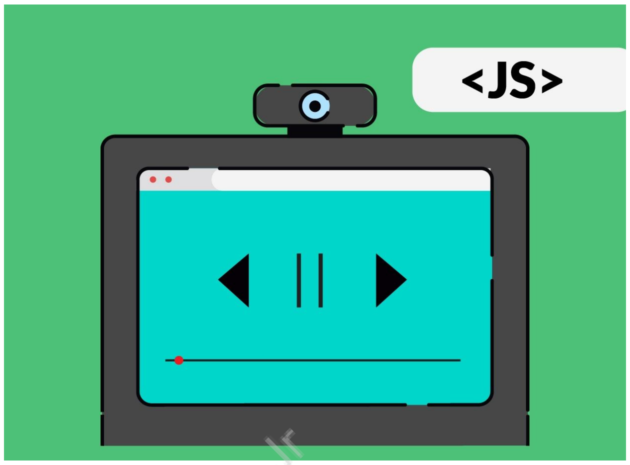
ADV-C128

PRETRAINED NET MODELS



What is our GOAL for this MODULE?

We learned how to manipulate the volume and the speed of the song by the movements of our body parts.

What did we ACHIEVE in the class TODAY?

- We started JS coding for the Web App.

Which CONCEPTS/ CODING did we cover today?

- Added setVolume() and rate() function.
- Added code for initializing posenet model
- Added code for modelLoaded() function
- Added code for executing posenet
- Added code for gotPoses() function
- Added code for fetching x and y coordinates of leftWrist and rightWrist

How did we DO the activities?

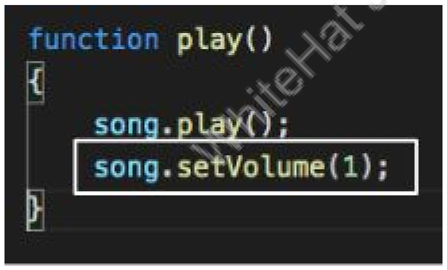
1. Add a code where we can set the volume for the song. For which we'll have to use **setVolume()** function (predefined function) of p5.js which would be used to set the volume for the song.

Syntax of **setVolume():**
variableName.setVolume(volume value):

- **variableName -** This variable will be holding the audio file
- **setVolume** - this is the predefined function name of p5.js
- **volume value** - means at what volume you want to play the song
- this takes a value between 0.0 (silence) and 1.0 (full volume):
    - Means **variableName.setVolume(0.1) -** Very low
    - Means **variableName.setVolume(0.3) -** Little low
    - Means **variableName.setVolume(0.5) -** Medium
    - Means **variableName.setVolume(0.7) -** Little high
    - Means **variableName.setVolume(0.9) -** High
    - Means **variableName.setVolume(1) -** Full volume

So this code will set the volume as full, and in the next class learn how to change the volume as per the movement of the left hand wrist.

This code will be added inside the **play()** function. Because the volume should be set after you play the audio. We are playing the audio inside play() function.



```
function play()
{
    song.play();
    song.setVolume(1);
}
```

2. Add a code to set the speed of the song. For that use **rate()** function which is a predefined function of p5.js to control the speed.

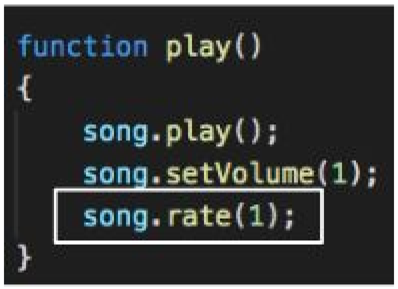
Syntax of **rate():**
variableName.rate(playback rate):

- **variableName -** This variable will be holding the audio file
- **rate** - this is predefined function name of p5.js
- **playback rate** - means at what speed you want to play the song.
- Set the playback rate. 1.0 is normal, .5 is half-speed, 2.0 is twice as fast.
    - Means **variableName.rate(0.5) -** Very slow

- Means **variableName.rate(1) -** Normal
- Means **variableName.rate(1.5) -** Little fast
- Means **variableName.rate(2) -** Twice as fast
- Means **variableName.rate(2.5) -** Very fast

So this code will set the speed as normal, and in the next class learn how to change the speed as the movement of the right hand wrist.

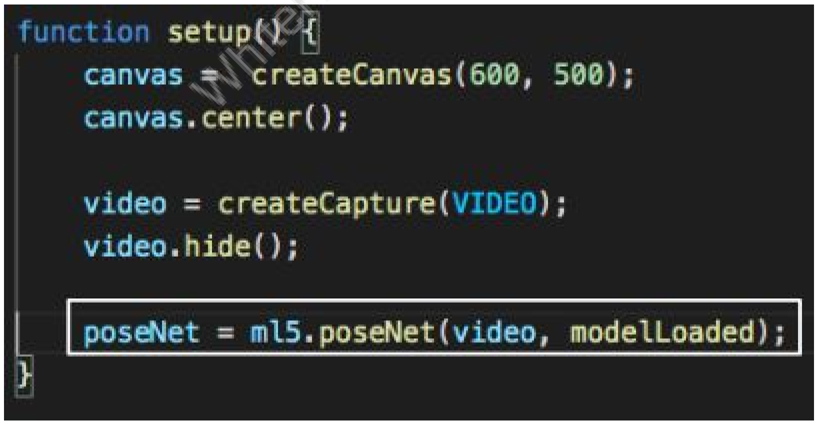
This code will be added inside a **play()** function. Because the speed should be set after you play the audio. We are playing the audio inside play() function.



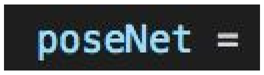
```
function play()
{
    song.play();
    song.setVolume(1);
    song.rate(1);
}
```

We have created a full music web app, where we are playing a song with full volume and normal speed. Now when we want to make this web app dynamic, it means changing the speed and volume of the song by the movements of the left and right wrist respectively. For that we need to write a code for initializing and executing the posenet model.

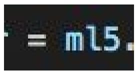
3. Write code for initializing the posenet model.



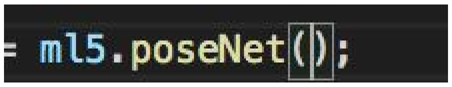
```
function setup() {
    canvas = createCanvas(600, 500);
    canvas.center();

    video = createCapture(VIDEO);
    video.hide();

    poseNet = ml5.poseNet(video, modelLoaded);
}
```

- Define a variable 

```
poseNet =
```

- Write the library name ml5.js 

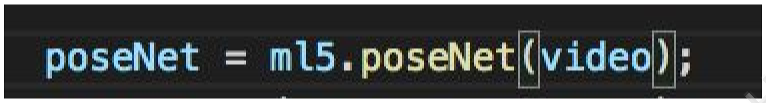
```
= ml5.
```

- Then write poseNet, like this - 

```
= ml5.poseNet();
```

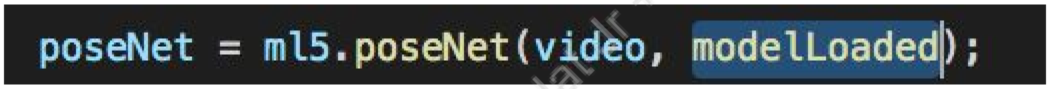
  - **poseNet** is a predefined function of ml5.js. And is used to **initialize the posenet model.**
  - The above line of code will initialize **poseNet() function.** Typically for calling/initializing any ml5.js function we use this way only - **ml5.js.functionName().**

4. Now inside this **posenet()** function we need to pass two parameters:
   - **1st** is the input(means the image OR the video on which we want postnet to take actions) in our case it is - the live view of the webcam.



