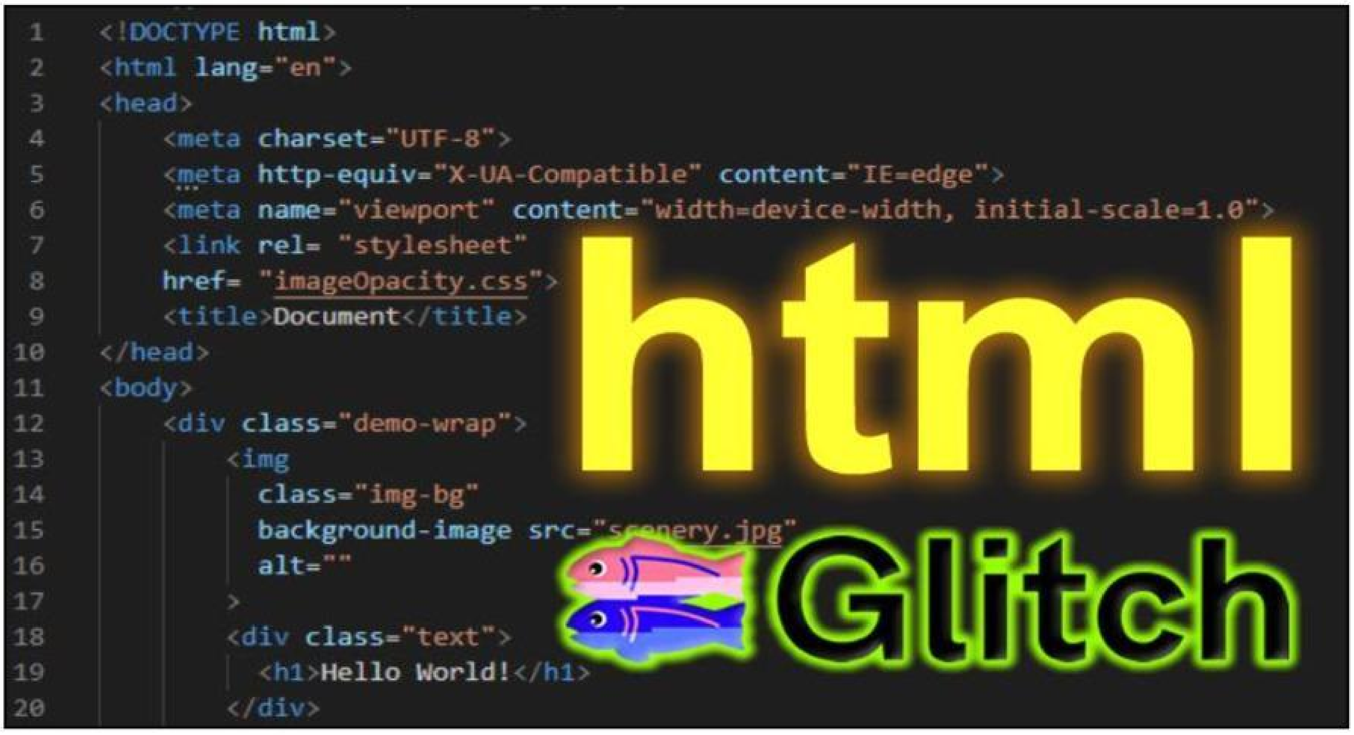
Project 197

DP EDUCATION Coding School



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel= "stylesheet"
    href= "imageOpacity.css">
    <title>Document</title>
</head>
<body>
    <div class="demo-wrap">
        <img
          class="img-bg"
          background-image src="scenery.jpg"
          alt=""
        >
        <div class="text">
          <h1>Hello World!</h1>
        </div>
```

See the webpage

Glitch Start here

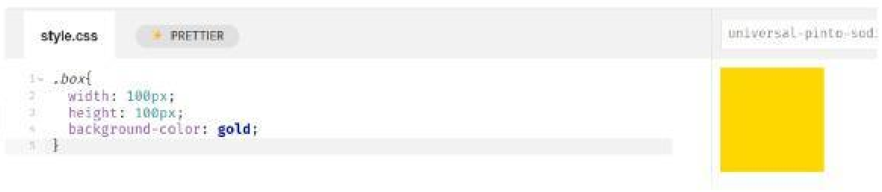
Measurement Units

❖ Measurement Units can be divided into two parts.

ABSOLUTE	RELATIVE	
px	%	Relative to the size of the container
	vw vh	relative to the viewport
	em rem	Relative to the font size

❖ Code the index.html page and the style.css page as follows.

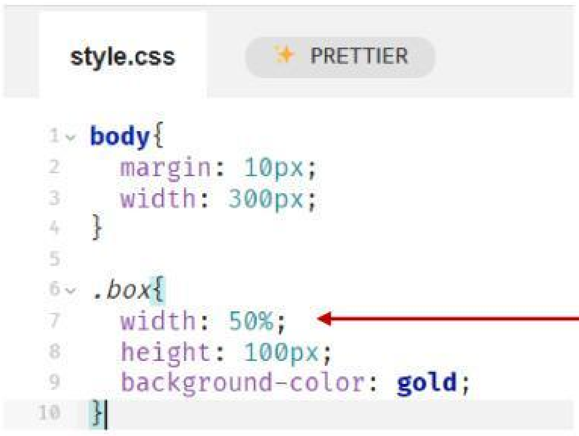
```html
<body>
  <div class="box"></div>
</body>
```



