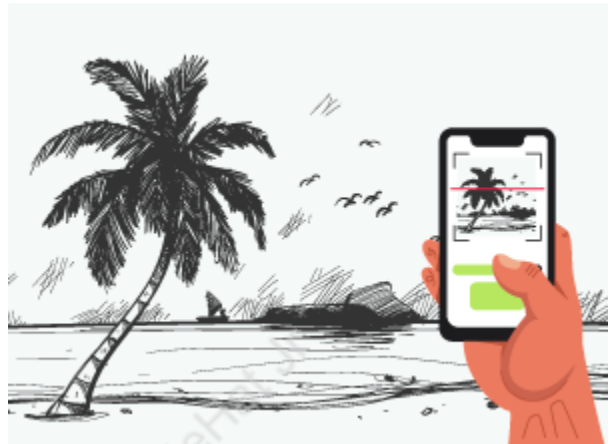


SKETCH IDENTIFICATION



What was our GOAL for this MODULE?

The goal was to venture into the sea of pretrained neural networks and study them for their possible applications of interest. Learning the neural network methods builds confidence to sift through complex code and apply only that which is needed to solve a problem. Handling complexity is essential in AI.

What did we ACHIEVE in the class TODAY?

- Added style to the HTML elements.
- Started JavaScript coding.

Which CONCEPTS/ CODING did we cover today?

- We coded in JS to add a canvas and position it to the center.
- We added style to the header.
- We added style to the label and confidence text on the webpage.
- We added animation to the canvas.

How did we DO the activities?

Complete CSS code:

```
body
{
background-color: #fbd139;
}

h1
{
color: rgb(38, 0, 255);
background-color: cornsilk;
}

#show_sketch
{
float: right;
}
```

```
canvas
{
  border:5px solid ■ white;
  border-radius:20px;
  animation-name: example;
  animation-duration: 4s;
  margin-top: 50px;
  animation-iteration-count: infinite;
}
@keyframes example {
  from {box-shadow: 1px 1px 38px ■ grey;}
  to {box-shadow: 1px 1px 38px ■ white;}
}

p
{
  padding-top:10px;
  font-size:25px;
}
```

JS Code:

```
function setup() {  
  canvas = createCanvas(280, 280);  
  canvas.center();  
  background("white");  
}  
  
function clearCanvas() {  
  background("white");  
}
```

We started adding style to the elements we have created in the html file:

- 1) **We set the background color of the web app:** We started by adding the background color to the webpage. For this, we needed to add code to set the **background-color** of the webpage. We have already defined the **body** tag in the html file in the previous class. We accessed the **body** tag which looks like this-

