position.



The .features-menu element needs to have a lower z-index than the Features label. The default z-index value is 0, so make both of them higher than that. It can conveniently wrapped the Features label in a , allowing to style it via a child selector, like this

```
.dropdown > span {
  z-index: 2;
  position: relative;  /* This is important! */
  cursor: pointer;
}

.features-menu {
  /* ... */
  z-index: 1;
}
```

The Features label appear on top of the submenu. The position: relative; line. It's required because only positioned elements pay attention to their z-index property.

Now, lets see the Position Property:

1. Static

position: static is the default value. Whether declare it or not, elements are positioned in a normal order on the webpage. Let's give an example:

First, define our HTML structure:

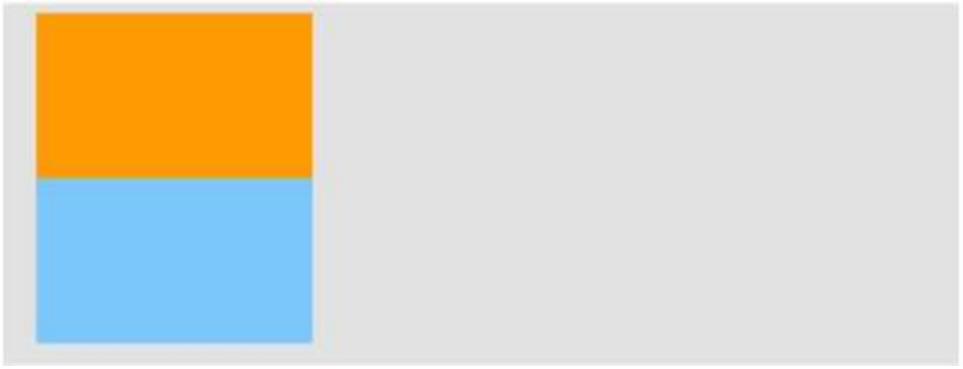
```
<body>
    <div class="box-orange"></div>
    <div class="box-blue"></div>
</body>
```

Then, create 2 boxes and define their widths, heights & positions:

```
.box-orange {          // without any position declaration
    background: orange;
    height: 100px;
    width: 100px;
}

.box-blue {
    background: lightskyblue;
    height: 100px;
    width: 100px;
    position: static;    // Declared as static
}
```

Results:



Hence from the result it can observe defining position: static or not doesn't make any difference. The boxes are positioned according to the normal document flow.

2. Relative

position: relative: An element’s new position relative to its normal position.

Starting with position: relative and for all non-static position values, able to change an element’s default position by using the helper properties. Move the orange box next to the blue one. Orange box is moved 100px to bottom & right, relative to its normal position.

```
.box-orange {
  position: relative;  // We are now ready to move the element
  background: orange;
  width: 100px;
  height: 100px;
  top: 100px;          // 100px from top relative to its old position
  left: 100px;         // 100px from left
}
```

Results:

3.Absolute

In position: relative, the element is positioned relative to itself. However, an absolutely positioned element is relative to its parent.

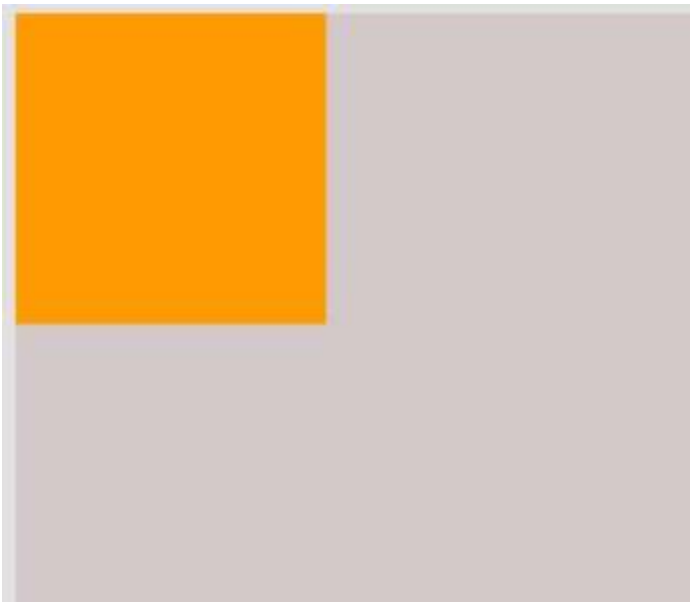
An element with position: absolute is removed from the normal document flow. It is positioned automatically to the starting point (top-left corner) of its parent element. If it doesn't have any parent elements, then the initial document <html> will be its parent.

Since position: absolute removes the element from the document flow, other elements are affected and behave as the element is removed completely from the webpage. Add a container as parent element.

```html
<body>
  <div class="container">
    <div class="box-orange"></div>
    <div class="box-blue"></div>
  </div>
</body>
```

```css
.box-orange {
  position: absolute;
  background: orange;
  width: 100px;
  height: 100px;
}
```

Results:



It looks like the blue box has disappeared, but it hasn't. The blue box behaves like the orange box is removed, so it shifts up to the orange box's place. So Let's move the orange box 5 pixels:

```
.box-orange {
  position: absolute;
  background: orange;
  width: 100px;
  height: 100px;
  left: 5px;
  top: 5px;
}
```

Results:

The coordinates of an absolute positioned element are relative to its parent if the parent also has a non-static position.