```css
.box{
  width: 100px;
  height: 100px;
  background-color: gold;
}
```

Here, the size of the box is always the same value. Regardless of the size of the screen, the size of the box does not change.

❖ Now let's consider relative values.

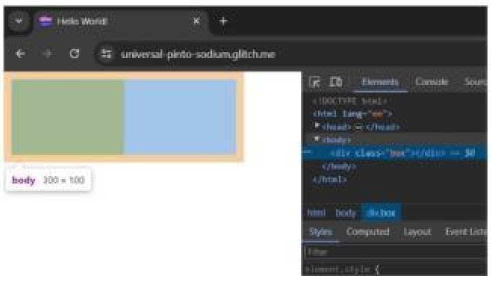


```css
body{
  margin: 10px;
  width: 300px;
}

.box{
  width: 50%;
  height: 100px;
  background-color: gold;
}
```

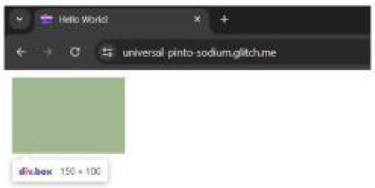
Here the width is given as 50%.

This means that the width of the element is taken as 50% of its parent element.

The parent element of the div box class is the body element.

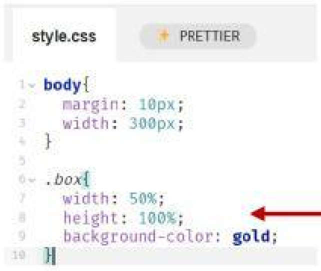


```
<!DOCTYPE html>
<html lang="en">
▶<head> ⋯ </head>
▼<body>
    <div class="box"></div> == $0
  </body>
</html>
```

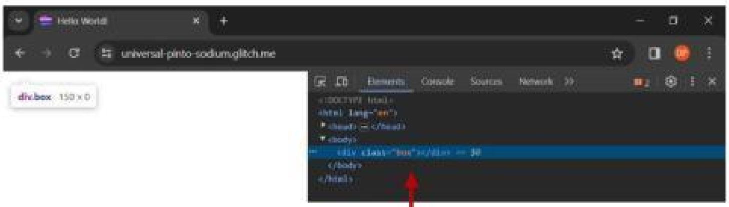


Here, the width of the body element is 300 px, so the width of the box element is 150 px.

- ❖ When the size is not given for the width of the body element, half of the width of the screen is taken for the width of the box element.
- ❖ Now let's give relative values for the height of the box element.



```
body{
  margin: 10px;
  width: 300px;
}

.box{
  width: 50%;
  height: 100%;
  background-color: gold;
}
```

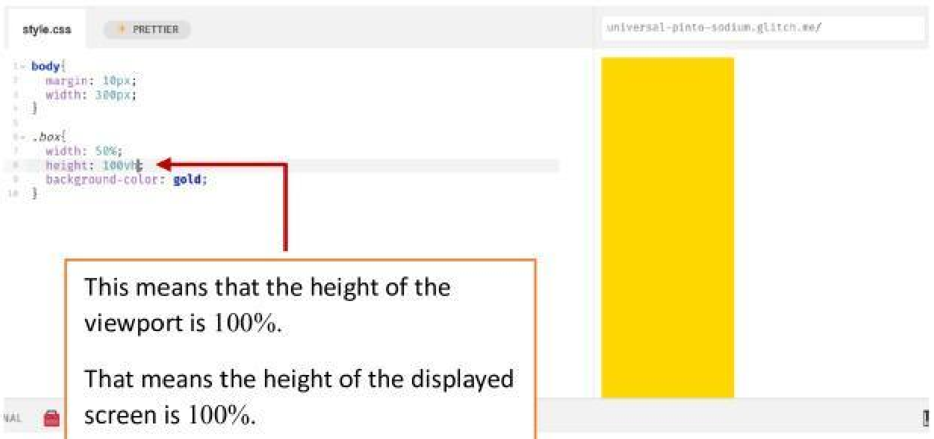