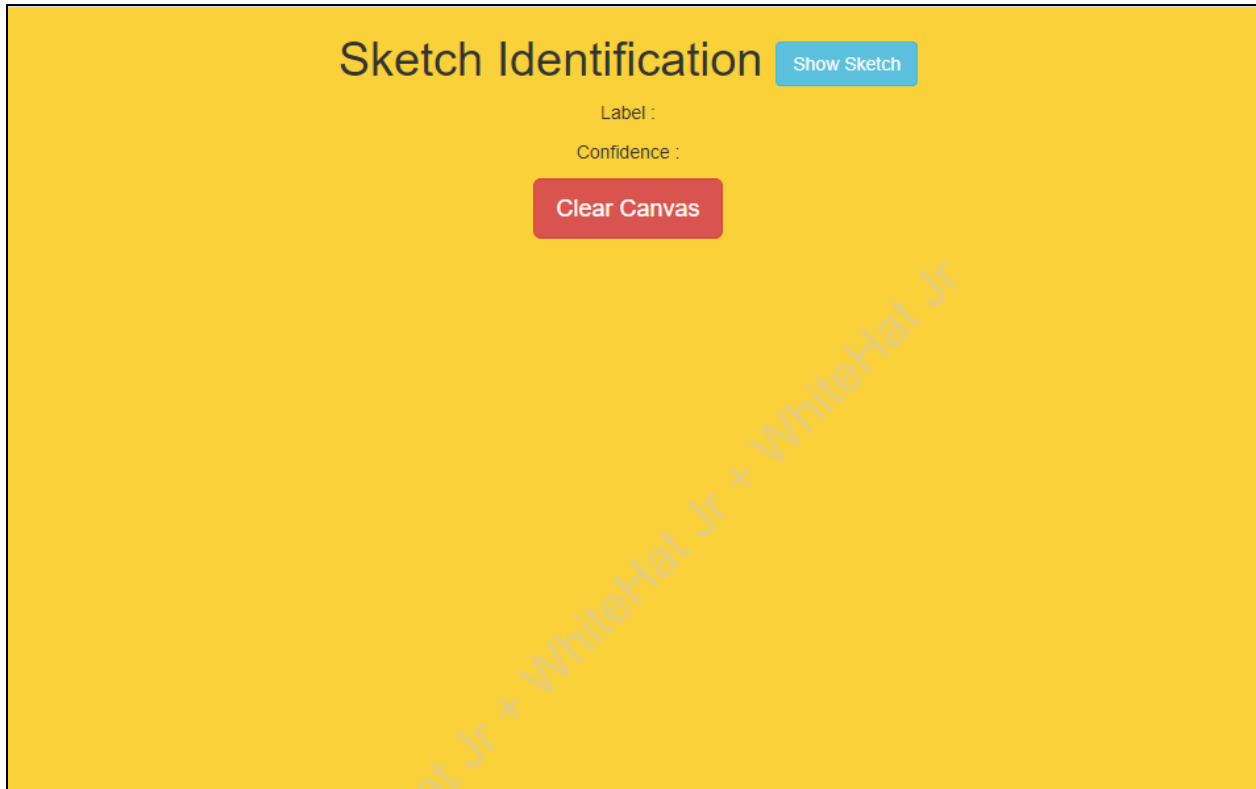
```
body  
{  
  
}
```

And inside this tag, we needed to add the code to set the background color of the web page and we added the yellow color to the webpage, like this:

```
body  
{  
  background-color: #fbd139;  
}
```

You can add any color you want, to set as the background color.

So the output should look like this:




- 2) **We added color to the heading:** Now, we added color to the heading which is the 'Sketch Identification' text. In the html file, we already added the heading using the **h1** tag. Now, we set the font color and the background color of the heading.

So, the code would look like this:

```
h1
{
  color: ■ rgb(38, 0, 255);
  background-color: ■ cornsilk;
}
```


Here, we had to access the **h1** tag which looked like this:

```
h1
{
}
```

And inside these curly brackets, we need to set the color of the font using the **color** property and select the color like this: `color:  rgb(38, 0, 255);`

We have set the color by setting the **rgb** values. You can do it by setting the HEX value OR giving the color a name directly. You can add any color you want.

Then, in the next line, we added the background color for the heading, and for that, we need to use the **background-color** property.

```
background-color:  cornsilk;
```

So, we achieved the following output:

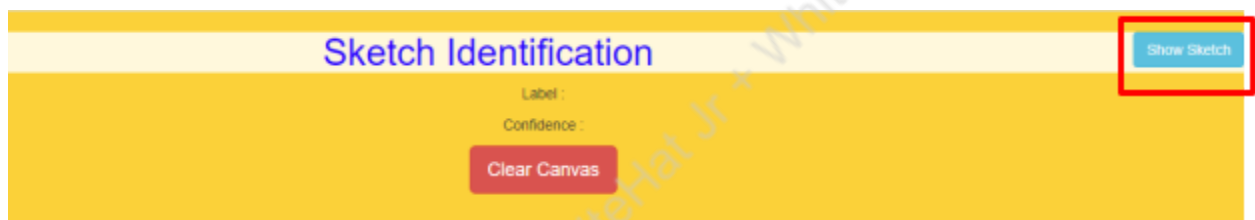


- 3) **Align the button 'Show Sketch':** Then, we aligned the button named **Show Sketch**, to the right side of the web page. For this, we needed to add the code below:

```
#show_sketch
{
  float:right;
}
```

Now, we have created the button with the **id** as **show_sketch**. While styling it using the CSS code, we accessed it by adding the **#** (hash) sign before the **id** name.