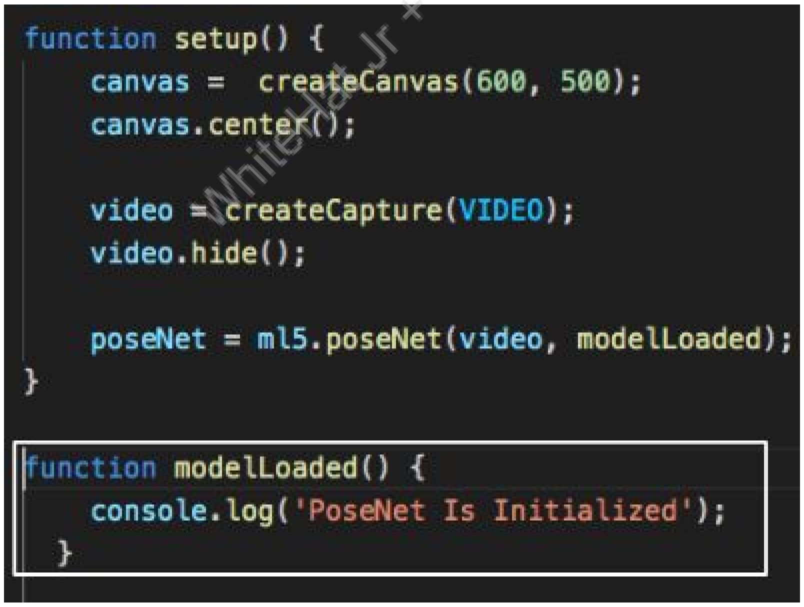
```
poseNet = ml5.poseNet(video);
```

   - **2nd** is the function to confirm that the PoseNet is initialized.



```
poseNet = ml5.poseNet(video, modelLoaded);
```

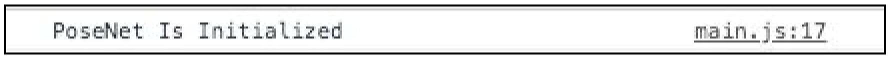
5. Write code of modelLoaded() function outside the setup() function.



```
function setup() {
    canvas =  createCanvas(600, 500);
    canvas.center();

    video = createCapture(VIDEO);
    video.hide();

    poseNet = ml5.poseNet(video, modelLoaded);
}

function modelLoaded() {
    console.log('PoseNet Is Initialized');
  }
```

   - **modelLoaded()** function will be used to show the status of PoseNet (i.e. PoseNet is initialized) on console.

- Output on console Screen:

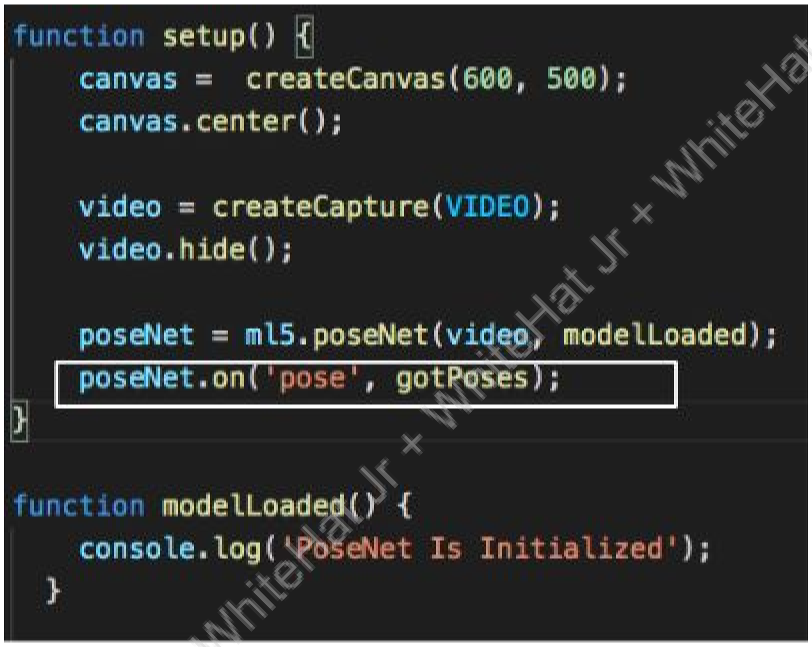


```
PoseNet Is Initialized                    main.js:17
```

Now poseNet is initialized, also we have given the input (passed the live webcam view on which we want posenet() function to take actions) to posenet() function.

6. Write the code to execute the posenet , and get the results.

So now write code for executing posenet, and this code will come inside the **setup()** function.



```
function setup() {
    canvas =  createCanvas(600, 500);
    canvas.center();

    video = createCapture(VIDEO);
    video.hide();

    poseNet = ml5.poseNet(video, modelLoaded);
    poseNet.on('pose', gotPoses);
}

function modelLoaded() {
    console.log('PoseNet Is Initialized');
  }
```

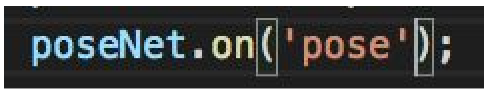
- Write the variable which is holding the posnet initialization which is -

```
poseNet
```

- Write **on()** function, which is a predefined function of ml5.js used to start executing posenet -

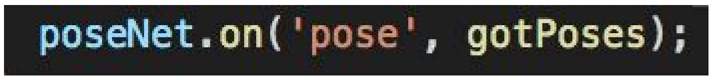
```
poseNet.on();
```

- Then inside these we need to pass 2 parameters.
    - We need to get the **pose**(x and y coordinates of the 17 parts which we discussed in the previous class). So in the **1st** parameter by default write **pose** in single quotes:



```
poseNet.on('pose');
```

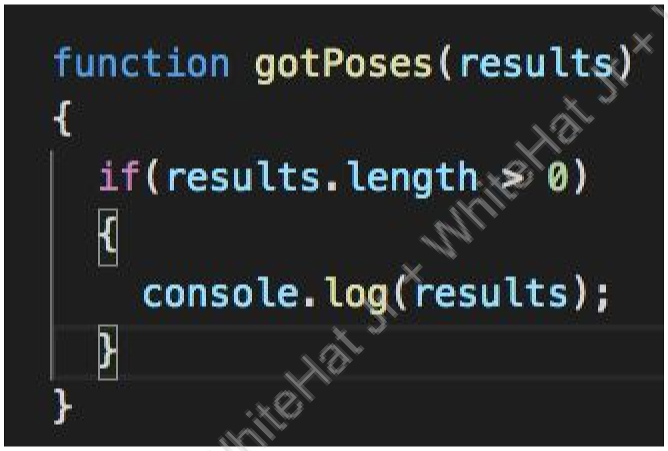
- In the 2nd parameter will pass a function "gotPoses" which will get all the poses(x and y coordinates of the 17 parts which we discussed in the previous class) from the model.



```
poseNet.on('pose', gotPoses);
```

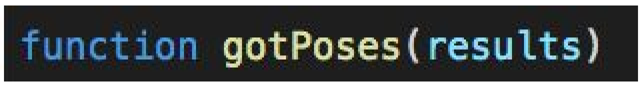
- Define `gotPoses` function in the next step.

7. Write the JS code for **gotPoses()** function.



```
function gotPoses(results)
{
  if(results.length > 0)
  {
    console.log(results);
  }
}
```

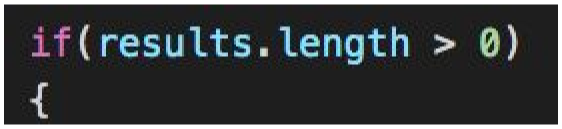
- First define the **gotPoses() function.**



```
function gotPoses(results)
```

  - And inside a **gotPoses() function** write results.
  - Results contain the x and y coordinates of all 17 body parts.

- Write all our code inside an if condition. So first define an if condition inside the **gotPoses() function**, that will check if the length of results is greater than 0 then only go inside “if” condition, it means - **If the results is empty then nothing will happen.**



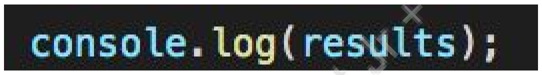
```
if(results.length > 0)
{
```

Situations:

- The webcam doesn’t start due to some reason.
- Or there is no one in front of the webcam, meaning posenet won’t be able to detect anyone.
- Or the posenet starts before the webcam, which will also result in the same which is the posenet won’t be able to detect anyone.
- Or there are any other errors.

