# **PHẦN TỬ CANVAS TRONG HTML 5**

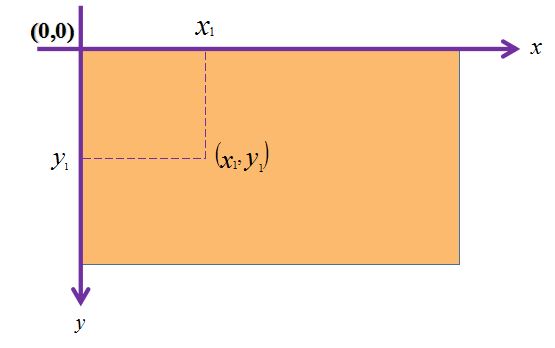
# **1 Canvas Là Gì?**

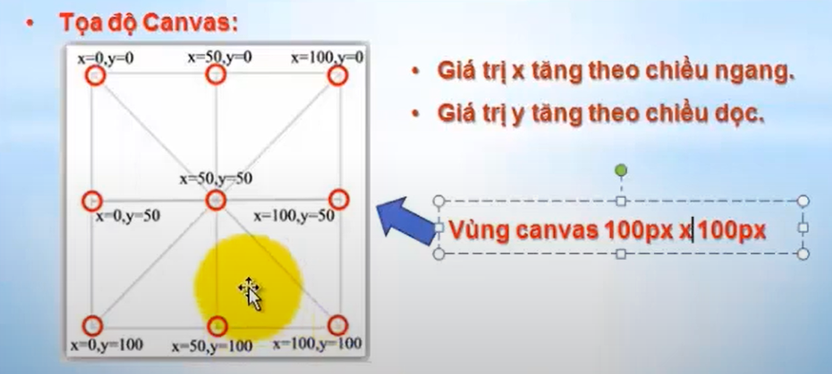
# **Phần tử <canvas>** trong [HTML](https://quantrimang.com/hoc/html) được sử dụng để vẽ đồ họa cho trang web **Phần tử <canvas>** sử dụng để vẽ các thành phần đồ họa từ đơn giản như các đường đến các đồ họa phức tạp (đường thẳng, hình hộp, hình tròn, viết chữ, chèn hình ảnh...).

## **Tọa độ trong canvas**

Canvas sử dụng trục tọa độ 2 chiều, với góc trái phía trên là tọa độ gốc (0,0)

* X : tọa độ theo phương nằm ngang, tăng dần từ trái sang phải
* Y : tọa độ theo phương thẳng đứng, tăng dần từ trên xuống dưới





## Khai báo Canvas

<canvas id="myCanvas" width="200" height="100"></canvas>

Thêm Đường Viền, Sử Dụng Thuộc Tính Style

<canvas id="myCanvas" width="500" height="200" style="border:1px solid #000000;"></canvas>

Kết quả:  


**Vẽ đường thẳng trong Canvas**

Để vẽ đường thẳng trên một đối tượng canvas, ta sẽ dùng hàm sau:

* Thường để vẽ các đường đầu tiên ta gọi beginPath() để khởi tạo một path mới (nó chứa tập hợp các lệnh đường cần vẽ).
* moveTo(x,y): xác định điểm bắt đầu
* lineTo(x,y): xác định điểm kết thúc
* Cuối cùng để thực hiện vẽ tập đó gọi hàm stroke()

<canvas id="myCanvas" width="500" height="200" style="border:1px solid #8080ff;">

</canvas

<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

// Đường thứ nhất

ctx.beginPath(); // Khai báo vẽ đường thẳng mới

ctx.moveTo(0,0); // Xác định điểm bắt đầu

ctx.lineTo(500,200); // Xác định điểm kết thúc

ctx.strokeStyle = 'red' // Màu của đường thẳng

ctx.stroke(); // Tiến hành vẽ

//Đường thứ hai

ctx.beginPath(); // Khai báo vẽ đường thẳng mới

ctx.moveTo(0,200); // Xác định điểm bắt đầu

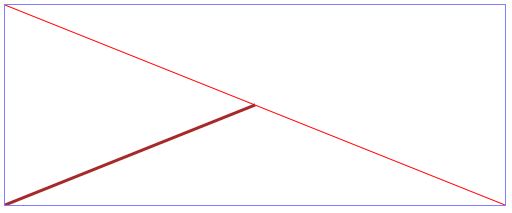
ctx.lineTo(250,100); // Xác định điểm kết thúc

