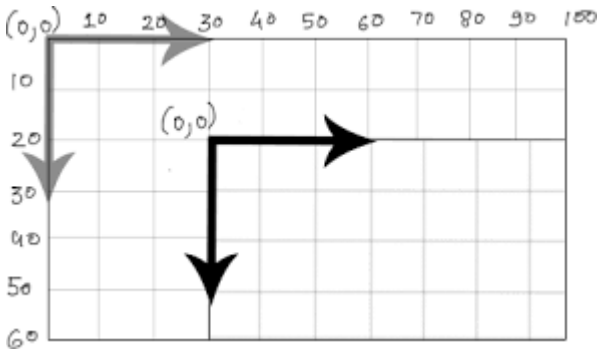
- **Coordinates [Image -1]**



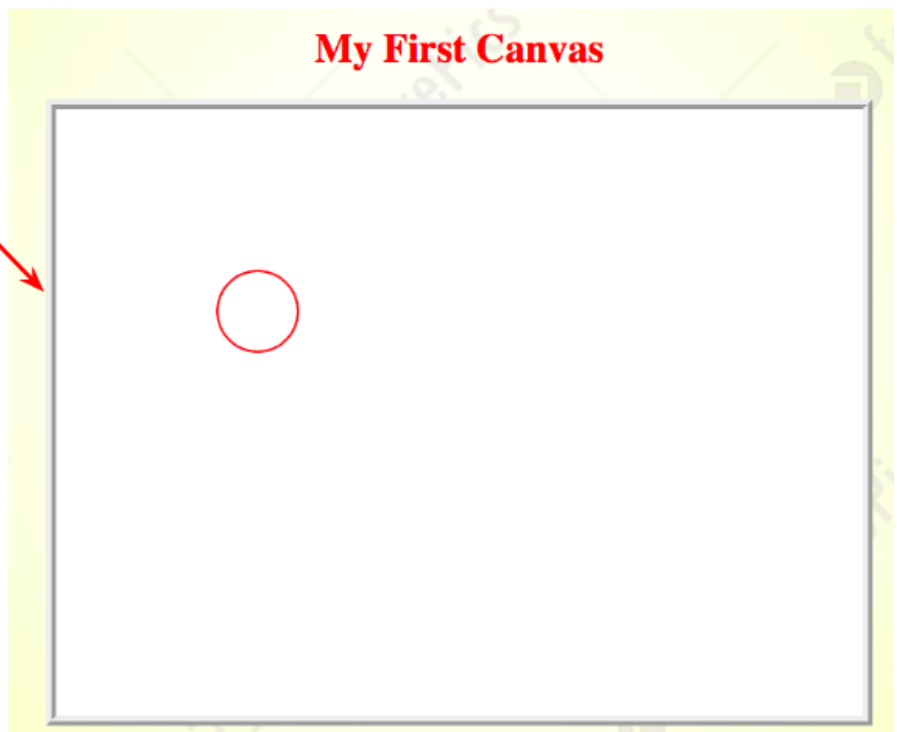
- **HTML code fo canvas -**

```
<body class="body_backgorund">

<center>
    <h1>My First Canvas</h1>
    <canvas id="myCanvas" width="800" height="600">
    </canvas>
    <br>
```