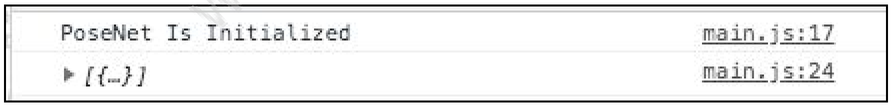
IF we don’t use this ‘if’ condition and any of the above situations occur then posenet will stop and our web app will give an error.

- Now inside the if condition console the results. As results array will have all the data coming as a result from posenet which is the x and y coordinates of all 17 body parts.



```
console.log(results);
```

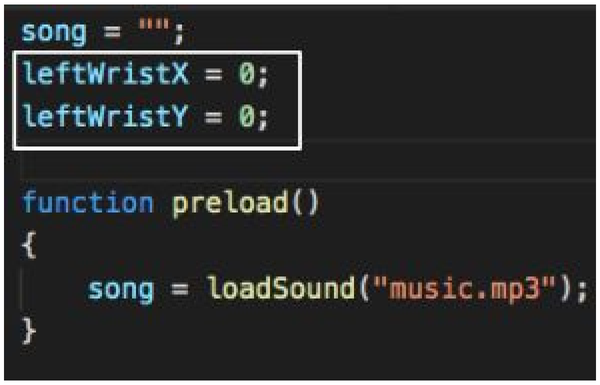
- Output on console screen:



```
PoseNet Is Initialized                                   main.js:17
▸ [{…}]                                                  main.js:24
```

- The output on the console screen will be very fast as the code is running continuously in real time.
- And then we can read this result array, and fetch the x and y coordinates of leftWrist and rightWrist.

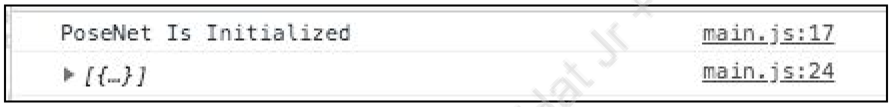
8. So first define 2 variables for holding x and y coordinates of the **leftWrist** and initialize with a value of 0, define these variables at the beginning of the **main.js** file.



```
song = "";
leftWristX = 0;
leftWristY = 0;

function preload()
{
    song = loadSound("music.mp3");
}
```

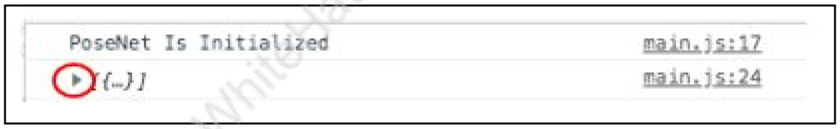
Now run https://mahdihat791.github.io/Ai-DJ/ and open the console screen.

Output on console screen:



```
PoseNet Is Initialized                     main.js:17
▶ [{…}]                                    main.js:24
```

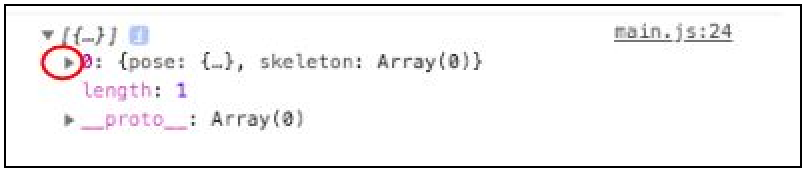
9. We need to read this object and fetch the x and y coordinates of leftWrist, while reading the object write the code:
   - First click on the arrow to expand:



```
PoseNet Is Initialized                     main.js:17
▶ [{…}]                                    main.js:24
```

   - We want to read the objects of **results** so first write `results`

10. Then click on the arrow next to `0:` to expand:



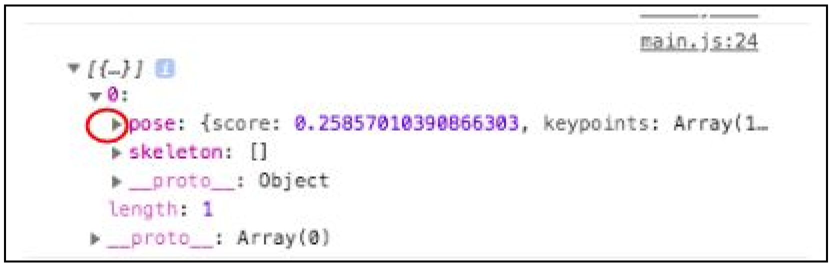
```
▼ [{…}]                                    main.js:24
  ▶ 0: {pose: {…}, skeleton: Array(0)}
    length: 1
  ▶ __proto__: Array(0)
```

© 2019 The content of this email is confidential and intended for the recipient specified in message only. It is strictly forbidden to share any part of this message with any third party without a written consent of the sender. If you received this message by mistake, please reply to this message and follow with its deletion, so that we can ensure such a mistake does not occur in the future.

- We have clicked on 0 index which is inside the "results" object, so code will be

```
results[0]
```

11. Then click on the arrow next to `pose:` to expand:



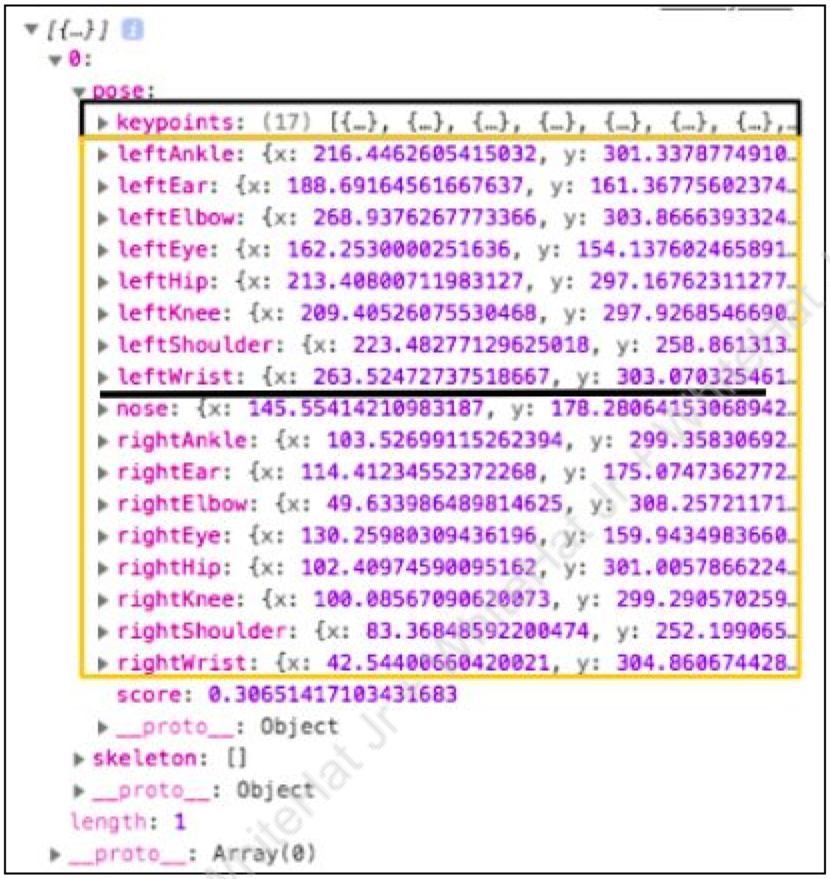
```
                                                     main.js:24
▼[{…}] 
  ▼0:
    ▶pose: {score: 0.25857010390866303, keypoints: Array(1…
    ▶skeleton: []
    ▶__proto__: Object
   length: 1
  ▶__proto__: Array(0)
```

- Then inside 0 index we have clicked on pose object, so code will be -

```
results[0].pose
```

This much of code will remain the same every time when we want to access the x and y coordinates of any body parts.