ctx.strokeStyle = 'brown' // Màu của đường thẳng

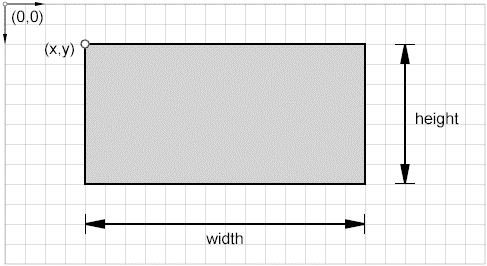
ctx.lineWidth = 3; //Độ rộng

ctx.stroke(); // Tiến hành vẽ

</script>



## **Vẽ hình chữ nhật (Draw Rectangles)**



The rect() phương pháp thêm một hình chữ nhật vào đường dẫn.

*context*.rect(*x, y, width, height*)

Để vẽ một hình chữ nhật, chỉ định tọa độ x và y (góc trên bên trái) và chiều cao và chiều rộng của hình chữ nhật. Có ba phương pháp hình chữ nhật:

<script>

var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

context.beginPath();

context.rect(188, 50, 200, 100);

context.fillStyle = 'yellow';

context.fill();

context.lineWidth = 7;

context.strokeStyle = 'black';

context.stroke();

</script>

****

**Viết chữ trong Canvas**

Để viết một đoạn văn bản, ta chú trọng đến thuộc tính và hàm sau:

* font: định kiểu chữ cho đoạn văn bản.
* fillText(text,x,y): Viết chữ trên canvas.
* strokeText(text,x,y):Viết chữ trên canvas nhưng không tô màu chữ.

// Drawing Basic Text

ctx.font = "30px Arial";// Đặt kiểu chữ và kích thước của văn bản.

ctx.fillStyle = "blue"; // Đặt màu cho văn bản.

ctx.fillText("Hello, World!", 50, 100); // Vẽ text vị trí (50, 50)

// Shadows on Text

ctx.shadowColor = "red // Đặt màu của bóng.

ctx.shadowBlur = 5; // Đặt độ mờ của bóng.

ctx.shadowOffsetX = 2; //Đặt độ dịch chuyển bóng theo trục X và Y.

ctx.shadowOffsetY = 2;

ctx.fillText(" Text with Shadows", 50, 150);

// Stroke text

ctx.lineWidth = 2;

ctx.strokeStyle = "green";

ctx.strokeText("Text with Stroke", 145, 200);

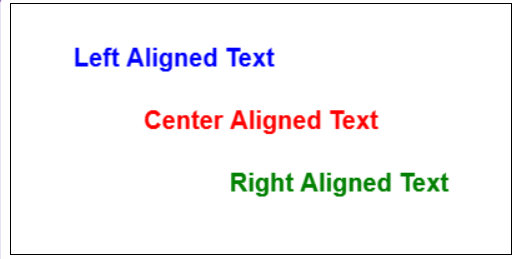
// Text Align  
 ctx.textAlign = "center";

ctx.fillStyle = "purple";

ctx.fillText("Centered Text", 160, 250);  
 // Text base line  
 ctx.textBaseline = "bottom";

ctx.fillStyle = "orange";

ctx.fillText("Text with Bottom Baseline", 170, 300);



**Màu Gradient (Màu biến đổi tuyến tính) trong Canvas**

Có 2 loại gradients:

* **createLinearGradient(x,y,x1,y1):** Quét màu xuất phát từ 1 phía, bốn tham số biểu diễn điểm bắt đầu *(x,y)* và điểm kết thúc *(x1,y1)* của gradient.
* **createRadialGradient(x,y,r,x1,y1,r1):** Quét màu xuất phát từ tâm, bộ tham số đầu tiên định nghĩa một đường tròn với tọa độ tâm *(x,y)* và bán kính r bắt đầu; và bộ tham số thứ hai là vòng tròn với tọa độ tâm *(x1,y1*) và bán kính r1 kết thúc gradient.

Hàm *addColorStop()* quy định điểm dừng quét màu dọc theo điểm quét (gradient). Điểm gradient có thể ở bất kỳ giữa 0 và 1.