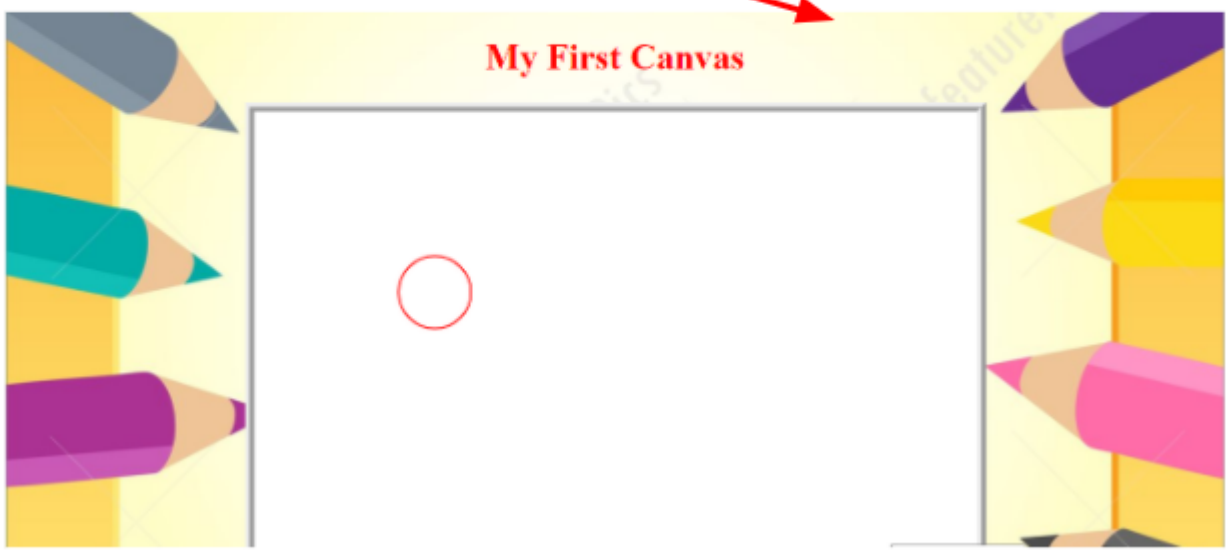
Output -

```
<center>
    <h1>My First Canvas</h1>
    <canvas id="myCanvas" width="800" height="600">
    </canvas>
    <br>
```



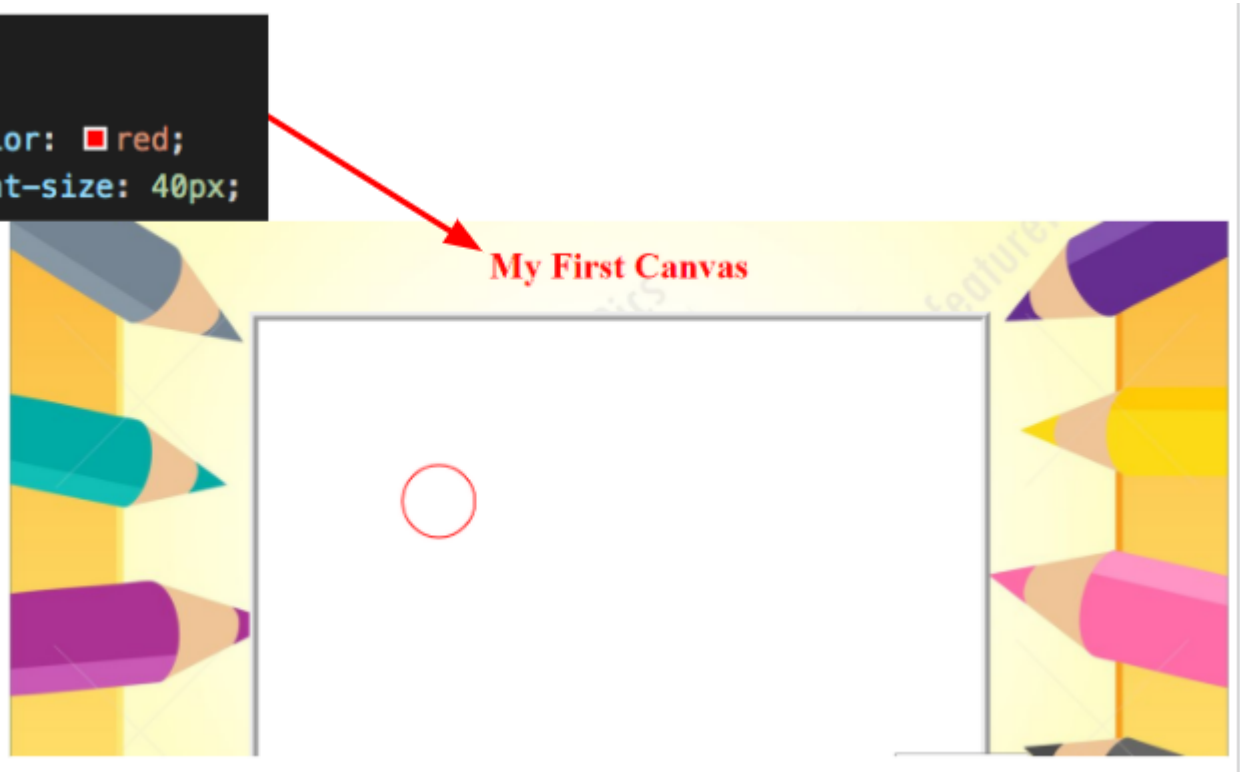
- **Style background of the web page**

```
.body_backgorund {
  background-image: url("bg1.jpg");
  background-position: center;
  background-size: cover;
```