12. Then inside the **pose** object there are the two important parts `keypoints:` and 17 body parts with x and y coordinates. `keypoints:` has the same thing which is 17 body parts with x and y coordinates. So we don't need to go in `keypoints:`, because we will find all the x and y coordinates of 17 body parts outside only:

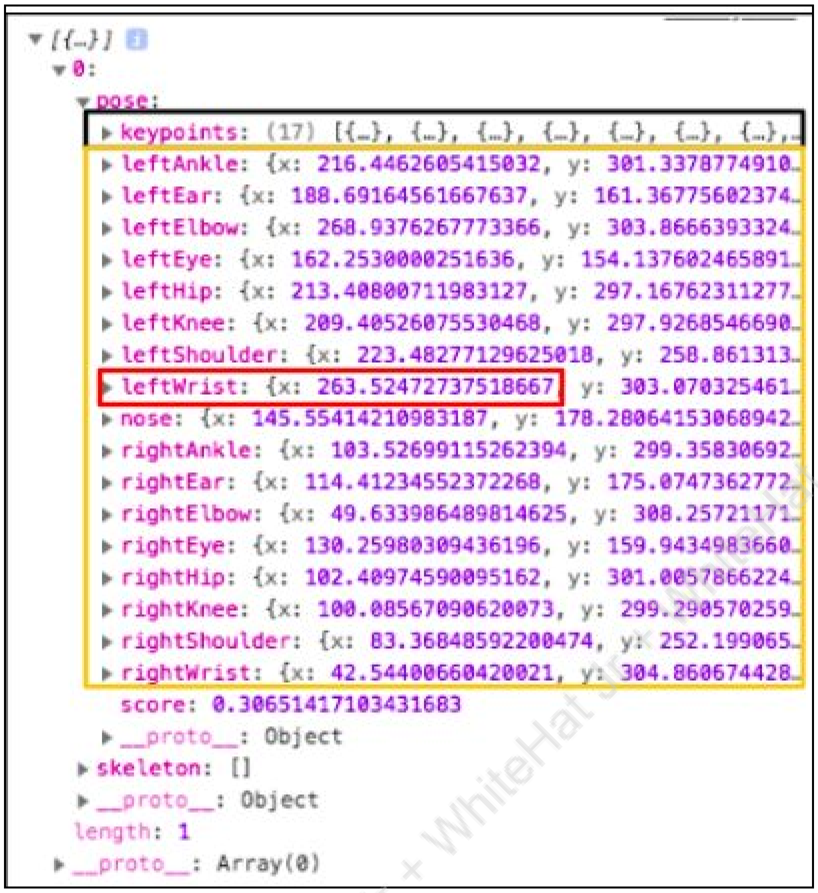


```
▼ [{…}]
  ▼ 0:
    ▼ pose:
      ▶ keypoints: (17) [{…}, {…}, {…}, {…}, {…}, {…}, {…},…
      ▶ leftAnkle: {x: 216.4462605415032, y: 301.3378774910…
      ▶ leftEar: {x: 188.69164561667637, y: 161.36775602374…
      ▶ leftElbow: {x: 268.9376267773366, y: 303.8666393324…
      ▶ leftEye: {x: 162.2530000251636, y: 154.137602465891…
      ▶ leftHip: {x: 213.40800711983127, y: 297.16762311277…
      ▶ leftKnee: {x: 209.40526075530468, y: 297.9268546690…
      ▶ leftShoulder: {x: 223.48277129625018, y: 258.861313…
      ▶ leftWrist: {x: 263.52472737518667, y: 303.070325461…
      ▶ nose: {x: 145.55414210983187, y: 178.28064153068942…
      ▶ rightAnkle: {x: 103.52699115262394, y: 299.35830692…
      ▶ rightEar: {x: 114.41234552372268, y: 175.0747362772…
      ▶ rightElbow: {x: 49.633986489814625, y: 308.25721171…
      ▶ rightEye: {x: 130.25980309436196, y: 159.9434983660…
      ▶ rightHip: {x: 102.40974590095162, y: 301.0057866224…
      ▶ rightKnee: {x: 100.08567090620873, y: 299.290570259…
      ▶ rightShoulder: {x: 83.36848592200474, y: 252.199065…
      ▶ rightWrist: {x: 42.54400660420021, y: 304.860674428…
        score: 0.30651417103431683
      ▶ __proto__: Object
    ▶ skeleton: []
    ▶ __proto__: Object
    length: 1
  ▶ __proto__: Array(0)
```

- If we want the coordinates of the leftWrist, so **inside results -> inside 0 index -> inside pose -> there is a leftWrist**. Write -

```
results[0].pose.leftWrist
```

13. Inside leftWrist we want x coordinate:

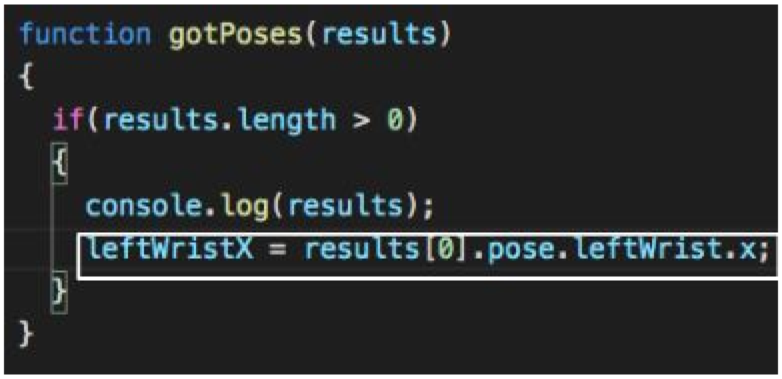


```
▼[{…}] 
 ▼0:
  ▼pose:
   ▶keypoints: (17) [{…}, {…}, {…}, {…}, {…}, {…}, {…},…
   ▶leftAnkle: {x: 216.4462605415032, y: 301.3378774910…
   ▶leftEar: {x: 188.69164561667637, y: 161.36775602374…
   ▶leftElbow: {x: 268.9376267773366, y: 303.8666393324…
   ▶leftEye: {x: 162.2530000251636, y: 154.137602465891…
   ▶leftHip: {x: 213.40800711983127, y: 297.16762311277…
   ▶leftKnee: {x: 209.40526075530468, y: 297.9268546690…
   ▶leftShoulder: {x: 223.48277129625018, y: 258.861313…
   ▶leftWrist: {x: 263.52472737518667, y: 303.070325461…
   ▶nose: {x: 145.55414210983187, y: 178.28064153068942…
   ▶rightAnkle: {x: 103.52699115262394, y: 299.35830692…
   ▶rightEar: {x: 114.41234552372268, y: 175.0747362772…
   ▶rightElbow: {x: 49.633986489814625, y: 308.25721171…
   ▶rightEye: {x: 130.25980309436196, y: 159.9434983660…
   ▶rightHip: {x: 102.40974590095162, y: 301.0057866224…
   ▶rightKnee: {x: 100.08567090620073, y: 299.290570259…
   ▶rightShoulder: {x: 83.36848592200474, y: 252.199065…
   ▶rightWrist: {x: 42.54400660420021, y: 304.860674428…
    score: 0.30651417103431683
   ▶__proto__: Object
  ▶skeleton: []
  ▶__proto__: Object
  length: 1
 ▶__proto__: Array(0)
```

- So the code will be - **inside results -> inside 0 index -> inside pose -> inside the leftWrist -> x coordinate.** Code -