Sau khi thiết lập gradient, ta dùng *fillStyle* hoặc *strokeStyle* để gán gradient vào ô canvas,  
**Ví dụ: Dùng createLinearGradient()**

<!DOCTYPE html>

<html>

<body>

<canvas id="myCanvas" width="500" height="200" style="background: pink"></canvas>

<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

// Tạo gradient

var grd = ctx.createLinearGradient(0,40,470,150);

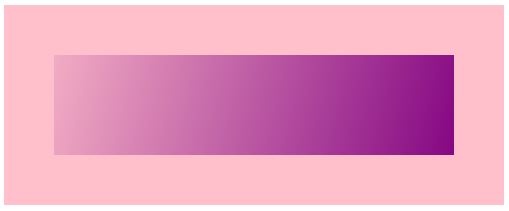
grd.addColorStop(0,"pink");

grd.addColorStop(1,"purple");

ctx.fillStyle = grd;

ctx.fillRect(50,50,400,100);

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



**Createradialgradient()**<!DOCTYPE html>

<html>

<body>

<canvas id="myCanvas" width="500" height="200" style="background: LightSkyBlue">

</canvas>

<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

// Create

gradientvar grd = ctx.createRadialGradient(125,100,10,125,100,150);

grd.addColorStop(0,"coral");

grd.addColorStop(1,"Moccasin");

var grd1 = ctx.createRadialGradient(375,100,10,375,100,150);

grd1.addColorStop(0,"moccasin");

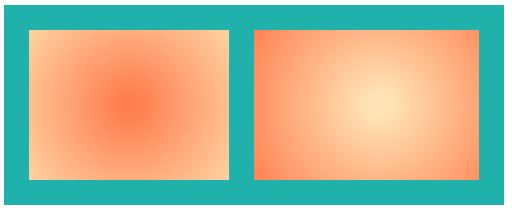
grd1.addColorStop(1,"coral");// Fill with gradientctx.fillStyle = grd;ctx.fillRect(25,25,200,150);

ctx.fillStyle = grd1;ctx.fillRect(250,25,225,150);

</script>

</body>

</html>



## **Line Cap**

Thuộc tính lineCap được sử dụng để lấy hoặc đặt kiểu nắp dòng hiện tại. Có ba kiểu nắp:

* butt : Mặc định. Một cạnh phẳng được đặt vuông góc với mỗi đầu của dòng mà không cần thêm nắp.
* round : Một hình bán nguyệt hoặc nắp kết thúc tròn được thêm vào mỗi đầu của dòng.
* square : Một nắp cuối hình vuông được thêm vào mỗi đầu của dòng.

var context = canvas.getContext("2d");

context.beginPath();

context.moveTo(20,20);

context.lineTo(20,200);

context.lineWidth = 25;

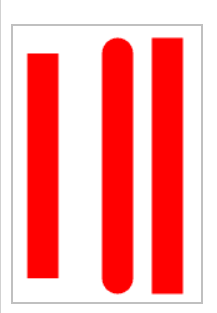
context.strokeStyle = '#ff0000';

context.lineCap = 'butt';

//context.lineCap = 'round';

//context.lineCap = 'square';

context.stroke();

****

## **Line Join**

Thuộc tính lineJoin được sử dụng để lấy hoặc đặt loại góc được tạo khi hai dòng nối với nhau. Có ba giá trị:

* bevel : Một hình tam giác đầy kết nối hai đường được nối, tạo ra một góc vát.
* round: Các đường nối với một góc tròn.
* square : Mặc định. Các đường nối với một góc mượt mà.

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

var lStart = 50;

var lEnd = 200;

var yStart = 20;

ctx.beginPath();

ctx.lineWidth = 25;

ctx.lineJoin = "bevel";

//ctx.lineJoin = "round";

//ctx.lineJoin = "miter";

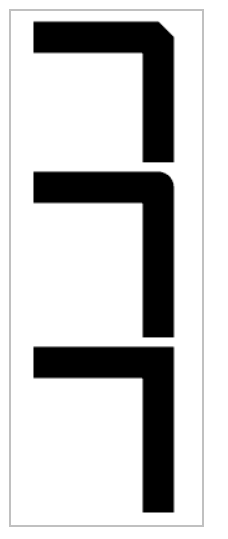
ctx.moveTo(50, 20);

ctx.lineTo(150, 20);  
 // Di chuyển đến điểm (50, 20) và vẽ đường thẳng đến điểm (150, 20).

ctx.lineTo(150, 120);

// Vẽ thêm một đoạn đường thẳng từ điểm trước đó đến điểm (150, 120).

ctx.stroke();

****

## **Shadows**

Trong canvas HTML5, bạn có thể thêm bóng trên hình dạng, đường kẻ, văn bản hoặc hình ảnh có thể tạo cảm giác về chiều thứ ba. Để thêm bóng với HTML5 Canvas, bạn có thể sử dụng các thuộc tính sau của ngữ cảnh canvas.

