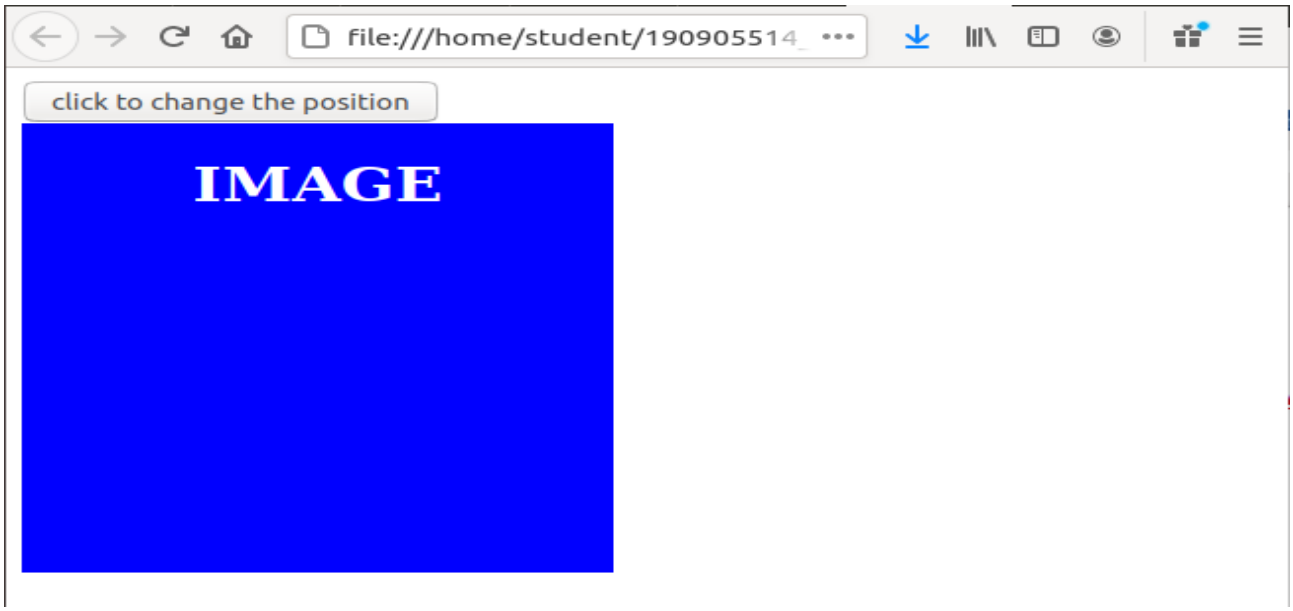
190905514 **MOHAMMAD TOFIK**

WEEK 7 LAB 7 :

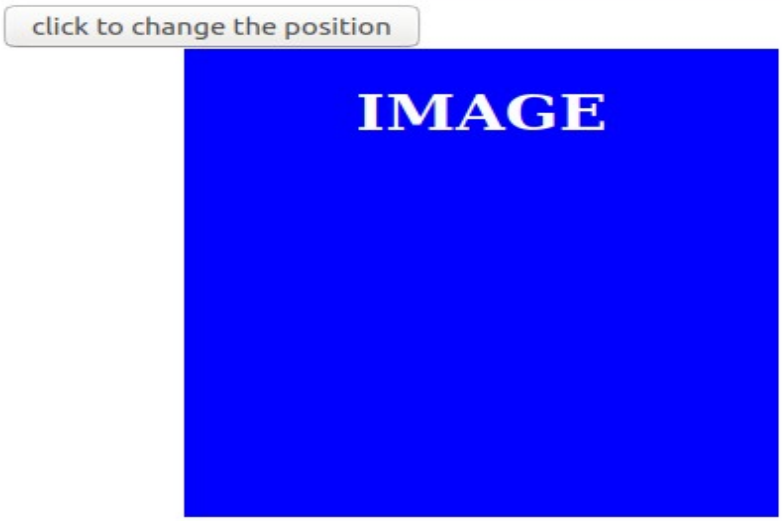
1.changing the position using javascript and css

```
<!DOCTYPE html>
<html>
<head>
<style>
#position {
  position: absolute;
  width: 300px;
  height: 300px;
  background-color: blue;
  color: white;
  text-align: center;
}
</style>
</head>
<body>

<button onclick="fun1()">click to change the position</button>

<div id="position">
  <h1 >IMAGE</h1>
</div>

<script>
function fun1() {
  document.getElementById("position").style.left = "100px";
}
</script>

</body>
</html>
```