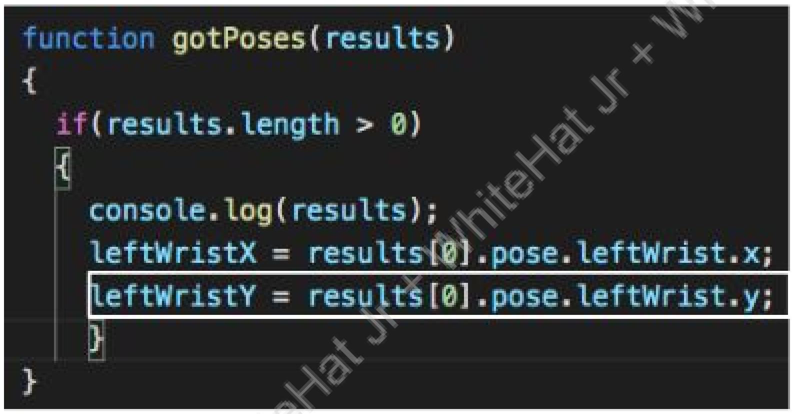
```
results[0].pose.leftWrist.x;
```

14. Now we have fetched the code for getting the x coordinate of leftWrist, so update the leftWristX variable with this code, inside the **gotPoses()** function:



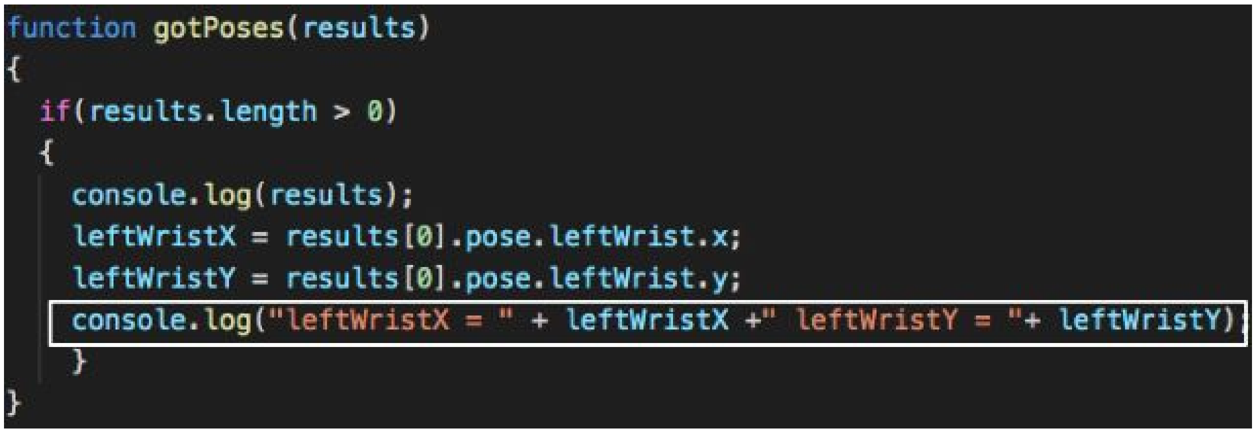
```
function gotPoses(results)
{
  if(results.length > 0)
  {
    console.log(results);
    leftWristX = results[0].pose.leftWrist.x;
  }
}
```

15. Now we have fetched the code for getting the y coordinate of the leftWrist, so update the leftWristY variable with this code, inside **gotPoses()** function.



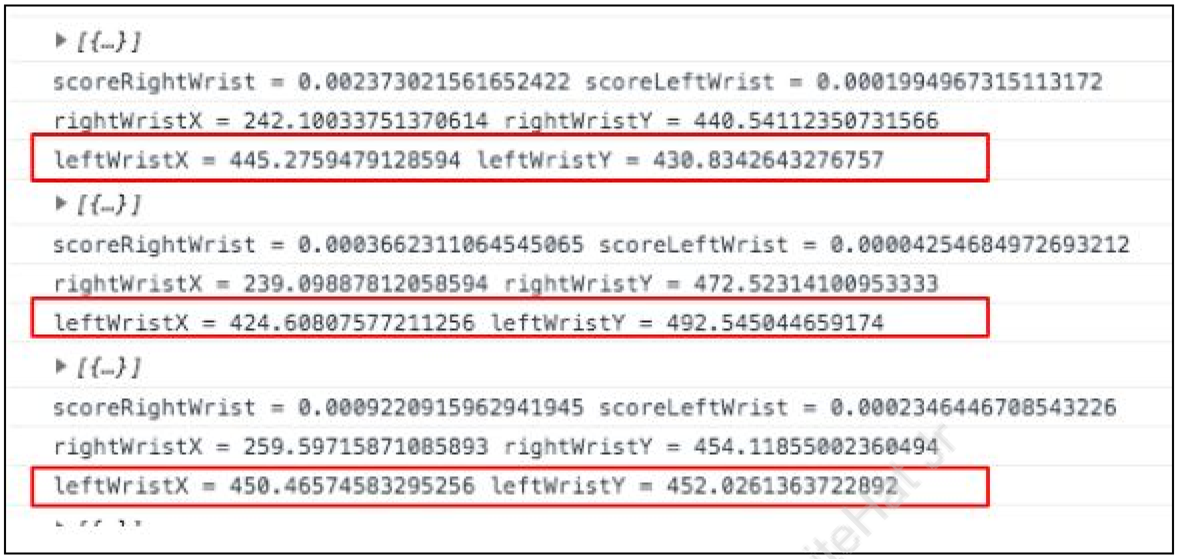
```
function gotPoses(results)
{
  if(results.length > 0)
  {
    console.log(results);
    leftWristX = results[0].pose.leftWrist.x;
    leftWristY = results[0].pose.leftWrist.y;
  }
}
```

16. Now lets console these two variables:



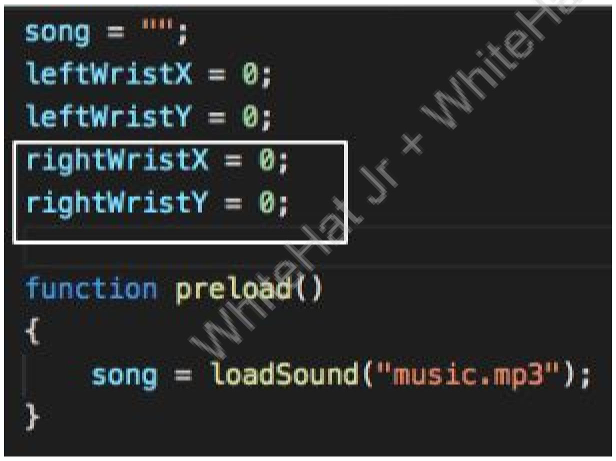
```
function gotPoses(results)
{
  if(results.length > 0)
  {
    console.log(results);
    leftWristX = results[0].pose.leftWrist.x;
    leftWristY = results[0].pose.leftWrist.y;
    console.log("leftWristX = " + leftWristX +" leftWristY = "+ leftWristY);
  }
}
```

- **Output on console screen:**



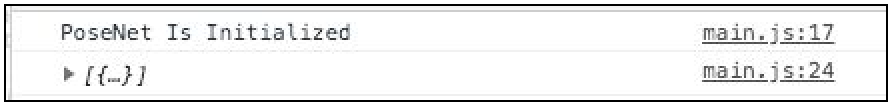
```
▸ [{…}]
scoreRightWrist = 0.002373021561652422 scoreLeftWrist = 0.0001994967315113172
rightWristX = 242.10033751370614 rightWristY = 440.54112350731566
leftWristX = 445.2759479128594 leftWristY = 430.8342643276757
▸ [{…}]
scoreRightWrist = 0.0003662311064545065 scoreLeftWrist = 0.00004254684972693212
rightWristX = 239.09887812058594 rightWristY = 472.52314100953333
leftWristX = 424.60807577211256 leftWristY = 492.545044659174
▸ [{…}]
scoreRightWrist = 0.0009220915962941945 scoreLeftWrist = 0.0002346446708543226
rightWristX = 259.59715871085893 rightWristY = 454.11855002360494
leftWristX = 450.46574583295256 leftWristY = 452.0261363722892
```

17. So first define 2 variables for holding the x and y coordinates of the **rightWrist** and initialize with a value of 0, define these variables at the beginning of the **main.js** file.



```
song = "";
leftWristX = 0;
leftWristY = 0;
rightWristX = 0;
rightWristY = 0;

function preload()
{
    song = loadSound("music.mp3");
}
```

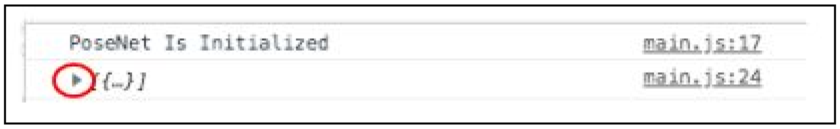
18. Fetch coordinates of the rightWrist. For that again run the https://mahdihat791.github.io/Ai-DJ/ and open the console screen.
    - Output on console screen:



```
PoseNet Is Initialized                    main.js:17
▸ [{…}]                                   main.js:24
```

- The output on the console screen will be very fast as the code is running continuously in real time.