* shadowOffsetX
* shadowOffsetY
* shadowColor
* shadowBlur

## **shadowOffsetX() Property**

Thuộc tính được sử dụng để lấy hoặc đặt khoảng cách ngang của bóng từ một hình dạng. **Syntax:**

[ctx](https://www.w3resource.com/html5-canvas/index.php#ctx).shadowOffsetX = h\_distance;

## Trong đó h\_distance (loại: số) là khoảng cách ngang của bóng đổ từ một hình dạng.

## **shadowOffsetY() Property**

Thuộc tính được sử dụng để lấy hoặc đặt khoảng cách thẳng đứng của bóng từ một hình dạng.

**Syntax:**

[ctx](https://www.w3resource.com/html5-canvas/index.php#ctx).shadowOffsetX = v\_distance;

Trong đó v\_distance (loại: số) là khoảng cách thẳng đứng của bóng đổ từ hình dạng.

## **shadowColor() Property**

Thuộc tính được sử dụng để lấy hoặc đặt màu để sử dụng cho bóng.

**Syntax:**

[ctx](https://www.w3resource.com/html5-canvas/index.php#ctx).shadowColor

Trong đó shadowColor (type: string) là màu CSS.

## **shadowBlur() Property**

Thuộc tính được sử dụng để lấy hoặc đặt mức độ mờ hiện tại được áp dụng cho bóng.

**Syntax:**

[ctx](https://www.w3resource.com/html5-canvas/index.php#ctx).shadowBlur = blur\_value

Trong đó blur\_value là lượng mờ (loại: số) được áp dụng cho bóng.

var ctx = canvas.getContext('2d');

ctx.shadowColor = "black";

ctx.shadowBlur = 6; // Giá trị mờ của bóng

ctx.shadowOffsetX = 6; // khoảng cách ngang hình dạng với bóng

ctx.shadowOffsetY = 6; // khoảng cách đứng hình dạng với bóng

ctx.shadowColor = "orange";

ctx.strokeRect(25, 25, 200, 200);

ctx.shadowColor = "green";

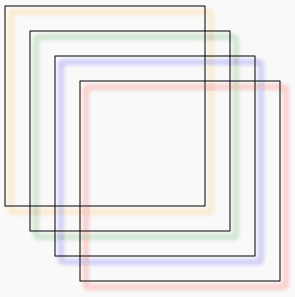
ctx.strokeRect(50, 50, 200, 200);

ctx.shadowColor = "blue";

ctx.strokeRect(75, 75, 200, 200);

ctx.shadowColor = "red";

ctx.strokeRect(100, 100, 200, 200);



**Vẽ cung tròn, Curves - hình tròn trong Canvas**

Để vẽ cung tròn - hình tròn, ta dùng lệnh: *arc(x,y,r,start,stop)*

* x và y: tọa độ tâm
* r: là bán kính
* start: là góc bắt đầu
* end: là góc kết thúc

<canvas id="myCanvas" width="600" height="200" style="border:1px solid teal;background: LightYellow"></canvas>

<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

// Cung tròn 1

ctx.beginPath();

ctx.arc(100, 100, 90, Math.PI, 0); // Tọa độ tâm (100,100), bán kính 90

ctx.strokeStyle = 'steelblue'; // Màu của đường

ctx.lineWidth = 5; //Độ rộng

ctx.stroke();

// Hình tròn

ctx.beginPath();

ctx.arc(300, 100, 90, 0, 2\*Math.PI); // Tọa độ tâm (300,100), bán kính 90

ctx.strokeStyle = 'salmon' // Màu của đường

ctx.lineWidth = 3; //Độ rộng

ctx.stroke();

// Cung tròn 2

ctx.beginPath();

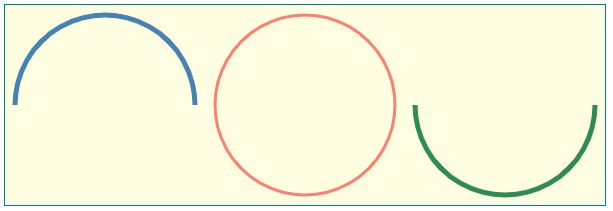
ctx.arc(500, 100, 90, 0, Math.PI); // Tọa độ tâm (500,100), bán kính 90

ctx.strokeStyle = 'seagreen'; // Màu của đường

ctx.lineWidth = 5; //Độ rộng

ctx.stroke();