```
<!DOCTYPE html>
<html lang="en">
▶<head> ⋯ </head>
▼<body>
    <div class="box"></div> == $0
  </body>
</html>
```

When Height is given as 100%, the box does not jump.

Here the height of the box is zero. The height for the box cannot be given as a percentage.

- ❖ Viewport height is used for that.



```css
body{
  margin: 10px;
  width: 300px;
}

.box{
  width: 50%;
  height: 100vh;
  background-color: gold;
}
```

This means that the height of the viewport is 100%.

That means the height of the displayed screen is 100%.

Even if you change the size of the browser, the height of the box will change according to the size of the screen.

- ❖ vw is the viewport width. Similarly, vh is used for height.

- ❖ em means the size of the relevant element is adjusted according to the font size.

```css
.box{
  width: 10em; /* 10 x font size of current element */
  height: 100vh;
  background-color: gold;
}
```

Here, the width of the box element is equal to the font size of that element multiplied by the value we give.

Since the value we give here is 10, it should be 10 times the font size of the box element.

But since a font size is not defined in the box element, the font size of the parent element is used.

But here a font size is not defined in the body element.

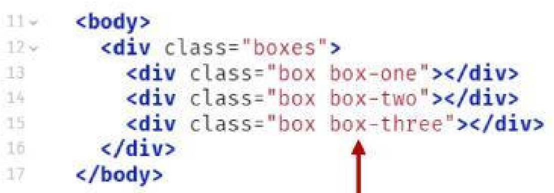
So here the font size of the html element is taken. The browser gives the base font size for the html element as 16 pixels.

Accordingly, the width should be 16 x 10 = 160 pixels. Preview it. The width of the Box element should be 160 pixels.

- ❖ rem is the size of the corresponding element according to the font size of the root element. The root element is the html element.

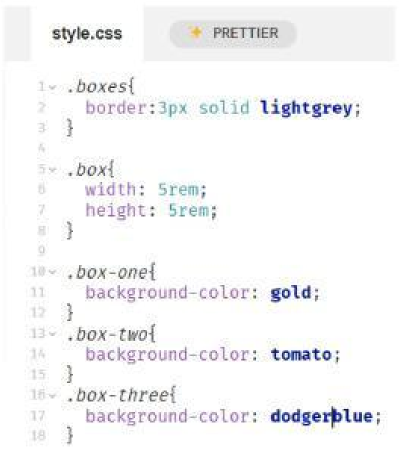
Positioning

- ❖ Here we get how to position an element correctly and how to prepare a layout.
- ❖ For that, let's code as follows.



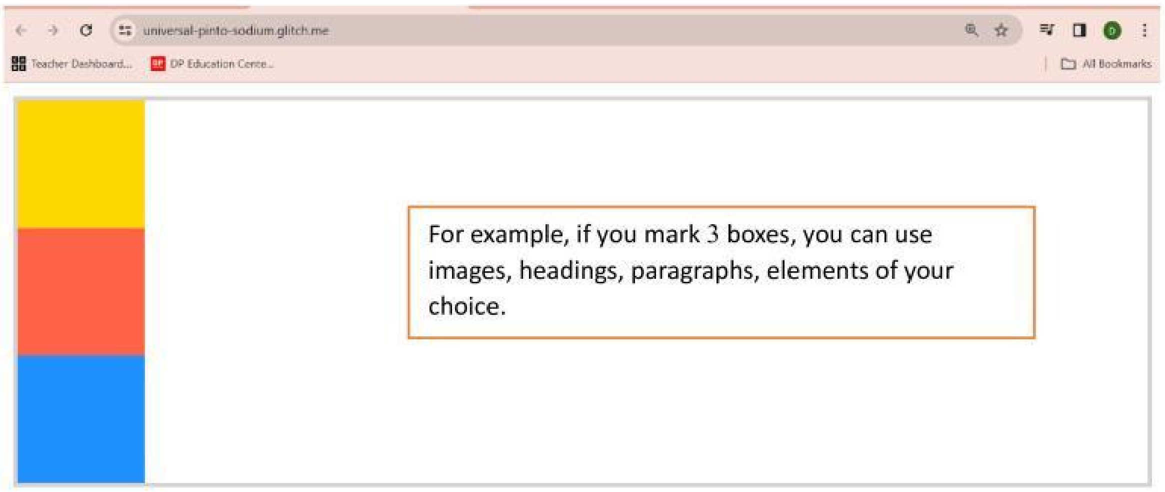
```html
<body>
  <div class="boxes">
    <div class="box box-one"></div>
    <div class="box box-two"></div>
    <div class="box box-three"></div>
  </div>
</body>
```

Let's add the extra class to add background colors for boxes.



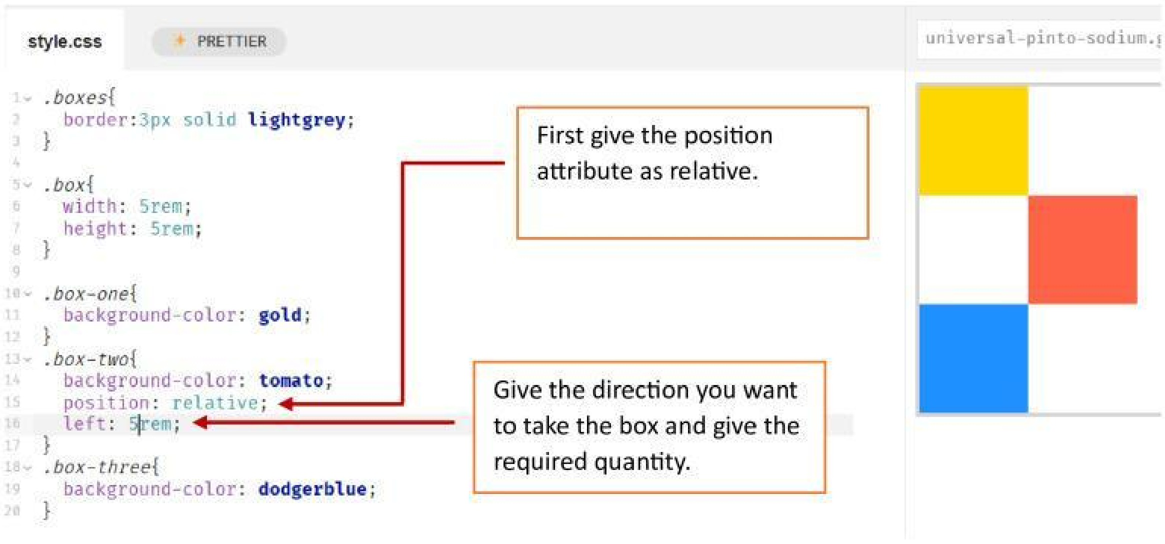
```css
.boxes{
  border:3px solid lightgrey;
}