BEFORE :

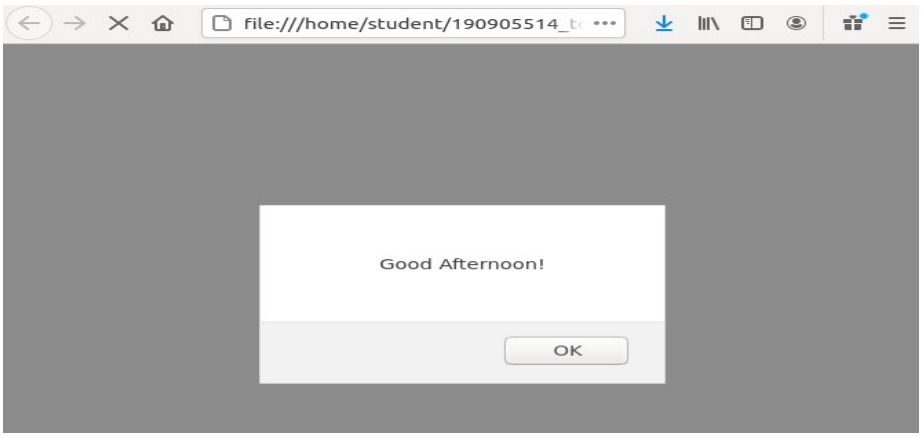


AFTER :



2. wish a user create javascript.

```
<!DOCTYPE html>
<html>
<body>
  <head>
    <title>Wishing | user</title>
  </head>
  <script type="text/javascript">
    var day = new Date();
    var hr = day.getHours();
    if (hr >= 0 && hr < 12) {
      alert("Good Morning!");
    } else if (hr == 12) {
       alert("Good Noon!");
    } else if (hr >= 12 && hr <= 17) {
       alert("Good Afternoon!");
    } else {
                    alert("Good Evening!");
    }

  </script>
</body>
</html>
```



3.create an animation of using html5 and canvas.

```
<!DOCTYPE html>
<html><title>RAIN | DROP | USING | HTML AND CSS  | JAVASCRIPT
<head>
<style>
html,
body {
  height: 100vh;
  width: 100vw;
  display: flex;
  justify-content: center;
  align-items: center;
  overflow: hidden;
  margin: 0;
  padding: 0;

}

body > canvas {
  border: 1px solid black;
  background-color: brown;
  z-index: 10;
}

body > #sky-top {
  height: 100% !important;
  width: 100% !important;
  background-color: rgb(46, 46, 46);
  position: absolute;
  z-index: 1;
  animation: lightning 20s ease-in-out infinite;
}

@keyframes lightning {
  0% {
    background-color: rgb(46, 46, 46);
  }
  6.25% {
    background-color: rgb(46, 46, 46);
  }
  8% {
    background-color: rgb(255, 255, 255);
  }
  9% {
    background-color: rgb(46, 46, 46);
  }
  11% {
    background-color: rgb(255, 255, 255);
  }
  30% {
    background-color: rgb(46, 46, 46);
  }
  100%{
    background-color: rgb(46, 46, 46);
  }
}

body > #sky-bottom {
  height: 100% !important;
  width: 100% !important;
  position: absolute;
```

```
  z-index: 2;
  background: linear-gradient(rgba(255, 255, 255, 0), rgb(120, 140, 155));
}
</style></head>
<body>

<div id="sky-top"></div>
<div id="sky-bottom"></div>
<canvas id="canvas"></canvas>

<!-- canvas fullscreen -->
<script>
  const height = document.body.offsetHeight;
  const width = document.body.offsetWidth;
  const cvs = document.getElementById('canvas');
  cvs.setAttribute("height", height);
  cvs.setAttribute("width", width);
  const canvas = document.getElementById("canvas");
const context = canvas.getContext('2d');
const canvasHeight = canvas.height;
const canvasWidth = canvas.width;

const clearCanvas = function(x, y,height, width) {
  rectX = x || 0;
  rectY = y || 0;
  rectHeight = height || canvasHeight;
  rectWidth = width || canvasWidth;
  context.clearRect(rectX, rectY, rectWidth, rectHeight);
  context.beginPath();
}
const circle = function(x, y, radius, filled) {
  const offset = radius / 2;
  x = x - offset;
  y = y - offset;
  context.beginPath();
  context.arc(x, y, radius, 0, 2 * Math.PI);
  if (filled) {
    context.stroke();
  }
  context.strokeStyle = '#fff';
  context.closePath();
}


const createVector = function(x, y) { return { x, y } }

const vectorAddition = function(vectorA, vectorB) {
  if (typeof vectorB === 'number') {
    return { x: vectorA.x + vectorB, y: vectorA.y + vectorB };
  }
  return { x: vectorA.x + vectorB.x, y: vectorA.y + vectorB.y };
}

const vectorSubtraction = function(vectorA, vectorB) {
  if (typeof vectorB === 'number') {
    return { x: vectorA.x - vectorB, y: vectorA.y - vectorB };
  }
  return { x: vectorA.x - vectorB.x, y: vectorA.y - vectorB.y };
}

const vectorMultiplication = function(vectorA, vectorB) {
  if (typeof vectorB === 'number') {
    return { x: vectorA.x * vectorB, y: vectorA.y * vectorB };
```

