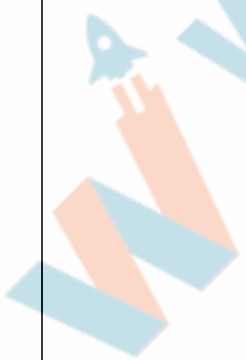


| | | |
|---------------------------------|---|--|
| Topic | CANVAS | |
| Class Description | Extending the Applications of Web, kids learn to visualize shapes and drawing using Canvas leading to creating Web Apps with high user interaction. | |
| Class | ADV-C82 | |
| Class time | 55 mins | |
| Goal | <ul style="list-style-type: none"> ● Make a web-base paint application. | |
| Resources Required | <ul style="list-style-type: none"> ● Teacher Resources <ul style="list-style-type: none"> ○ Use gmail login credentials ○ Earphone with mic ○ Notepad and Pen ● Student Resources <ul style="list-style-type: none"> ○ Use gmail login credentials ○ Earphone with mic (optional) ○ Notepad and Pen | |
| Class structure | Warm Up Teacher-Led Activity Student-Led Activity Wrap Up | 5 Mins 15 Mins 35 Mins 5 Mins |
| Class Steps | Say | Do |
| Step 1: Warm up (5 mins) | <p>In the last class, we learned how to use canvas, draw circles on canvas, and what are x and y coordinates on canvas.</p> <p>Now taking this to the next level by not just restricting ourselves to make circles, we will be making any shape</p> | Ask the student to get into Fullscreen mode. |

| | | |
|---|--|--|
| | <p>or design we want same as we do in the paint application.</p> <p>In today's class we will build a mini paint web app, sounds exciting!</p> <p>And we will achieve this by using some functions of mouse events on canvas.</p> | |
| Teacher Initiates Screen Share | | |
| <p>Step 2: Teacher-Led Activity (15 mins)</p> |  | <p>Teacher should run Teacher-Activity-1 and show it to the student how we can paint on the canvas.</p> <p>Also open the console, and just click inside the canvas. Don't leave the click and move the mouse very slowly inside the canvas.</p> <p>As you move your mouse the Last coordinates of x and y are printed on the console screen. Also the new x and y coordinate are printed on the console screen.</p> |
| <p>*Note: Please follow the steps mentioned in student activity, when you start student activity.</p> <p>Understanding mouse events:</p> <p>Before starting the class first make the student understand what mouse events are.</p> <p>Run Teacher-Activity-4. and follow the steps given below:</p> | | |

In this we have assigned the div with **addeventlistener** and these events are mouse events. We have done similar thing in the previous class where we had assigned the canvas with **addeventlistener** and the event was a mousedown event.

1. **First just move the mouse inside the div**, and you can see the color changes to **green** and the text changes to **mouse move**. This means only a **mousemove** event has occurred.
2. **Now just click inside the div and do not leave the click**, and you can see the color changes to **red** and the text changes to **mouse down**. This means only a **mousedown** event has occurred.
3. **Now leave the click**, and you can see the color changes to **blue** and the text changes to **mouse up**. This means only a **mouseup** event has occurred.
4. **Now just click inside the div and don't leave the click and move the mouse inside the div**. You can see the color changes to **yellow** and the text changes to **mouse down + mouse up**. This means **mousedown** and **mouseup** events have occurred together.

So for achieving the painting functionality on canvas we will follow the same logic of the forth point, which means **we will only paint on the canvas when mousedown and mouseup events occur together**.

5. **Now move the mouse out of the div**, and you can see the color changes to **grey** and the text changes to **mouse leave**. It says mouse leave because mouse has left the div. This means only a **mouseleave** event has occurred.