.box{
  width: 5rem;
  height: 5rem;
}

.box-one{
  background-color: gold;
}
.box-two{
  background-color: tomato;
}
.box-three{
  background-color: dodgerblue;
}
```

- ❖ Then the result will be as follows.



For example, if you mark 3 boxes, you can use images, headings, paragraphs, elements of your choice.

❖ Let's assume that the second box needs to be taken in the right direction. Then the position property is used.

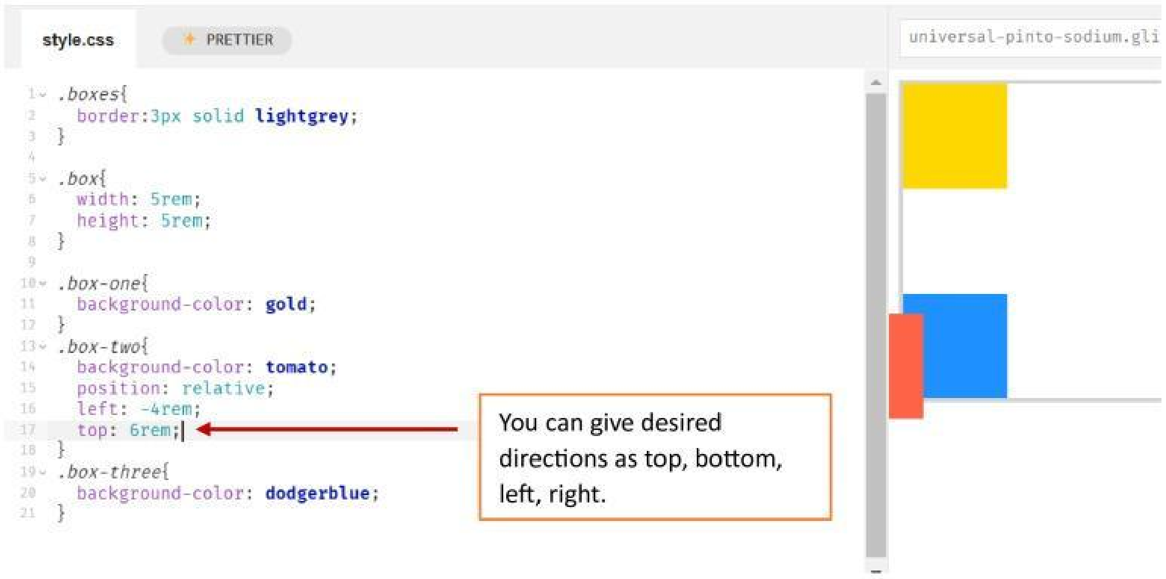