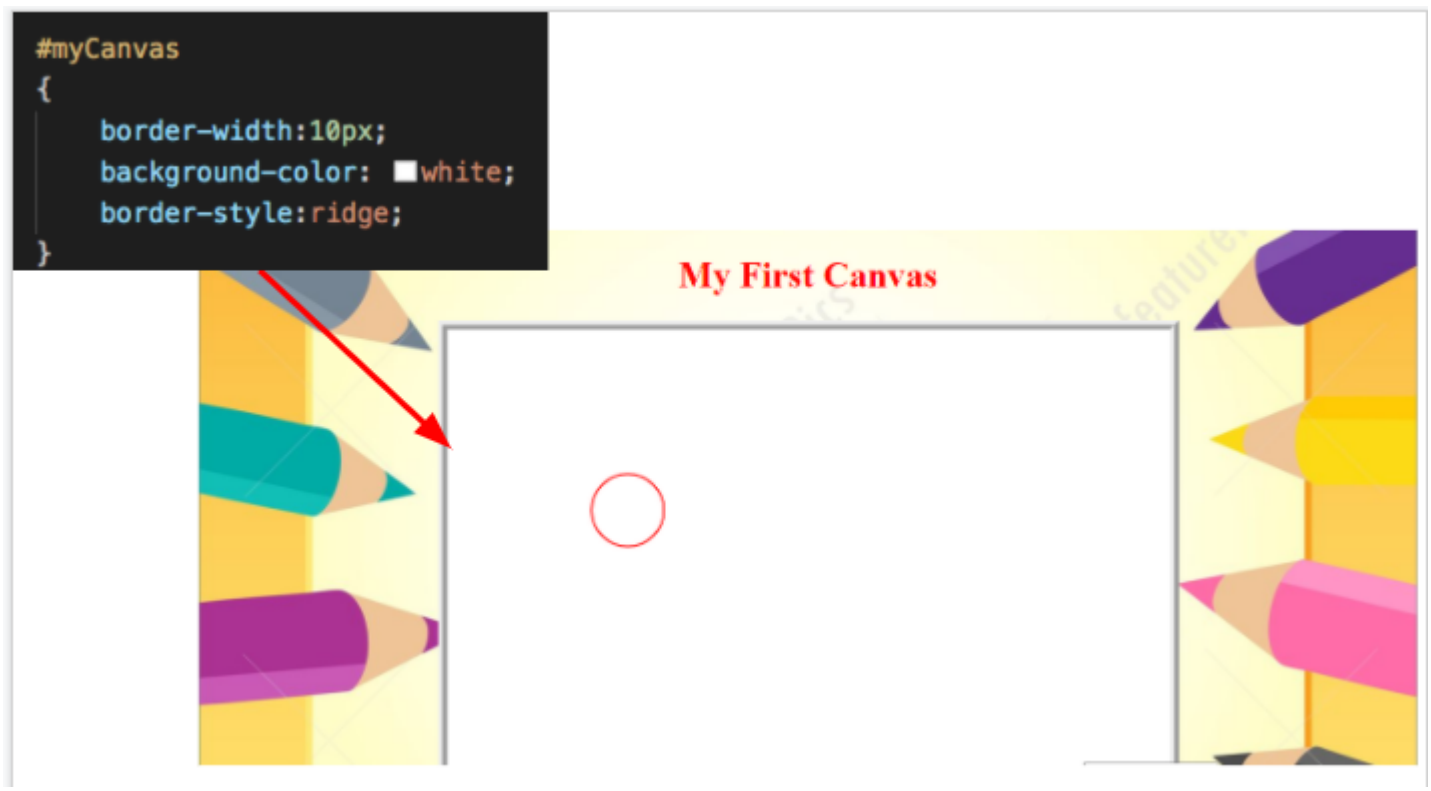
- **Style h1**

```
h1
{
    color: red;
    font-size: 40px;
```



- **Style canvas**

```
#myCanvas
{
    border-width:10px;
    background-color: white;
    border-style:ridge;
}
```



- **JS Code**

Storing HTML Canvas element in variable

```
canvas = document.getElementById("myCanvas");
```

```
color = "red";
```

- **Professional coding**

```
//Workable Code
canvas.getContext("2d").beginPath();
canvas.getContext("2d").strokeStyle =
color;
canvas.getContext("2d").lineWidth = 2;
canvas.getContext("2d").arc(200, 200,
40 ,0 , 2*Math.PI);
canvas.getContext("2d").stroke();
```

The above code will work but we moving to professional coding so we will use the following code

```
ctx= canvas.getContext("2d");
```

```
ctx.beginPath();
ctx.strokeStyle = color;
ctx.lineWidth = 2;
ctx.arc(200, 200, 40 ,0 , 2 * Math.PI);
ctx.stroke();
```

So as a result of professional coding we have reduced the line of the code.

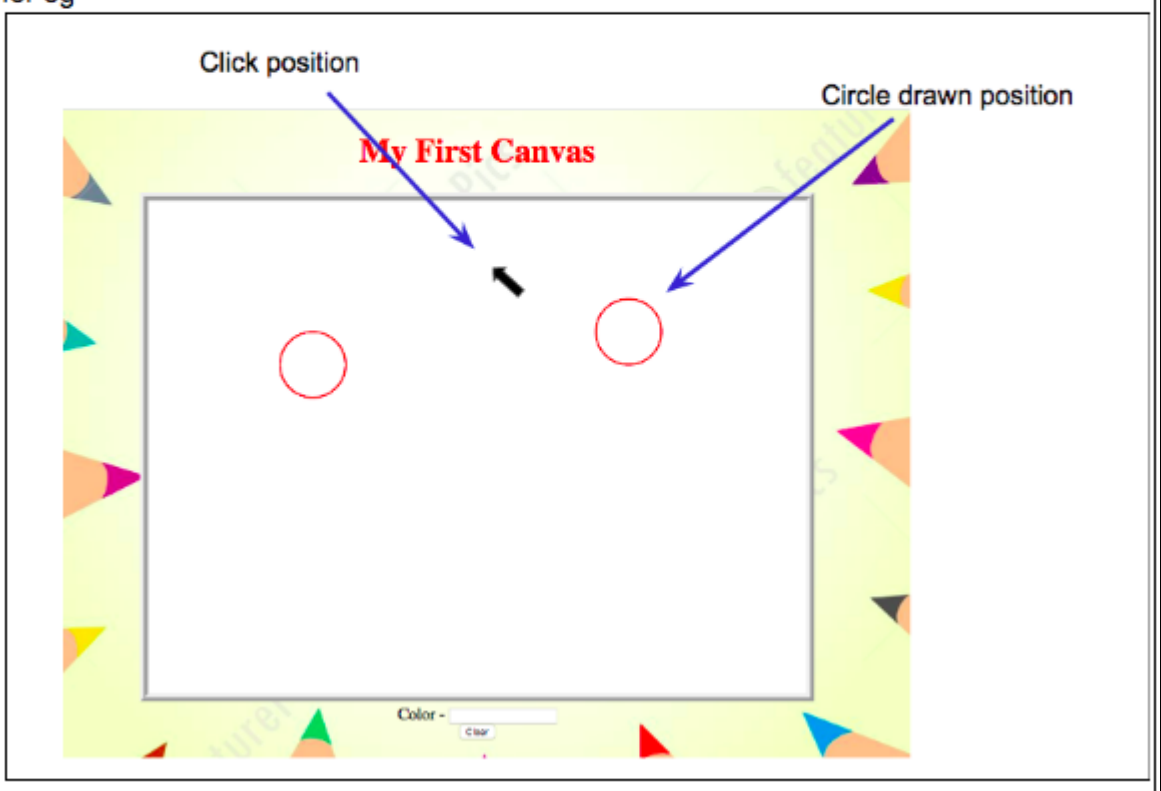
- **Explanation of offset**

make the code as -

```
mouse_x = e.clientX
```

```
mouse_y = e.clientY
```

And then run the code, and try to draw the circle you will see when you click on the canvas, the circle is not getting drawn at the point where you clicked, it's getting drawn somewhere else
for eg -

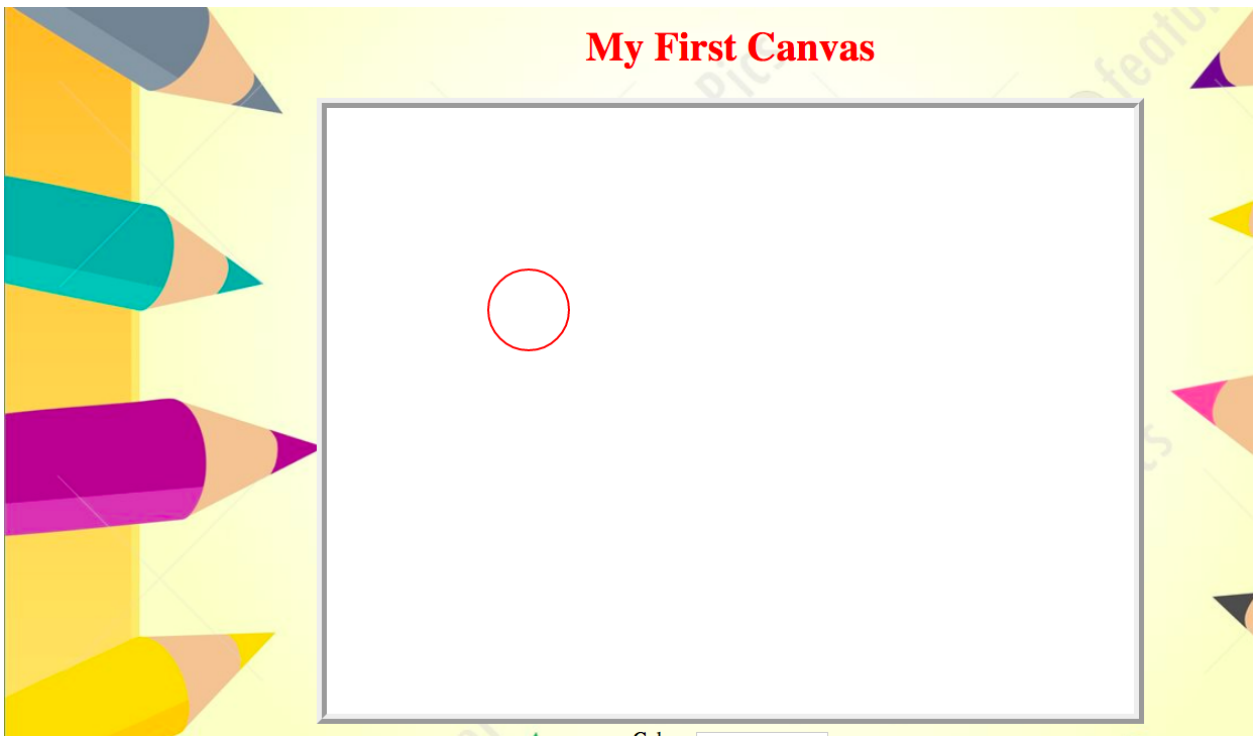


So that's why we have to use offset for getting the actual position of the coordinates

- **Predefine circle code [Image -2]**

```
ctx.beginPath();
ctx.strokeStyle = color;
ctx.lineWidth = 2;
ctx.arc(200, 200, 40 ,0 , 2 * Math.PI);
ctx.stroke();
```

Output



Explaining addEventListener

Syntax -
element.addEventListener("event", my_function);
my_function(e)
{
//any code
}

- **element** can be any HTML element
- **addEventListener -** it sits with the element and when the event occurs it runs the function, this is like the same we used in AddListener block in our Chatapp. Remember? We used this block to monitor if the chat message sent to the firebase
- **event -** it can be any event for eg - **click, mousemove, mousedown(means when the mouse is clicked)**
- **my_function -** it will be any function we define, so when this event occurs we want this function(which we have defined) to occur.
- **my_function(e) -** it will be the function we define. This function will perform certain tasks which are written inside it. **e** means the event of the function. This **e** has relation with the event, for eg if the event is **mousedown**, then this **e** has relation with **mousedown** event.

For eg -

```
button = document.getElementById("button");
button.addEventListener('click' , my_function);
function my_function(e) {
    console.log('I am button');
}
```

First we get the button which is the HTML element and put it inside the **variable button**.
Then we will attach the button variable to the **addEventListener**
Then we will define the type of event, here we have define **click event.**
Then we will call our function which is **my_function**
Now define our function **my_function**, which will be the code for consoling "**I am button**" on the console screen.
So when the button is clicked The function will be executed and it will print "I am button" in the console.

- **ARC**

Syntax - arc(x, y, r, startAngle, endAngle);

x - The horizontal coordinate of the arc's center, which is x-coordinate.
y - The vertical coordinate of the arc's center, which is y-coordinate.
r - The arc's radius.
startAngle - The angle at which the arc starts, measured from the x-axis.
endAngle - The angle at which the arc ends, measured from the x-axis.

- **addEventListener code -**

```
canvas.addEventListener("mousedown", my_mousedown);

function my_mousedown(e)
{
    //taking color from input box
    //additional activity start
    color = document.getElementById("color").value;
    console.log(color);
    //addition activity ends

     mouse_x = e.clientX - canvas.offsetLeft;
     mouse_y = e.clientY - canvas.offsetTop;

    console.log("X = " + mouse_x + " ,Y =  " + mouse_y);
    circle(mouse_x , mouse_y);
}
```

Output of - `console.log("X = " + mouse_x + " ,Y =  " + mouse_y);`

```
X = 273 ,Y =  184                    main.js:26
                                     main.js:20
X = 631 ,Y =  278                    main.js:26
```

- **Circle code -**

```
function circle(mouse_x , mouse_y)
{
ctx.beginPath();
ctx.strokeStyle = color;
ctx.lineWidth = 2;
ctx.arc(mouse_x, mouse_y, 40 ,0 , 2*Math.PI);
ctx.stroke();
}
```

Play around with this to learn more about arc-
https://mahdihat791.github.io/arc_learn/