Javascript coding:

1. First we will get the **canvas** and make its reference as we did in the previous class.

```
canvas = document.getElementById('myCanvas');  
ctx = canvas.getContext("2d");
```

2. Then we will declare a variable to store the mouse event as it occurs, so that we can track which mouse event is happening, and also check if the mousedown event and mousemove event are occurring together. If yes then we will draw. Same way when we clicked inside the div and moved the mouse, the color of the div became yellow in [Teacher-Activity-4](#)

```
var mouseEvent = "empty";
```

- We have kept its value as empty because when the page loads no mouse

event has occurred.

- So when **mouseDown** event occurs, this variable will be updated to **mouseDown**.
- When **mouseUp** event occurs, this variable will be updated to **mouseUp**.
- When **mouseLeave** event occurs, this variable will be updated to **mouseLeave**.
- For **mousemove** we won't update this variable because, on mouseMove we will draw.

3. Now we will declare two variables to store the last x and y coordinates of the mouse.

- This variable will be used for drawing purposes. More will be explained later in the class.

```
var mouseEvent = "empty";  
var last_position_of_x, last_position_of_y;
```

4. Now define two variables - one having color and second having the width. These variables will be used for drawing.

```
color = "black";  
width_of_line = 1;
```

5. Now we will write the mouseDown event.

```
canvas.addEventListener("mousedown", my_mousedown);  
|  
function my_mousedown(e)  
{  
    //Addictional Activity start  
    color = document.getElementById("color").value;  
    width_of_line = document.getElementById("width_of_line").value;  
    //Addictional Activity ends  
  
    mouseEvent = "mouseDown";  
}
```

Explaining the code:

- First we will add a mousedown event listener to the canvas.

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

- Then we will define the **my_mousedown** function.

```
function my_mousedown(e)  
{
```

***Note:** The below code is an additional activity, don't do it now, do it after all the main activity is finished. And the explanation is there in the additional activity section.

```
//Addictional Activity start  
color = document.getElementById("color").value;  
width_of_line = document.getElementById("width_of_line").value;  
//Addictional Activity ends
```

//additional activity ends

- Now we will update the **mouseEvent** variable with the value **mouseDown**, because this is inside mouseDown event.

```
mouseEvent = "mouseDown";
```

6. Now we will write the mouseUp event.

```
canvas.addEventListener("mouseup", my_mouseup);  
function my_mouseup(e)  
{  
    mouseEvent = "mouseUP";  
}
```

Explaining the code:

© 2020 - WhiteHat Education Technology Private Limited.

Note: This document is the original copyright of WhiteHat Education Technology Private Limited.

Please don't share, download or copy this file without permission.

- We will add a mouseup event listener to the canvas.

```
canvas.addEventListener("mouseup", my_mouseup);
```

- Define **my_mouseup** function.

```
function my_mouseup(e)  
{
```

- Now we will update the **mouseEvent** variable with the value **mouseUp**, because this is inside **mouseUp** event.

```
mouseEvent = "mouseUP";
```

7. Now we will write the mouseLeave event.

```
canvas.addEventListener("mouseleave", my_mouseleave);  
function my_mouseleave(e)  
{  
    mouseEvent = "mouseleave";  
}
```

Explaining the above code:

- We will add a **mouseleave** event listener to the canvas.

```
canvas.addEventListener("mouseleave", my_mouseleave);
```

- Define **my_mouseleave** function.

```
function my_mouseleave(e)  
{
```

- Now we will update the **mouseEvent** variable with the value **mouseleave**,

because this is inside mouseleave event.

```
mouseEvent = "mouseleave";
```

8. Now we will write mouseMove event.

- This is the main event where the drawing will happen.

```
canvas.addEventListener("mousemove", my_mousemove);
function my_mousemove(e)
{
    current_position_of_mouse_x = e.clientX - canvas.offsetLeft;
    current_position_of_mouse_y = e.clientY - canvas.offsetTop;

    if (mouseEvent == "mousedown") {
        ctx.beginPath();
        ctx.strokeStyle = color;
        ctx.lineWidth = width_of_line;

        console.log("Last position of x and y coordinates = ");
        console.log("x = " + last_position_of_x + "y = " + last_position_of_y);
        ctx.moveTo(last_position_of_x, last_position_of_y);

        console.log("Current position of x and y coordinates = ");
        console.log("x = " + current_position_of_mouse_x + "y = " + current_position_of_mouse_y);
        ctx.lineTo(current_position_of_mouse_x, current_position_of_mouse_y);
        ctx.stroke();
    }

    last_position_of_x = current_position_of_mouse_x;
    last_position_of_y = current_position_of_mouse_y;
}
```