```css
.boxes{
  border:3px solid lightgrey;
}

.box{
  width: 5rem;
  height: 5rem;
}

.box-one{
  background-color: gold;
}
.box-two{
  background-color: tomato;
  position: relative;
  left: 5rem;
}
.box-three{
  background-color: dodgerblue;
}
```

First give the position attribute as relative.

Give the direction you want to take the box and give the required quantity.

❖ Positive values as well as negative values can be given here.

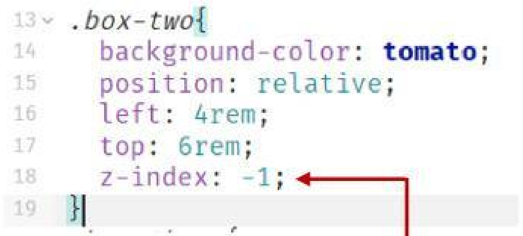


```css
.boxes{
  border:3px solid lightgrey;
}

.box{
  width: 5rem;
  height: 5rem;
}

.box-one{
  background-color: gold;
}
.box-two{
  background-color: tomato;
  position: relative;
  left: -4rem;
  top: 6rem;
}
.box-three{
  background-color: dodgerblue;
}
```

You can give desired directions as top, bottom, left, right.

❖ Here the orange color box is above the blue orange color box.

❖ Let's code as below to raise the blue box.

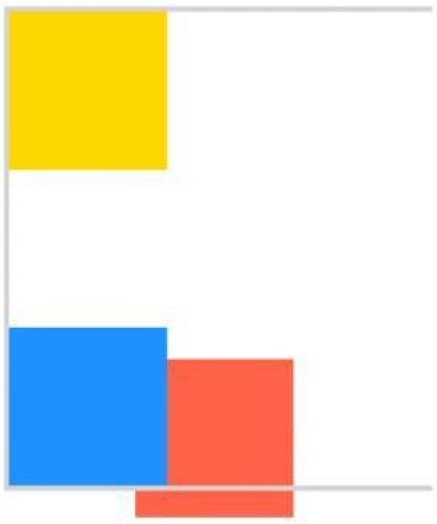


```css
.box-two{
  background-color: tomato;
  position: relative;
  left: 4rem;
  top: 6rem;
  z-index: -1;
}
```

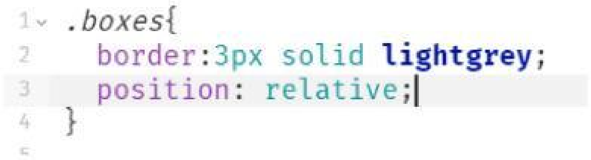
The default value of z-index is 0. The value can be positive as well as negative. Here, the value of z-index in the blue box is 0.

Since the value of the orange box is -1, the orange box goes down and the blue box is at the top.

When there are two elements placed on top of each other, the element with higher z-index value is placed higher.



- ❖ We can move the element in relative position. Relative to the normal position of that element. It does not affect other elements on the page.
- ❖ Now let's see about absolute position.
- ❖ Here an element is moved relative to the container. Now let's see how to do it.
- ❖ First go to the container and give its position attribute as relative.

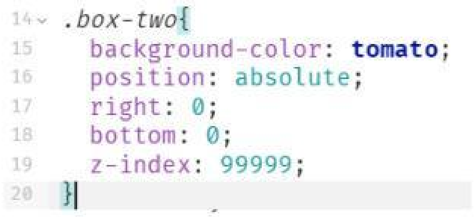