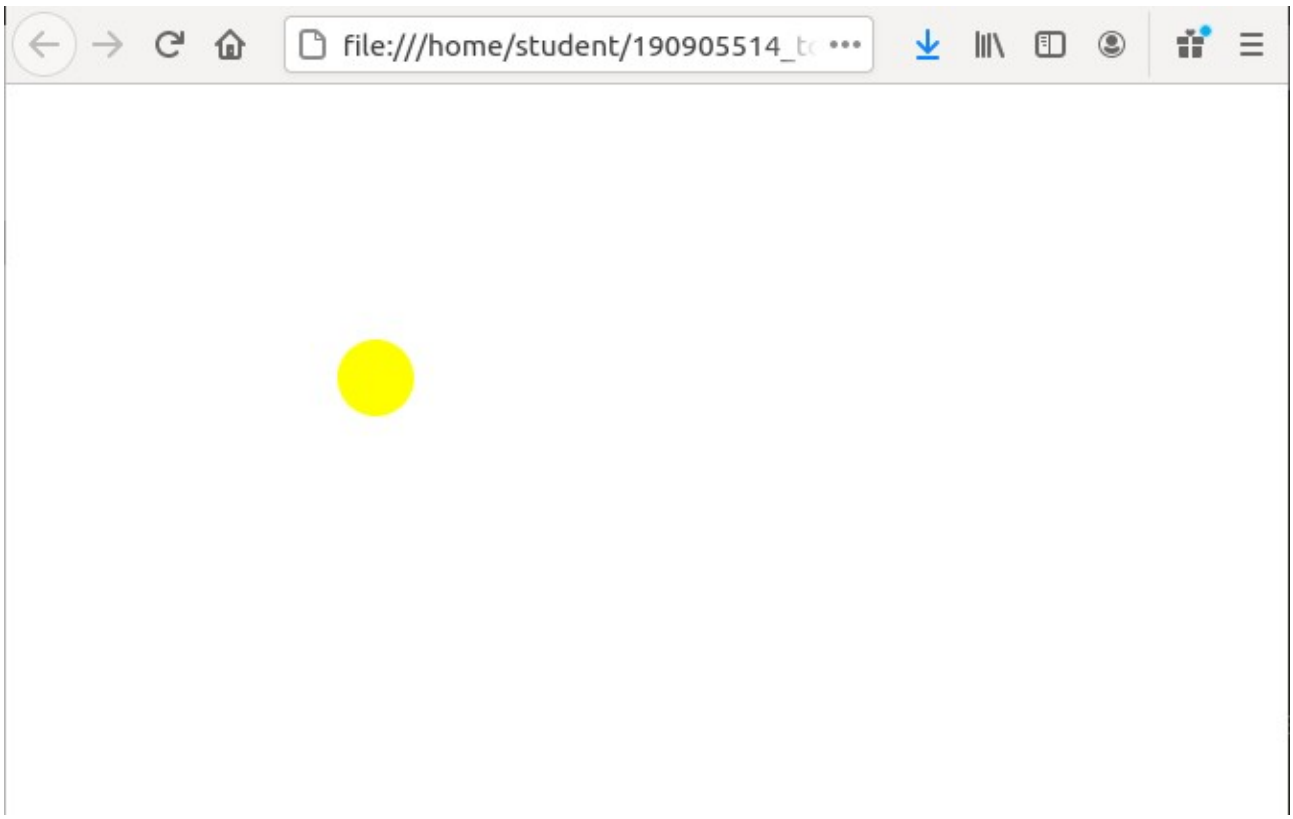
```
  }
  return { x: vectorA.x * vectorB.x, y: vectorA.y * vectorB.y };
}

const vectorDivision = function(vectorA, vectorB) {
  if (typeof vectorB === 'number') {
    return { x: vectorA.x / vectorB, y: vectorA.y / vectorB };
  }
  return { x: vectorA.x / vectorB.x, y: vectorA.y / vectorB.y };
}

const getRandomFloat = function(min, max) {
  const random = Math.random() * (max - min + 1) + min;
  return random;
}

const getRandomInteger = function(min, max) {
  return Math.floor(getRandomFloat(min, max));
}

const checkRaindropCollision = function(location, radius) {
  let rain = { collided: false, location: null }
  if ((location.y - canvasHeight) >= radius) {
    rain.collided = true;
    rain.location = createVector(getRandomInteger(radius, canvasWidth-radius), radius - 10)
  } else if ((location.x + radius) <= 0) {
    rain.collided = true;
    rain.location = createVector(canvasWidth - radius, location.y)
  } else if ((location.x + radius) >= canvasWidth) {
    rain.collided = true;
    rain.location = createVector(radius, location.y)
  } else {
    rain.collided = false;
    rain.location = null;
  }
  return rain;
}

const raintype = {
  drizzle: { count: 30, speed: 0.27 },
  light: { count: 100, speed: 0.3 },
  medium: { count: 250, speed: 0.4 },
  downpour: { count: 500, speed: 0.5 },
  afteshower: { count: 3, speed: 0.4 }
}

var environment = {
  wind: createVector(-0.05, 0),
  raintype: raintype.medium,
}

class RainParticle {
  constructor(x, accX, accY){
    this.damping = 0.025;
    this.location = createVector(x, canvasHeight);
    this.radius = 0.4;
    this.velocity = createVector(0, 0);
    this.acceleration = createVector(accX, -(accY * this.damping));
    this.mass = 1;
  }

  draw(particles, index) {
    const { x, y } = this.location;
```

```
    if (this.acceleration.y >= 0.3) {
      delete particles[index];
      return particles.splice(index, 1)
    }
    return circle(x, y, this.radius, true);
  }

  splash() {
    this.velocity = vectorAddition(this.velocity, this.acceleration);
    this.location = vectorAddition(this.location, this.velocity);
    this.acceleration = vectorAddition(this.acceleration, { x: 0, y: 0.12 });
  }
}

class Raindrop {
  constructor(x, y, radius, accY){