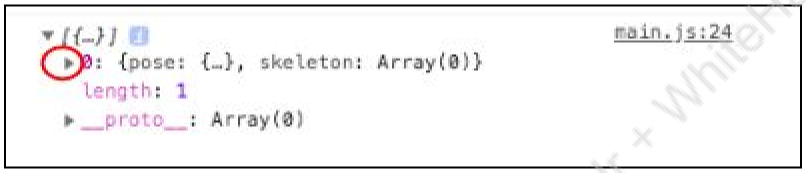
19. We need to read this object again to fetch the x and y coordinates of rightWrist, while reading the object also write the code. First click on the arrow to expand:



```
PoseNet Is Initialized                main.js:17
▶ [{…}]                               main.js:24
```

- We want to read objects of **results** so first write `results`

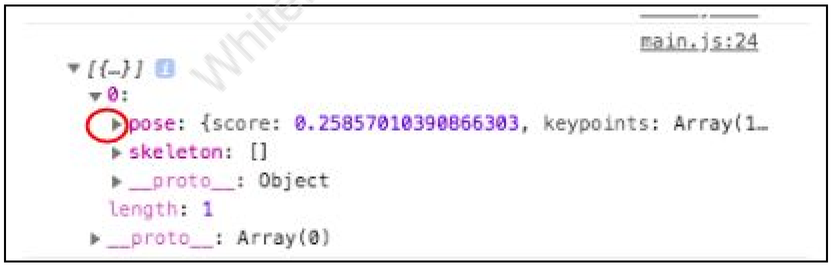
20. Then click on the arrow next to `▶ 0:` to expand:



```
▼ [{…}]                               main.js:24
  ▶ 0: {pose: {…}, skeleton: Array(0)}
    length: 1
  ▶ __proto__: Array(0)
```

- We have clicked on 0 index which is inside the results object so code will be - `results[0]`

21. Then click on the arrow next to `▶ pose:` to expand:



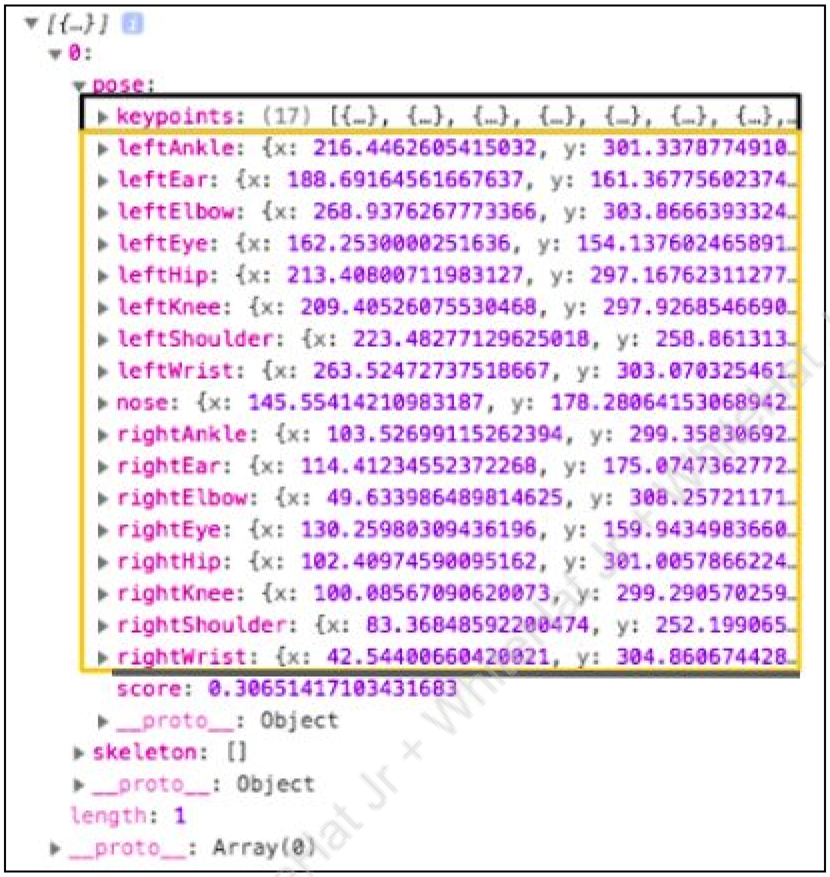
```
                                      main.js:24
▼ [{…}]
  ▼ 0:
    ▶ pose: {score: 0.25857010390066303, keypoints: Array(1…
    ▶ skeleton: []
    ▶ __proto__: Object
    length: 1
  ▶ __proto__: Array(0)
```

- Then inside 0 index we have clicked on pose object, so code will be - `results[0].pose`

This much of code will remain the same every time when we want to access the x and y coordinates of any body parts.

22. Then inside the **pose** object there are two important parts `▶ keypoints:` and 17

body parts with x and y coordinates. ▸ keypoints: has the same thing which is 17 body parts with x and y coordinates. So we don't need to go in ▸ keypoints: , because we will find all the x and y coordinates of 17 body parts outside only.