```css
.boxes{
  border:3px solid lightgrey;
  position: relative;
}
```

This is a very important step.

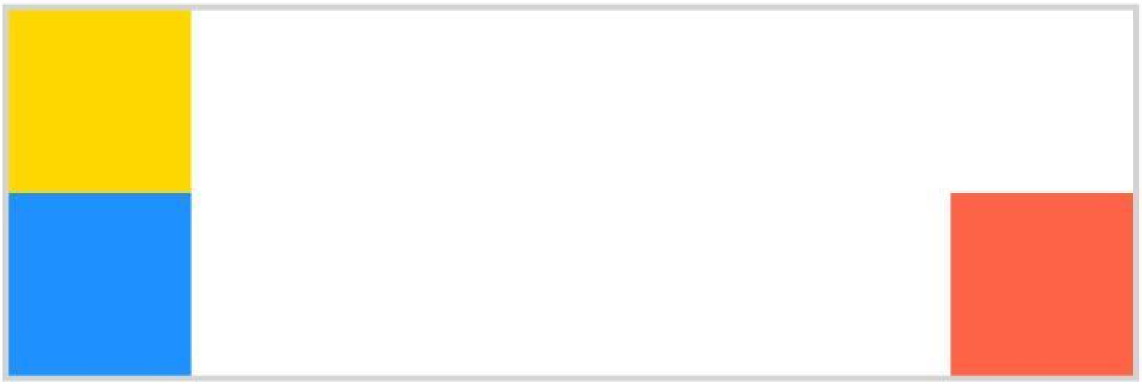
If the orange box is moved relative to its container, the position attribute of its container should be relative.

- ❖ Now go to the Orange box and enter the following code.



```css
.box-two{
  background-color: tomato;
  position: absolute;
  right: 0;
  bottom: 0;
  z-index: 99999;
}
```

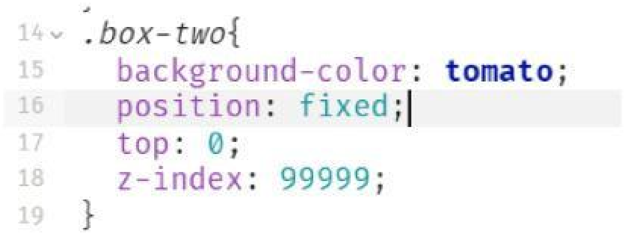
❖ Then it will look like below.



Here the Orange box is positioned relative to the container. The important thing here is that the blue box is positioned at the place where the orange box used to be. In absolute position, that element is removed from the normal flow. That is, when looking at the parents' side, that element does not exist. We can imagine that it has another layer. So the container sees only the yellow box and the blue box. The orange box is the different layer.

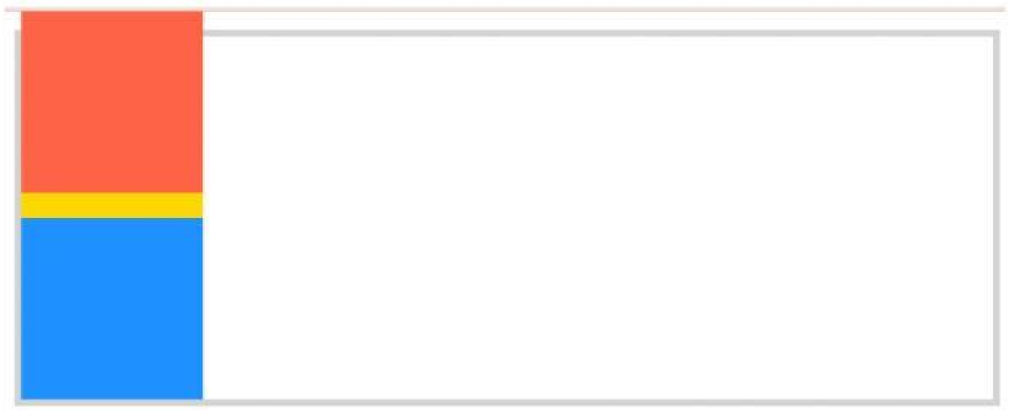
❖ Now let's see about fixed position.

❖ Go to the orange box and enter the code below.



```css
.box-two{
  background-color: tomato;
  position: fixed;
  top: 0;
  z-index: 99999;
}
```

❖ Then it will look like below.



Even if you scroll down here, the orange box is always at the top.