    this.location = createVector(x, y);
    this.radius = radius;
    this.velocity = createVector(0, 0);
    this.acceleration = createVector(0, accY);
    this.mass = 1;

    this.wind = environment.wind;
    this.acceleration = vectorAddition(this.acceleration, this.wind);
  }

  draw() {
    const { x, y } = this.location;
    return circle(x, y, this.radius, true);
  }

  fall() {
    if (this.velocity.y <= (environment.raintype.speed * 50)) {
      this.velocity = vectorAddition(this.velocity, this.acceleration);
    }
    this.location = vectorAddition(this.location, this.velocity);
  }

  relive(raindrop) {
    const { location } = raindrop;
    this.location = createVector(location.x, location.y);
    this.velocity = createVector(0, 0);
  }
}

const particleX = [-0.12, 0.06, 0, 0.06, 0.12];
const getParticleX = function() {
  const index = Math.floor(Math.random() * particleX.length);
  return particleX[index];
}

// init all objects here
let raindrop = [];
let particles = [];
const raindropCount = environment.raintype.count;

for (let i = 0 ; i < raindropCount ; i++) {
  let x = getRandomInteger(2, canvasWidth - 2);
  let y = getRandomInteger(-2000 , 0);
  // let accY = getRandomFloat(1, 5) * 0.05;
  let accY = environment.raintype.speed;
  raindrop[i] = new Raindrop(x, y, 1.3, accY);
}
```

```
// initiate all draw functions here
const setup = function() {
  for (let i = 0 ; i < raindropCount ; i++) {
    raindrop[i].draw();
  }
}

// continuous animation loop
const animate = function() {
  clearCanvas(); // don't remove this

  for (let i = 0 ; i < raindropCount ; i++) {
    let { location, radius, velocity } = raindrop[i];
    let rain = checkRaindropCollision(location, radius);
    if (rain.collided) {
      let particle1 = new RainParticle(location.x, getParticleX(), velocity.y);
      particles.push(particle1);
      let particle4 = new RainParticle(location.x, getParticleX(), velocity.y);
      particles.push(particle4);

      raindrop[i].relive(rain);
    }
    raindrop[i].fall();
    raindrop[i].draw();
  }

  for (let i = 0; i < particles.length; i ++) {
    particles[i].splash();
    particles[i].draw(particles, i);
  }
  requestAnimationFrame(animate);
}

setup();
requestAnimationFrame(animate);
</script>
</body>
</html>
```