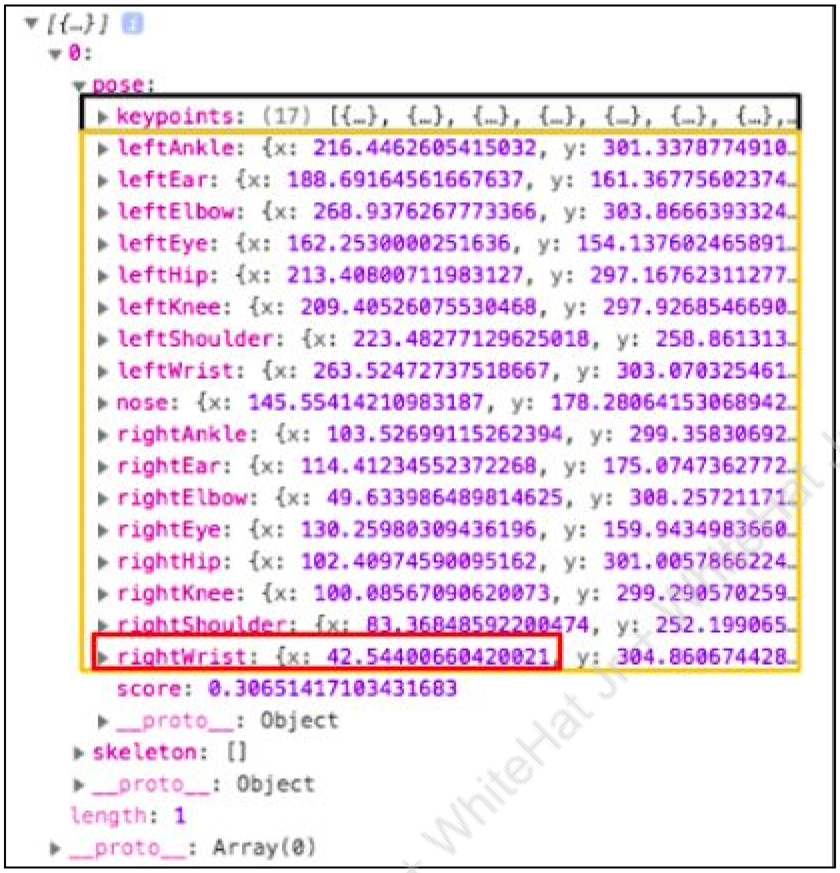


```
▼ [{…}] 
  ▼ 0:
    ▼ pose:
      ▸ keypoints: (17) [{…}, {…}, {…}, {…}, {…}, {…}, {…},…
      ▸ leftAnkle: {x: 216.4462605415032, y: 301.3378774910…
      ▸ leftEar: {x: 188.69164561667637, y: 161.36775602374…
      ▸ leftElbow: {x: 268.9376267773366, y: 303.8666393324…
      ▸ leftEye: {x: 162.2530000251636, y: 154.137602465891…
      ▸ leftHip: {x: 213.40800711983127, y: 297.16762311277…
      ▸ leftKnee: {x: 209.40526075530468, y: 297.9268546690…
      ▸ leftShoulder: {x: 223.48277129625018, y: 258.861313…
      ▸ leftWrist: {x: 263.52472737518667, y: 303.070325461…
      ▸ nose: {x: 145.55414210983187, y: 178.28064153068942…
      ▸ rightAnkle: {x: 103.52699115262394, y: 299.35830692…
      ▸ rightEar: {x: 114.41234552372268, y: 175.0747362772…
      ▸ rightElbow: {x: 49.633986489814625, y: 308.25721171…
      ▸ rightEye: {x: 130.25980309436196, y: 159.9434983660…
      ▸ rightHip: {x: 102.40974590095162, y: 301.0057866224…
      ▸ rightKnee: {x: 100.08567090620073, y: 299.290570259…
      ▸ rightShoulder: {x: 83.36848592200474, y: 252.199065…
      ▸ rightWrist: {x: 42.54400660420821, y: 304.860674428…
        score: 0.30651417103431683
      ▸ __proto__: Object
    ▸ skeleton: []
    ▸ __proto__: Object
    length: 1
  ▸ __proto__: Array(0)
```

- So we want the coordinates of the rightWrist, so **inside results -> inside 0 index -> inside pose -> there is a rightWrist.** So write -

```
results[0].pose.rightWrist
```

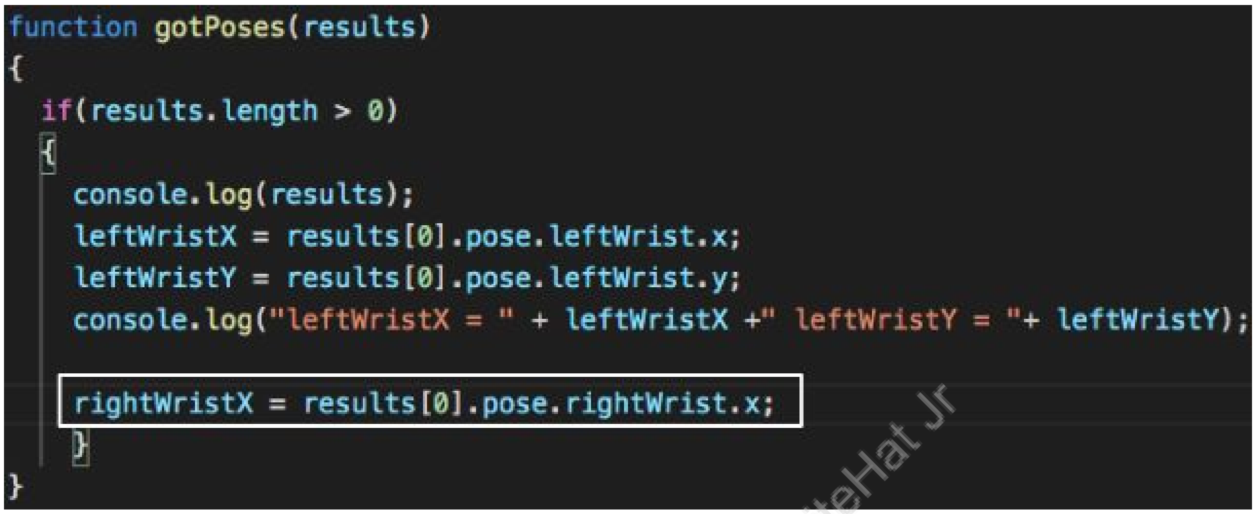
23. Inside rightWrist we want x coordinate:


```
▼ [{…}] 
  ▼ 0:
    ▼ pose:
      ▶ keypoints: (17) [{…}, {…}, {…}, {…}, {…}, {…}, {…},…
      ▶ leftAnkle: {x: 216.4462605415032, y: 301.3378774910…
      ▶ leftEar: {x: 188.69164561667637, y: 161.36775602374…
      ▶ leftElbow: {x: 268.9376267773366, y: 303.8666393324…
      ▶ leftEye: {x: 162.2530000251636, y: 154.137602465891…
      ▶ leftHip: {x: 213.40800711983127, y: 297.16762311277…
      ▶ leftKnee: {x: 209.40526075530468, y: 297.9268546690…
      ▶ leftShoulder: {x: 223.48277129625018, y: 258.861313…
      ▶ leftWrist: {x: 263.52472737518667, y: 303.070325461…
      ▶ nose: {x: 145.55414210983187, y: 178.28064153068942…
      ▶ rightAnkle: {x: 103.52699115262394, y: 299.35830692…
      ▶ rightEar: {x: 114.41234552372268, y: 175.0747362772…
      ▶ rightElbow: {x: 49.633986489814625, y: 308.25721171…
      ▶ rightEye: {x: 130.25980309436196, y: 159.9434983660…
      ▶ rightHip: {x: 102.40974590095162, y: 301.0057866224…
      ▶ rightKnee: {x: 100.08567090620073, y: 299.290570259…
      ▶ rightShoulder: {x: 83.36848592200474, y: 252.199065…
      ▶ rightWrist: {x: 42.54400660420021, y: 304.860674428…
        score: 0.30651417103431683
      ▶ __proto__: Object
    ▶ skeleton: []
    ▶ __proto__: Object
    length: 1
  ▶ __proto__: Array(0)
```

- So the code will be - **inside results -> inside 0 index -> inside pose -> inside the rightWrist -> x coordinate.** Code:

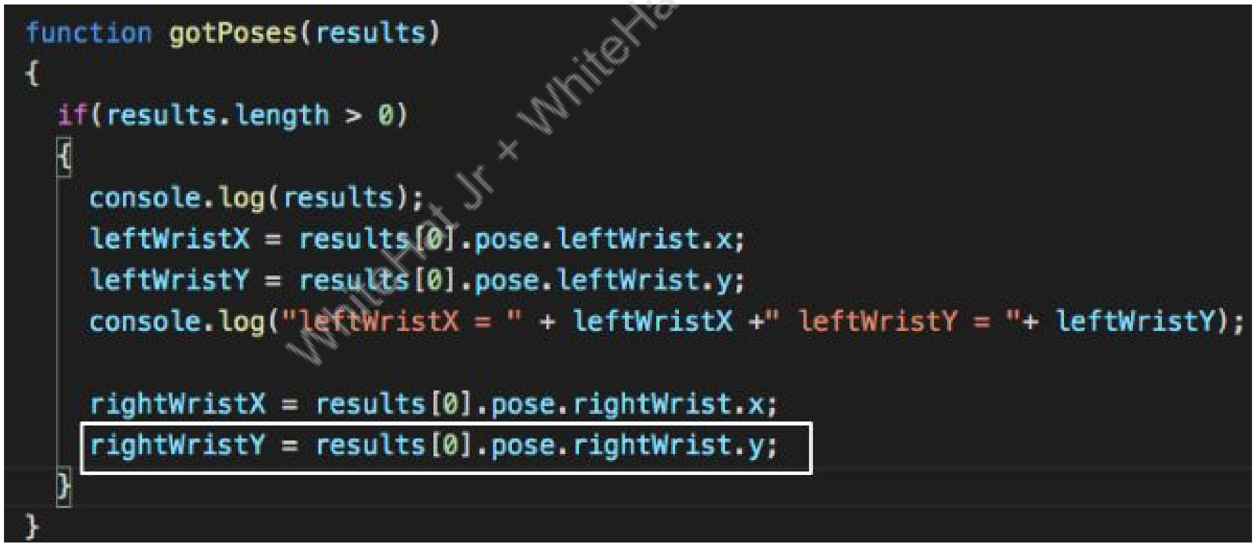
```
results[0].pose.rightWrist.x;
```

24. Now we have fetched the code for getting the x coordinate of rightWrist, so update rightWristX variable with this code, inside **gotPoses()** function:



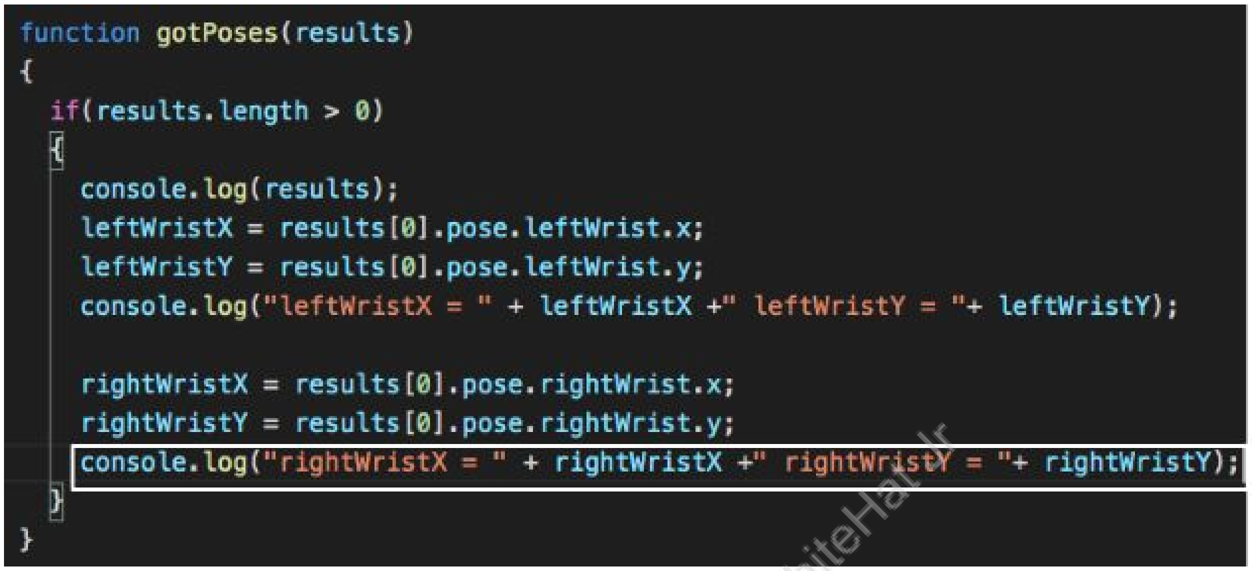
```
function gotPoses(results)
{
  if(results.length > 0)
  {
    console.log(results);
    leftWristX = results[0].pose.leftWrist.x;
    leftWristY = results[0].pose.leftWrist.y;
    console.log("leftWristX = " + leftWristX +" leftWristY = "+ leftWristY);

    rightWristX = results[0].pose.rightWrist.x;
  }
}
```

25. Now we have fetched the code for getting the y coordinate of the rightWrist, so update the rightWristY variable with this code, inside **gotPoses()** function.



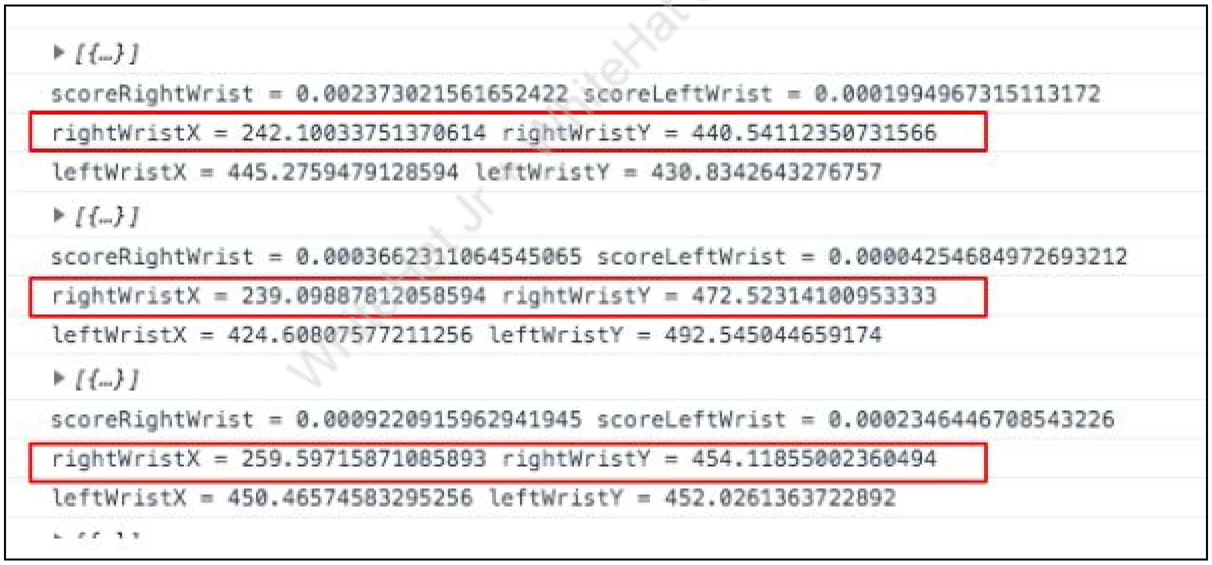
```
function gotPoses(results)
{
  if(results.length > 0)
  {
    console.log(results);
    leftWristX = results[0].pose.leftWrist.x;
    leftWristY = results[0].pose.leftWrist.y;
    console.log("leftWristX = " + leftWristX +" leftWristY = "+ leftWristY);

    rightWristX = results[0].pose.rightWrist.x;
    rightWristY = results[0].pose.rightWrist.y;
  }
}
```

26. Now let's console these two variables.



```
function gotPoses(results)
{
  if(results.length > 0)
  {
    console.log(results);
    leftWristX = results[0].pose.leftWrist.x;
    leftWristY = results[0].pose.leftWrist.y;
    console.log("leftWristX = " + leftWristX +" leftWristY = "+ leftWristY);

    rightWristX = results[0].pose.rightWrist.x;
    rightWristY = results[0].pose.rightWrist.y;
    console.log("rightWristX = " + rightWristX +" rightWristY = "+ rightWristY);
  }
}
```

- **Output on console screen:**



```
▸ [{…}]
scoreRightWrist = 0.002373021561652422 scoreLeftWrist = 0.0001994967315113172
rightWristX = 242.10033751370614 rightWristY = 440.54112350731566
leftWristX = 445.2759479128594 leftWristY = 430.8342643276757
▸ [{…}]
scoreRightWrist = 0.0003662311064545065 scoreLeftWrist = 0.000042546849726932l2
rightWristX = 239.09887812058594 rightWristY = 472.52314100953333
leftWristX = 424.60807577211256 leftWristY = 492.545044659174
▸ [{…}]
scoreRightWrist = 0.0009220915962941945 scoreLeftWrist = 0.0002346446708543226
rightWristX = 259.59715871085893 rightWristY = 454.11855002360494
leftWristX = 450.46574583295256 leftWristY = 452.0261363722892
```

What's NEXT?

We will continue building the AI DJ web app.