Explaining the code:

- We will add a **mousemove event** listener to the canvas.

```
canvas.addEventListener("mousemove", my_mousemove);
```

- Define **my_mousemove** function.

```
function my_mousemove(e)
{
```

- We will get the current position of x coordinate by doing the following:

```
e.clientX - canvas.offsetLeft;
```

- **e.clientX** will give the x coordinate of the cursor on the canvas when clicked.
- As the canvas can be placed anywhere on the screen and by just using **e.clientX** it won't give the actual x coordinate on the canvas. So to get the actual x coordinate on canvas with respect to the screen when clicked we do:
e.clientX - canvas.offsetLeft.
- After getting the x coordinate we will store it in variable **current_position_of_mouse_x**.

```
current_position_of_mouse_x = e.clientX - canvas.offsetLeft;
```

- We will get the current position of y coordinate by doing the following:

```
e.clientY - canvas.offsetTop;
```

- **e.clientY** will give the y coordinate on the canvas when clicked.
- As the canvas can be placed anywhere on the screen and by just using **e.clientY** it won't give the actual y coordinate on the canvas. So to get the actual y coordinate on canvas with respect to the screen when clicked we do:
e.clientY - canvas.offsetTop.
- After getting the y coordinate we will store in variable **current_position_of_mouse_y**.

```
current_position_of_mouse_y = e.clientY - canvas.offsetTop;
```

- Now we will check if the mouseDown event has occurred then only we

draw.

- We are doing this because we want that drawing should only be done when the mouse is clicked and moved. The same way when we clicked inside the div and moved the mouse then only the color of the div became yellow in [Teacher-Activity-4](#)

```
if (mouseEvent == "mouseDown") {
```

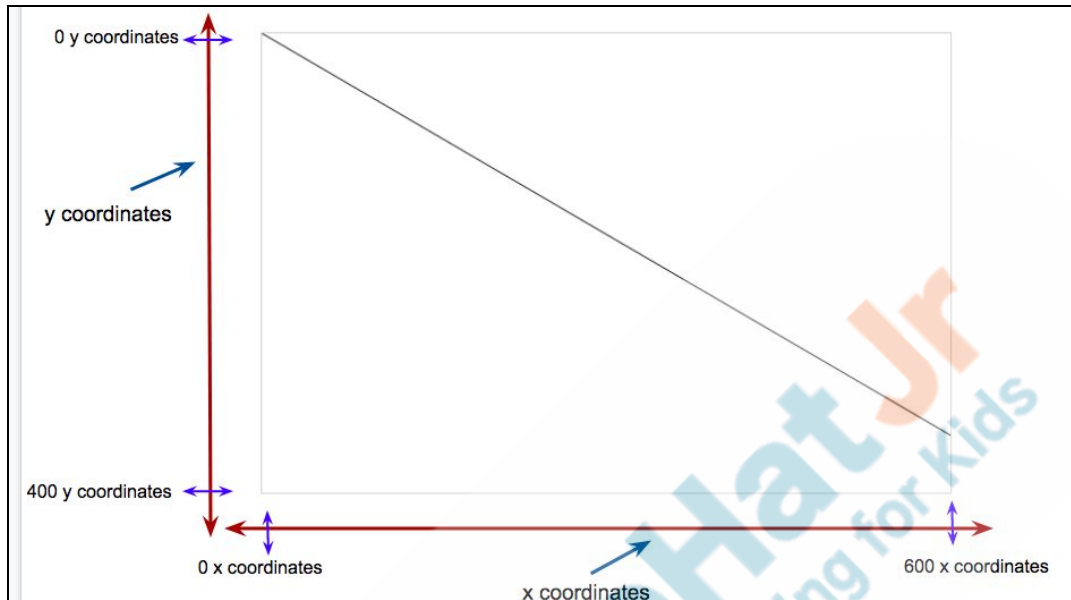
- The above code will check if the **mouseEvent** variable is equal to **mouseDown**, then only the drawing will happen.
 - The value of **mouseEvent** variable will be **mouseDown** only when **mousedown event has occurred** just before this mouse event (which is event mousemove)
 - Now inside this if condition, we know that the mouse is clicked, so we can start drawing, until the **mouseleave**.
- Now we will start drawing.

```
ctx.beginPath();  
ctx.strokeStyle = color;  
ctx.lineWidth = width_of_line;
```

- **beginPath()**: Begins a path, or resets the current path for drawing anything. It tells the canvas to start drawing the shape/object.
- **strokeStyle**: Sets the color for the drawing, we have set the value to variable **color** and variable **color** has black color, so the drawing will be of black color.
- **lineWidth**: Sets width for the drawing, we have set the value to variable **width_of_line** and variable **width_of_line** has 2 value, so the width of the drawing will be 2.

9. Now show [Teacher-Activity-2-Section-1](#) to the student and explain the canvas and

coordinates.



- The x coordinate starts from left of the canvas with value 0, and as we move on towards right the value of x coordinate will increase till the width of the canvas. In this case the width is 600.
- The y coordinate starts from top of the canvas with value 0, and as we move on down the value of y coordinate will increase till the height of the canvas. In this case the height is 400.

10. Now run [Teacher-Activity-5](#) and follow the steps:

ctx will be used as a reference of the canvas for drawing.

- **moveTo()** function:
 - The moveTo() function moves the point to the specified point in the canvas, without creating a line.
 - This can be used when we want to start drawing from a particular point.
- **syntax of moveTo: moveTo(x coordinate, y coordinate)**
- **Change the x coordinates by 100 one by one till 600.**

For example: `ctx.moveTo(100, 0);` then `ctx.moveTo(200, 0);` and so on.

- As you can see, if we increase the value of x coordinate, the creation of the line moves from left to right, because the x coordinate is 0 at the left side and as we move towards right the value of x coordinate increases.

```
ctx.moveTo(0, 0);
```

- **Change the y coordinates by 100 till 500 one by one.**

For example: `ctx.moveTo(0, 100);` then `ctx.moveTo(0, 200);` and so on.

- As you can see, if we increase the value of y coordinate the creation of the line moves from top to bottom, because the y coordinate is 0 at the top and as we move down the value of y coordinate increases.

```
ctx.moveTo(0, 0);
```

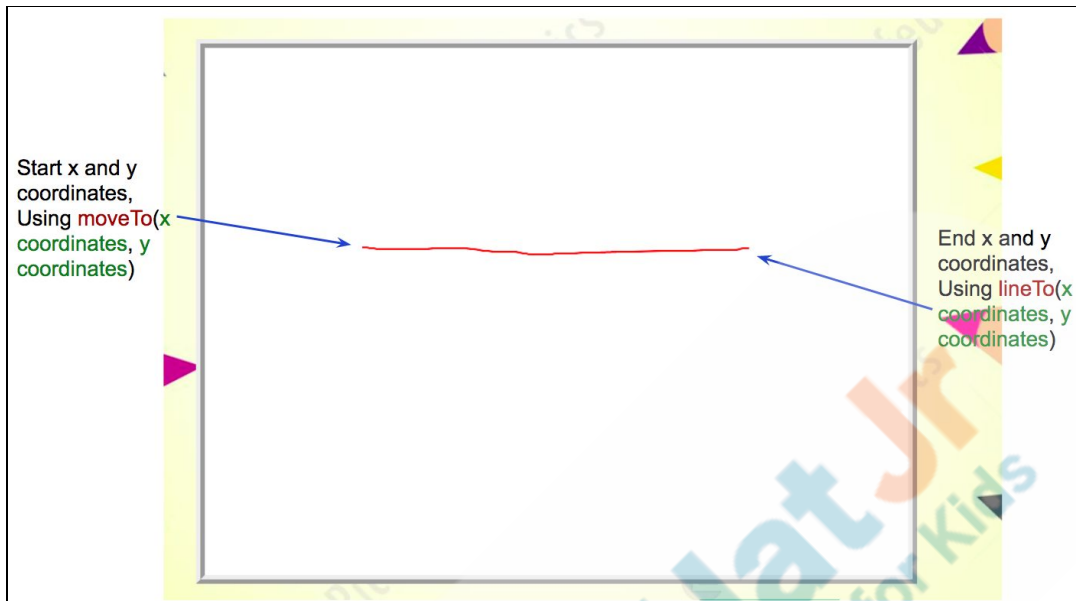
- **lineTo() function:**

- lineTo() function adds a new point and creates a line TO that point FROM the last specified point, which we have discussed in `moveTo()` in the canvas.

- **Syntax of lineTo: lineTo(x coordinate, y coordinate)**

- **Change the x and y coordinates randomly and show the changes.**

- As we change the x and y coordinates, the drawing ends changes at those coordinates. `ctx.lineTo(600, 400);`



- Now console the last x and y coordinate.

```
console.log("Last position of x and y coordinates = ");
console.log("x = " + last_position_of_x + "y = " + last_position_of_y);
```

- Now we will pass this **last_position_of_x** and **last_position_of_y** inside the **moveTo** function.
 - We are doing this because when the mouse moves from one coordinate (let's keep these coordinates as **old_coordinates**) to another coordinate (let's keep these coordinates as **new_coordinates**), we want that a drawing should happen between the **old_coordinates** and the **new_coordinates**.
- So, when we pass the last x and y coordinate to **moveTo** function, this means that the creation of the line should start from these coordinates.

```
ctx.moveTo(last_position_of_x, last_position_of_y);
```

- Now console the current x and y coordinate.

```
console.log("Current position of x and y coordinates = ");
console.log("x = " + current_position_of_mouse_x + "y = " + current_position_of_mouse_y);
```

- View the console screen and see how the console screen updates when

you move the mouse.

- Now we will pass this **current_position_of_mouse_x** and **current_position_of_mouse_y** inside the **lineTo** function.
 - We are doing this because as when we keep moving our mouse the drawing thing should keep happening till these new coordinates.
- So when we pass the current x and y coordinate to **lineTo**, this means that the creation of the line should end on these coordinates.

```
ctx.lineTo(current_position_of_mouse_x, current_position_of_mouse_y);
```

- Now we will draw the line using the **stroke()** function.

```
ctx.stroke();
```

- After this if condition ends. Refer to the complet 'if condition' code below.

```
if (mouseEvent == "mousedown") {  
  ctx.beginPath();  
  ctx.strokeStyle = color;  
  ctx.lineWidth = width_of_line;  
  
  console.log("Last position of x and y coordinates = ");  
  console.log("x = " + last_position_of_x + "y = " + last_position_of_y);  
  ctx.moveTo(last_position_of_x, last_position_of_y);  
  
  console.log("Current position of x and y coordinates = ");  
  console.log("x = " + current_position_of_mouse_x + "y = " + current_position_of_mouse_y);  
  ctx.lineTo(current_position_of_mouse_x, current_position_of_mouse_y);  
  ctx.stroke();  
}
```

- Now, we will put the **current_position_of_mouse_x** inside **last_position_of_x** and **current_position_of_mouse_y** inside **last_position_of_y**.
- This is done because:
 - When the mouse moves from one coordinate (**let's keep these coordinates as old_coordinates**) to another coordinate (**let's keep these coordinates as new_coordinates**), we want that a drawing thing should happen between the **old_coordinates** and the **new_coordinates**.

- To achieve this we need to store the **old_coordinates**. And we store the **old_coordinates** with respect to x and y coordinates as follows:

```
last_position_of_x = current_position_of_mouse_x;
last_position_of_y = current_position_of_mouse_y;
```

- Then these **old_coordinates** are passed to **moveTo()** function so that a drawing can be performed from **old_coordinates** till **new_coordinates**, which are passed to **lineTo()**.
 - In short, **moveTo()** will start the line and **lineTo()** will end the line.

Teacher Stops Screen Share

Now it is your turn.

- Ask Student to press ESC key to come back to panel
- Guide Student to start Screen Share
- Teacher gets into Fullscreen

Step 3: Student-Led Activity (35 mins)

***Note:** Please follow the following sets.

***Note:** Allow the student to copy and paste if the code is repeated.

1. Ask the student to download the folder - [C82CanvasPaint](#) from [Student-Activity-2](#)
2. It has HTML, CSS, JS files and background image.
3. The HTML and CSS code is prewritten as it has been already done in the previous class.
4. Ask the student to open the [C82CanvasPaint](#) folder in Visual Studio, and do a live test, by clicking on the bottom **GO LIVE**.

START CODING



[Student Activity 1 - CODE DIAGRAM](#)


[Student Activity 2- SOURCE CODE](#)

[Student Activity 3- UNDERSTANDING MOUSE EVENT](#)

[Student Activity 4 - UNDERSTANDING moveTo and lineTo FUNCTION](#)

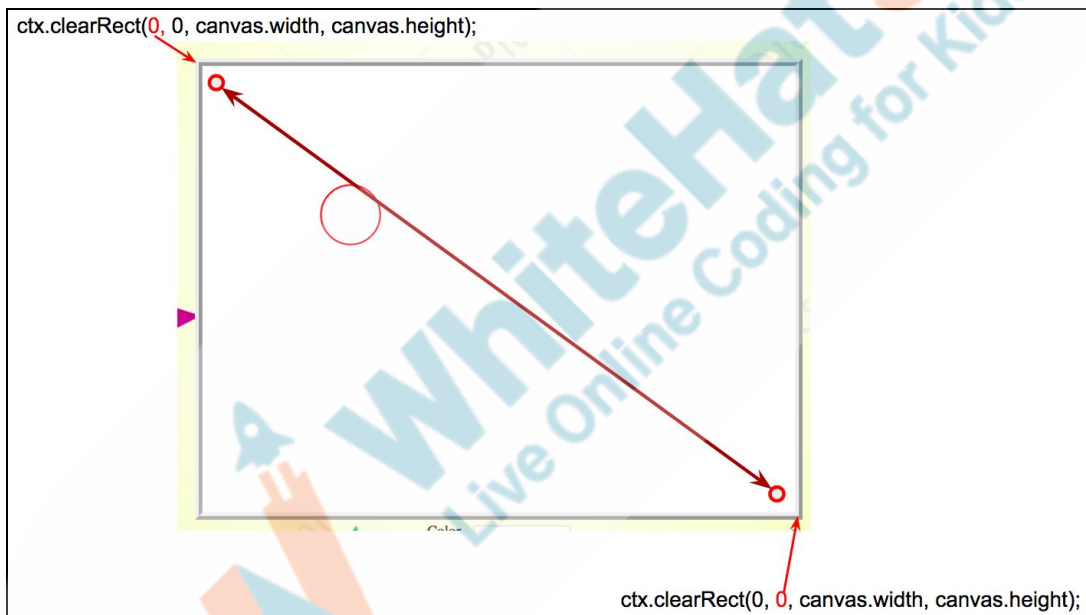
[Student Activity 5- REFERENCE FOR HOSTING A WEBSITE](#)

| | | |
|--|--|--|
| | <p>5. Let the student start coding in main.js</p> <p>6. Now let the student write the code as explained.</p> <p>Now lets upload all the files (index.html, style.css, main.js and all the images used) to host your website on GitHub.</p> <p>The steps to host the website is mentioned in Student-Activity-5, copy the github and paste it in the Submit Link Field on the student panel and click Submit Button.</p> <p>Test your webapp using this link on your Phone.</p> | <p>Encourage and help the student if the student gets stuck.</p> <p>NOTE TEACHERS - First ask the student to host the website, the steps to host the website is mentioned in Student-Activity-5, and copy the github link and paste it in the Submit Link Field on the student panel and click Submit Button.</p> <p>Test the webapp using this link on your Phone.</p> |
| Teacher Guides Student to Stop Screen Share | | |
| <p>Step 4: Wrap-Up (5 mins)</p> | <p>You did great today as well. You get two hats Off.</p> <p>*Note: Ask the student to play around with Student-Activity-3 for learning different MOUSE EVENTS</p> <p>*Note: Ask the student to play around with Student-Activity-4 for learning moveTo and lineTo functions.</p> <p>Q1. What are x and y coordinates on canvas?</p> <p>A. These are the points in pixels ,using these two coordinates</p> | <p>(Give at least 2 hats off) Press the Hats Off Icon for Creatively Solving Activities</p>  <p>Press the Hats Off Icon for Great Question</p>  <p>Press the Hats Off Icon for "Strong Concentration.."</p> |

| | | |
|---|---|---|
| | <p>we can plot anything on canvas</p> <p>Q2. What are mousedown, mouseup, mouseleave and mousemove events?</p> <p>A. These are mouse events which are triggered when the user presses a mouse button over an html element.</p> <p>Q3. How do the moveTo and lineTo functions work ?</p> <p>A. Creation of the line starts from the coordinates specified inside the moveTo() function, and creation of the line ends at the coordinates specified inside the lineTo() function</p> | <p>Strong Concentration  +10</p> |
| <p>Step 5: Project Pointers and Cues (5 min)</p> | <p>PAINT</p> <p>Goal of the Project:</p> <p>Today you extended the applications of the web. You learned to visualize shapes and drawings using the canvas element to elevate the user interaction.</p> <p>In this project, you will have to practice and apply what you have learned in the class. You are making a HTML page with canvas to draw the line using a round circle.</p> <p>Story:</p> <p>Miss Roshni likes to draw free-hand drawings. She is really good at it! Now she wants to create a computer</p> | |

| | | |
|--|--|--|
| | <p>program to draw freehand drawings so everyone can enjoy drawing such pictures. As a first step, she wants the program to draw lines using only circles on canvas in HTML.</p> <p>I am very excited to see your favourite food list and how you make this project, and I know you will do really well.</p> <p>Good Luck!</p> | |
| <div>Teacher Clicks</div> <div>✕ End Class</div> | | |
| Additional Activity | | |
| <p>1. We will add a clean button that is used to clean the canvas.</p> <ul style="list-style-type: none"> HTML code: We are calling clear function on a click of this button. <pre><button onclick="clearArea();">Clear</button></pre> <ul style="list-style-type: none"> JS code: <pre>function clearArea() { ctx.clearRect(0, 0, canvas.width, canvas.height); }</pre> <ul style="list-style-type: none"> Syntax of clear function: ctx.clearRect(x, y, Canvas_width, Canvas_height); <ul style="list-style-type: none"> ctx is the reference of canvas 2d which we created in the beginning. clearRect is a predefined function used to clean the canvas. x: The x-coordinate of the upper-left corner of the rectangle to be cleared. | | |

- **y**: The y-coordinate of the upper-left corner of the rectangle to be cleared.
- **Canvas_width**
- **Canvas_height**
- We have used: **ctx.clearRect(0, 0, Canvas.width, Canvas.height);**
 - **x**: We will set it to 0 because we want to clean the canvas from the top-left.
 - **y**: We will set it to 0 because we want to clean the canvas from the bottom-right.
 - **Canvas.width**: It will get the width of the canvas.
 - **Canvas.height**: It will get the height of the canvas.



2. Now we will take **colour** from the input box, and use this color for drawing, for that we need a input box.

- HTML code: We have an input box with **id = color**.

```
<span>Color - </span> <input type="text" id="color">
```


- JS code: Inside the **mousedown** function we will take the value from the input box using id and store it in **color** variable.

```
function my_mousedown(e)
{
    //Addictonal Activity start
    color = document.getElementById("color").value;
```

3. Now we will take width from the input box and use this width for drawing. For that we need an input box.

- HTML code: We have an input box with id **width_of_line**.

```
<span>Width Of the line - </span>
<input type="text" id="width_of_line">
```

- JS code:

```
function my_mousedown(e)
{
    //Addictonal Activity start
    color = document.getElementById("color").value;
    width_of_line = document.getElementById("width_of_line").value;
    //Addictonal Activity ends
```

- Inside the **mousedown** function we will take the value from the input box using id and store it in **width_of_line** variable.

| Activity | Activity Name | Links |
|--------------------|--|---|
| Teacher Activity 1 | WEBSITE LINK | https://jynyhy9vu5r8xz5zy0utww-on.driv.tw/ADV-C82.com/C82.html |
| Teacher Activity 2 | CODE DIAGRAM | https://docs.google.com/document/d/e/2PACX-1vSkG19gdUODSTXcgUq7CPK0DrmeNwhA3mMALawacL-ZnvGwGutRLrDN o8Dvu_6FLRzwhGqDzDfG3vHu/pub |
| Teacher Activity 3 | SOURCE CODE | https://drive.google.com/drive/folders/1_ZBk0gLFBRO6VNXheF42zpS6LWthQoHY?usp=sharing |
| Teacher Activity 4 | UNDERSTANDING MOUSE EVENT | https://www.w3schools.com/code/tryit.asp?filename=GB9X5ICKPN6L |
| Teacher Activity 5 | UNDERSTANDING moveTo and lineTo FUNCTION | https://www.w3schools.com/code/tryit.asp?filename=GB08V78VDL4U |
| Student Activity 1 | CODE DIAGRAM | https://docs.google.com/document/d/e/2PACX-1vSkG19gdUODSTXcgUq7CPK0DrmeNwhA3mMALawacL-ZnvGwGutRLrDN o8Dvu_6FLRzwhGqDzDfG3vHu/pub |
| Student Activity 2 | SOURCE CODE | https://drive.google.com/drive/folders/1p3boQ1xRcWeGXNwEUBdMxwEKZ0qkhClH?usp=sharing |
| Student Activity 3 | UNDERSTANDING MOUSE EVENT | https://www.w3schools.com/code/tryit.asp?filename=GB9X5ICKPN6L |
| Student Activity 4 | UNDERSTANDING moveTo and lineTo FUNCTION | https://www.w3schools.com/code/tryit.asp?filename=GB08V78VDL4U |

| | | |
|--------------------|---|---|
| Student Activity 5 | REFERENCE FOR HOSTING A WEBSITE ON GITHUB | https://docs.google.com/document/d/e/2PACX-1vSALRoY8p7bVtXNfuKEOG3Bmp5lqkQCnUqpCSMTuHvSCa37BumNEdYgwXPDIC0LZmbPzq8Hg-6frES8/pub |
| Project Solution | PAINT | https://codepen.io/rupin/pen/rNeNBBw |