4.create a html document that display a boumcing ball.

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>BOUNCING | BAll</title>
</head>
<body onLoad="init();">
  <canvas id="myCanvas" width="300" height="300" >
  </canvas>
  <script>
var context;
var x=50;
var y=10;
var x1=-15;
var x2=-15;

function init()
{
  context= myCanvas.getContext('2d');
  setInterval(draw,10);
}

function draw()
{
  context.clearRect(0,0, 300,300);
  context.beginPath();
  context.fillStyle="yellow";
  context.arc(x,y,20,0,Math.PI*2,true);
  context.closePath();
  context.fill();
if( x<0 || x>300) x1=-x1;
if( y<0 || y>300) x2=-x2;
x+=x1;
y+=x2;
}
</script>
</body>
</html>
```



5.develop a color picker using html and css javascript

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>DEVELOP A COLOR PICKER </title>
        <style>

                .spectrum-wrapper {
  cursor: crosshair;
  width: 360px;
  position: relative;
  height: 20px;
  user-select: none;
  user-drag: none;
}
.spectrum-layer {
  height: 100%;
  width: 100%;
  background: linear-gradient(
    to right,
    rgb(255 0 0),
    rgb(255 255 0),
    rgb(0 255 0),
    rgb(0 255 255),
    rgb(0 0 255),
    rgb(255 0 255),
    rgb(255 0 0)
  );
}
```

```
        </style>
</head>
<body>
        <div class="spectrum-wrapper">
  <div class="spectrum-layer"></div>
</div>
<div>
  <br />
move the mouse on the color
  <br />
  <br />
</div>

<div>Red: <span class="red"></span></div>
<div>Green: <span class="green"></span></div>
<div>Blue: <span class="blue"></span></div>
<div><br />MODIFIER IS =  <span class="hex"></span></div>

<script type="text/javascript">

const getSpectrumWrapper = () => document.querySelector(".spectrum-wrapper");

const spectrumRanges = [
  { from: [255, 0, 0], to: [255, 255, 0] },
  { from: [255, 255, 0], to: [0, 255, 0] },
  { from: [0, 255, 0], to: [0, 255, 255] },
  { from: [0, 255, 255], to: [0, 0, 255] },
  { from: [0, 0, 255], to: [255, 0, 255] },
  { from: [255, 0, 255], to: [255, 0, 0] }
];

const findColorValue = (from, to, leftRatio) => {
  return Math.round(from + (to - from) * leftRatio);
};

const findRgbFromMousePosition = (event) => {
  const wrapper = getSpectrumWrapper();
  const { clientX } = event;
  const { left, width } = wrapper.getBoundingClientRect();
  const leftDistance = Math.min(Math.max(clientX - left, 0), width - 1);
  const totalRanges = spectrumRanges.length;
  const rangeWidth = width / totalRanges;
  const includedRange = Math.floor(leftDistance / rangeWidth);
  const leftRatio = ((leftDistance % rangeWidth) / rangeWidth).toFixed(2);
  const { from, to } = spectrumRanges[includedRange];
  return {
    r: findColorValue(from[0], to[0], leftRatio),
    g: findColorValue(from[1], to[1], leftRatio),
    b: findColorValue(from[2], to[2], leftRatio)
  };
};

const rgbToHex = (r, g, b) => {
  const toHex = (rgb) => {
    let hex = Number(rgb).toString(16);
    if (hex.length < 2) {
      hex = `0${hex}`;
    }
    return hex;
  };
  const red = toHex(r);
  const green = toHex(g);
  const blue = toHex(b);
```

```
  return `#${red}${green}${blue}`;
};

getSpectrumWrapper().addEventListener("mousemove", (e) => {
  const { r, g, b } = findRgbFromMousePosition(e);
  const hexValue = rgbToHex(r, g, b);
  document.querySelector(".red").innerText = r;
  document.querySelector(".green").innerText = g;
  document.querySelector(".blue").innerText = b;
  document.querySelector(".hex").innerText = hexValue;
});


</script>

</body>
</html>
```