And inside the curly brackets, we used the **float** property of the **show_sketch** button element. This basically sets the alignment. Then we set the alignment to the right side of the webpage, so that the button **Show Sketch** appears at the extreme right of the web page. The output would look like this:



- 4) **Styled the <p> tag:** It's time for us to style the **<p>** tag. In the previous class, we added 2 **<p>** tags - one for holding the label and another one for holding the confidence. Now, we added style using the **padding-top** property and set the value to **10** pixels. And we also set the **font-size** property and set its value to **25** pixels.

```
p
{
  padding-top:10px;
  font-size:25px;
}
```

- 5) **Created a canvas:** Then, we created a canvas element in the **main.js** file. For this, we had to add the code in the **main.js** file provided to you, in the previous class.

```
function setup() {  
  canvas = createCanvas(280, 280);  
  canvas.center();  
  background("white");  
}
```

So, here, we created a function named **setup()**, and inside the curly brackets of it, we need to write code to create a canvas:

- We used the **createCanvas()** function and pass the **height** and **width** as **280** and **280** to declare parameters to this function, as shown below:

```
canvas = createCanvas(280, 280);
```

- Then, we placed the canvas at the center. For this, we used the **center()** function of the canvas object.

```
canvas.center();
```

- Then, we set the background color of the canvas in **white**. For this, we used the **background()** function, and inside the function, we set the color to **white**, like this:

```
background("white");
```

- 6) **Cleared the canvas:** As we had to create sketches on the canvas, we want to make another sketch after one sketch is completed. For this, we erased the previous sketch and empty the canvas, so that the canvas becomes ready for another sketch. We used a function for this which is named as the **clearCanvas()** function. We needed to call this function when the **Clear Canvas** button is clicked:


```
function clearCanvas() {
  background("white");
}
```

```
function clearCanvas() {
}
```

- So, we wrote the function as
- And inside it, we added the background color to the canvas, which means, after clearing the output, the background of the canvas is again set to **white**.

```
background("white");
```

- 7) **Add style to the canvas:** Now, we added style to the canvas. So, we again wrote code in the **style.css** file:

```
canvas
{
  border:5px solid white;
  border-radius:20px;
  animation-name: example;
  animation-duration: 4s;
  margin-top: 50px;
  animation-iteration-count: infinite;
}
@keyframes example {
  from {box-shadow: 1px 1px 38px grey;}
  to {box-shadow: 1px 1px 38px white;}
}
```

- We first accessed the canvas element and added a **border** to it, like this:

```
border:5px solid white;
```

We set the **border-width** to **5** pixels, **border-style** to **solid**, and the **border-color** to **white**.

→ Then, we made the corners of the canvas round. For this, we need to set the **border-radius** property to **20px**, like this:

```
border-radius: 20px;
```

→ Then, we added some animation to this canvas:

- **Animation-name: example;**

Here, we defined the **animation-name** as **example**.

```
animation-name: example;
```

In the next point, we defined this **example** where we added code of what the animation would be.

- **animation-duration: 4s;**

Here, we defined the time of animation - that is how long one animation should run.

```
animation-duration: 4s;
```

- **Animation-iteration-count: infinite;**

Here, we wanted the animation to run continuously. So, we need to specify the iteration count to be **infinite**.

```
animation-iteration-count: infinite;
```

→ Also, we needed to set the margin at the top to **50px**.

```
margin-top: 50px;
```

If your canvas appears above the **Clear Canvas** button, then increase the value of **margin-top**, so that it becomes possible to place the canvas below the **Clear Canvas** button.

→ Now, we added the box shadow to the canvas in the following way:

- **@keyframes example {}**
 - **@keyframes** is a keyword which is used to define an animation.
 - **example** is the name of the animation which we have called in the above point.
 - **from {box-shadow : 1px 1px 38px grey;}**
 - Here, we set the box shadow to grey when the animation starts.
 - **to {box-shadow: 1px 1px 38px white;}**
 - Now, we want the box shadow property to change from **grey** to **white**. So, we need to specify **white** here.
- Here, the first **1px** means - how much towards the right hand side of the canvas you want the color.
 - The second **1px** means - how much towards the bottom you want the color.
 - **38px** means - how extensive you want the color to spread.

```
@keyframes example {  
  from {box-shadow: 1px 1px 38px grey;}  
  to {box-shadow: 1px 1px 38px white;}  
}
```

Basically, this animation works continuously by changing the box shadow from **grey** to **white**.

We have used grey and white color here, but you can use any color you want.

OUTPUT:

Sketch Identification

Label :

Confidence :

Clear Canvas



What's NEXT?

In the next class, we will be completing the sketch identification webapp by adding JavaScript code for getting the sketch from canvas and comparing it with the doodlenet model and getting the result of it.

WhiteHat Jr + WhiteHat Jr + WhiteHat Jr