4.Fixed

Like position: absolute, fixed positioned elements are also removed from the normal document flow. The differences are:

- They are only relative to the <html> document, not any other parents.
- They are not affected by scrolling.

```css
.container {
  position: relative;
  background: lightgray;
}

.box-orange {
  position: fixed;
  background: orange;
  width: 100px;
  height: 100px;
  right: 5px;    // 5px relative to the most-right of parent
}
```

Here in the example, the orange box's position is changed to fixed, and this time it is relative 5px to the right of the <html>, not its parent (container):

Result :

5.position: sticky

It can be explained as a mix of position: relative and position: fixed.It behaves until a declared point like position: relative, after that it changes its behaviour to position: fixed .

```html
<html>
    <head>
        <title>Example Position: sticky</title>
    </head>
  <body>
    <div class="container">
        <div class="box-orange"></div>
        <div class="box-blue"></div>
      <p>Scroll down the page</p>
      <p class="sticky">I am sticky</p>
    </div>
  </body>
</html>
```

```css
.container {
  position: relative;
  background: lightgray;
  width: 50%;
  margin: 0 auto;
  height: 1000px;
}
.container p {
  text-align: center;
  font-size: 20px;
}
.box-orange {
  background: orange;
  width: 100px;
  height: 100px;
  position: fixed;
  right: 5px;
}
.box-blue {
  background: lightblue;
  width: 100px;
  height: 100px;
}
.sticky {
  position: sticky;
  background: red;
  top: 0;
  padding: 10px;
  color: white;
}
```

Results:

Background-image:

Background-image defines a pointer to an image resource which is to be placed in the background of an element.

Syntax

object.style.backgroundImage = "Any value as defined above";

```
<html>
    <head>
        <style>
            body {
                background-image: url('purple.jpg');
                background-color: #cccccc;
            }
        </style>
    </head>

    <body>
        <h1>Hello World!</h1>
    </body>
</html>
```

Results:

CSS Property: border-style

The border style, combined with border width and border color, can also be specified with the border shorthand property.

With one value, the border-style property can be used to specify a uniform style border around a box. With two, three, or four values, sides can be specified independently.

The border-style property may be specified using one, two, three, or four values.

1. When one value is specified, it applies the same style to all four sides.
2. When two values are specified, the first style applies to the top and bottom, the second to the left and right.
3. When three values are specified, the first style applies to the top, the second to the left and right, the third to the bottom.
4. When four values are specified, the styles apply to the top, right, bottom, and left in that order (clockwise).

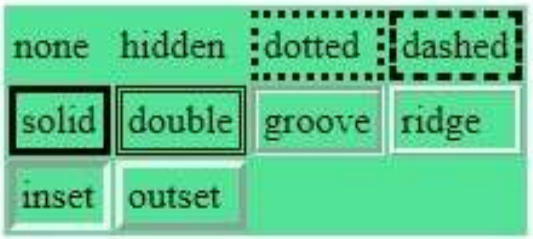
```
<!DOCTYPE html>
<html>

    <link rel="stylesheet" type="text/css" href="broderstyle.css">
    <table>
        <tr>
            <td class="b1">none</td>
            <td class="b2">hidden</td>
            <td class="b3">dotted</td>
            <td class="b4">dashed</td>
        </tr>
        <tr>
            <td class="b5">solid</td>
            <td class="b6">double</td>
            <td class="b7">groove</td>
            <td class="b8">ridge</td>
        </tr>
        <tr>
            <td class="b9">inset</td>
            <td class="b10">outset</td>
        </tr>
    </table>
</html>
```

```css
/* Define look of the table */
table {
  border-width: 3px;
  background-color: #52E396;
}
tr, td {
  padding: 2px;
}

/* border-style example classes */
.b1 {border-style:none;}
.b2 {border-style:hidden;}
.b3 {border-style:dotted;}
.b4 {border-style:dashed;}
.b5 {border-style:solid;}
.b6 {border-style:double;}
.b7 {border-style:groove;}
.b8 {border-style:ridge;}
.b9 {border-style:inset;}
.b10 {border-style:outset;}
/* save as .css */
```

Results:



Consider the below Table for the same.

Value	Description
none	No border.
solid	Solid line.
dotted	Series of dots.
dashed	Series of dashes.
double	Two solid lines.
groove	Representation of a carved groove. Opposite of ridge.
ridge	Representation of an embossed ridge. Opposite of groove.

Rounded Corners

The border-radius CSS property rounds the corners of an element's outer border edge. You can set a single radius to make circular corners, or two radii to make elliptical corners.

This property is a shorthand to set the four properties border-top-left-radius, border-top-right-radius, border-bottom-right-radius, and border-bottom-left-radius.

The radius applies to the whole background, even if the element has no border; the exact position of the clipping is defined by the background-clip property.

The border-radius property does not apply to table elements when border-collapse is collapse.

```
<!DOCTYPE html>
<html>

<head>

    <link rel="stylesheet" href="roundstyle.css">
</head>

<body>
    <div id="roundc1"></div>
    <div id="roundc2"></div>
    <div id="roundc3"></div>
</body>

</html>


<script type="text/javascript" src="scripts.js">
</script>
```

```css
#roundc1 {
    border-radius: 10px;
    background: #5E37BC;
    padding: 15px;
    width: 50px;
    height: 50px;
}
#roundc2 {
    border-radius: 10px;
    border: 3px solid #5E37BC;
    padding: 15px;
    width: 50px;
    height: 50px;
}
#roundc3 {
    border-radius: 10px;
    background: url('purple.jpg');
    background-size: cover;
    background-repeat: repeat;
    padding: 15px;
    width: 50px;
    height: 50px;
}
```

Results: