CANVAS

What is our GOAL for this MODULE?

We learned about the canvas element. We also learned to draw circles on a canvas.

What did we ACHIEVE in the class TODAY?

- Created a canvas web app to make circles on the canvas.

Which CONCEPTS/ CODING did we cover today?

- Created a canvas element.
- Learned about the background property in CSS.
- Drew circles on the canvas.
- Used event listeners with canvas.
- Got x and y coordinates of the mouse.

How did we DO the activities?

HTML and CSS code:

1. Give a class to the **body** tag.

```
<body class="body_backgorund" >
```

- CSS for the **body** tag:

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

2. Add a **center** tag. Put all the elements used inside this tag.
    - Use the **h1** tag to hold the heading.

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

    - CSS for **h1**:
        - Set the color and font-size.

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

    - Use the **canvas** tag**.** This tag gets the canvas box.
        - Provide **width** and **height** to it.
        - Give an **id** for reference.

    - **CSS for canvas**:
        - Increase the width of the border.
        - Set the background color to white.
        - Use a **border-style:ridge;** to specify a 3D ridged border. The effect depends on the border-color value.

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

    - Use a **br** tag and end the **center** tag.

3. Use the **script** tag to include the **main.js** file.

```
<script src="main.js"></script>
```

JavaScript code:

1. First, get the canvas element using its id and store it in the **canvas** variable so as to manipulate the canvas through JS code.

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

2. Get the reference of the canvas by using **getContext("2d")**.
    - This will be used for drawing things on canvas.
    - Use the **ctx** variable as a reference of canvas for drawing.

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

3. Set the **color** variable to apply color to the circle (or any other shape).

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

4. Code for drawing a default circle when you run the file:

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

    - **ctx** is used to refer to the canvas which we made.
    - In **ctx.anything,** 'anything' can be any method used in canvas. Here are a few methods:
        - **beginPath()**: Begins a path, or resets the current path for drawing. It tells the canvas to start drawing the shape/object.
        - **strokeStyle**: Sets the color for the drawing object.
            - Set this to the **color** variable. The **color** variable holds **red**. This means the color of the circle will be red.
        - **lineWidth**: Sets width for the drawing object.
        - **arc**: Creates an arc/curve (used to create a circle or part of a circle).

Explaining 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.

For better understanding run https://mahdihat791.github.io/arc_learn/

5. Now add **addEventListener.**
   - This is attached with an element, and when the event [**onclick, mousemove, etc**] occurs it runs the function written inside it.
   - **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 similar to the **AddListener** block we used in our Chatapp. We used this block to monitor if the chat message was sent to the firebase.
     - **event -** It can be any event.
       For example: **click**, **mousemove**, **mousedown** (means when the mouse is clicked)
     - **my_function**: It will be any function we define. When this event occurs the function defined by us gets executed.
     - **my_function(e):** It will be the function we define. This function will perform certain tasks that are written inside it.
       - **e** means the event of the function.
       - This **e** has a relation with the event
         For example: If the event is **mousedown**, then this **e** has a relation with the **mousedown** event.

   - **Example of event listener:**
     - First, get the button which is the HTML element. Put it inside the **button** variable.
     - Attach the button variable to **addEventListener.**

- Define the type of event. Here we have defined **click event.**
- Then call our **my_function.**
- Define our **my_function**, which will be the code for consoling "**I am button**" on the console screen.

- When the button is clicked. The function will be executed and it will print **"I am button"** in the console.

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

6. Now attach **canvas** and **addEventListener**.

```
function my_mousedown(e)
{

    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);
}
```


- First, attach a **canvas** element to **addEventListener** and set the event as **mousedown**. Then call the **my_mousedown** function. **mousedown** event means when the mouse is clicked.

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

- Now define the function **my_mousedown.**
  - **e** is the event variable of the **mousedown** event.

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

- Now get **x coordinates** when the mouse is clicked.
    - **e** indicates the event in which it is written. In this case it is written inside a **mousedown** event.
    - **e.clientX** gives 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. Use **e.clientX - canvas.offsetLeft** to get the actual x coordinate on canvas with respect to the screen when clicked**.**
- After getting the x coordinate, store it in the **mouse_x** variable.

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

- Now get y coordinates when the mouse is clicked.
    - **e.clientY:** gives 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. Use **e.clienty - canvas.offsetTop** to get the actual y coordinate on canvas with respect to the screen when clicked.
- After getting the y coordinate, store it in the **mouse_y** variable.

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

- Then console these coordinates to see in numbers.

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

- Output:

- After getting these coordinates, pass these coordinates to the **circle function**. Pass the x and y coordinates of the mouse to the **circle()** function to draw the circle at that point.

```
circle(mouse_x , mouse_y);
```

7. Circle function:

```
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();
}
```

- Define the **circle** function with **mouse_x** and **mouse_y** variables to build the circle where we have clicked.

```
function circle(mouse_x , mouse_y)
```

- **ctx** is used to refer to the canvas which we made.
- In **ctx.anything,** 'anything' can be any method used in canvas. Here are a few methods:
    - **beginPath()**: Begins a path, or resets the current path for drawing. It tells the canvas to start drawing the shape/object.
    - **strokeStyle**: Sets the color for the drawing object.
        - Set this to the **color** variable. The **color** variable holds **red**. This means the color of the circle will be red.
    - **lineWidth**: Sets width for the drawing object.
    - **arc**: Creates an arc/curve (used to create a circle or part of a circle).

Explaining arc:

- **ctx.arc(mouse_x, mouse_y, 40 ,0 , 2*Math.PI);**
    - **mouse_x** will be the x coordinate on the mouse where we click.
    - **mouse_y** will be the y coordinate on the mouse where we click.
    - **40** is the radius.
    - **0** is the start angle.
    - **2 * Math.PI** will be the end angle of the circle.
    - **stroke():** this draws the path that you define.

What's NEXT?

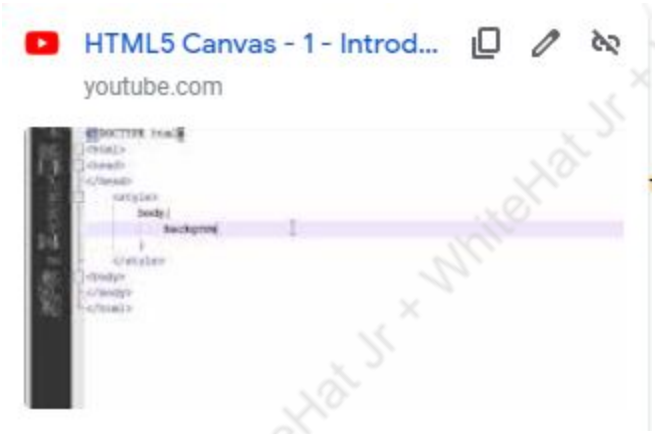
We will create a paint web application using canvas.

EXTEND YOUR KNOWLEDGE

Here are some Best References we've compiled together to enhance your knowledge and understanding of the concepts we learned today in the class. This will help you become a pro at coding and creating industry-grade tech products!

Short Videos: Watch these Short Videos to understand the application of the concepts learned in class in real-world applications.

1. https://www.youtube.com/watch?v=JM9_xJP8mNQ



2. https://www.youtube.com/watch?v=82njuSr6qv8

3. https://www.youtube.com/watch?v=qgCvIOM_o-c

Coding Playground: Try out these code examples to get more practice in making Websites and Playstore ready apps.

1. https://www.w3schools.com/html/html5_canvas.asp

2. https://www.w3schools.com/tags/canvas_arc.asp

3. https://www.geeksforgeeks.org/javascript-math-pi-property/