</script>

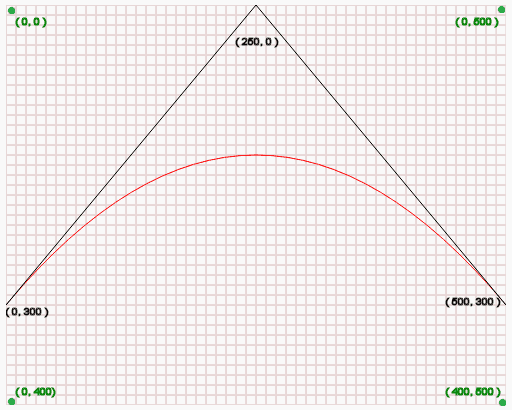


## **Quadraticcurveto() Method**

Một đường cong Bézier bậc hai đòi hỏi hai điểm. Điểm đầu tiên là điểm kiểm soát được sử dụng trong tính toán Bézier bậc hai và điểm thứ hai là điểm kết thúc cho đường cong.

**Syntax :**  
quadraticCurveTo(cp1x, cp1y, x, y);

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Type** | **Description** |
| cp1x | number | Tọa độ x của điểm kiểm soát Bézier. |
| cp1y | number | Tọa độ y của điểm kiểm soát Bézier. |
| x | number | Tọa độ x của điểm để thêm vào đường dẫn hiện tại. |
| y | number | Tọa độ y của điểm để thêm vào đường dẫn hiện tại. |



* (0, 300) là điểm khởi đầu của đường cong.
* (250, 0) i.e. (CP1X, CP1Y) là vị trí kiểm soát của đường cong.
* (500, 300) tức là (x, y) là điểm kết thúc của đường cong.

<canvas id="myCanvas" width="578" height="200"></canvas>

<script>

var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

context.beginPath();

context.moveTo(188, 150); // tọa độ bắt đầu

context.quadraticCurveTo(100, 0, 388, 150);

// (cp1x, cp1y) là tọa độ của điểm điều khiển, nơi mà đường cong "chạy" qua.

// (x, y) là tọa độ của điểm kết thúc của đường cong.

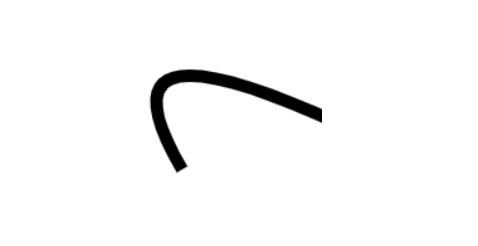
context.lineWidth = 10;

// line color

context.strokeStyle = 'black';

context.stroke();

</script>



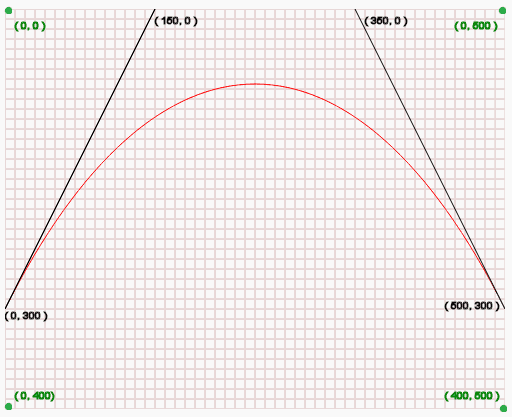
## **Beziercurveto() Method**

Một đường cong Bézier khối phải bao gồm ba điểm. Hai điểm đầu tiên (cp1x, cp1y) và (cp2x, cp2y) là các điểm kiểm soát được sử dụng trong tính toán Bézier khối và điểm cuối cùng (x, y) là điểm kết thúc cho đường cong.

**Syntax:**  
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)

|  |  |  |
| --- | --- | --- |
| **Thông số** | **Kiểu** | **Sự miêu tả** |
| CP1X | Number | Tọa độ x của điểm kiểm soát Bézier đầu tiên. |
| CP1Y | Number | Tọa độ y của điểm kiểm soát Bézier đầu tiên. |
| CP2X | Number | Tọa độ x của điểm kiểm soát Bézier thứ hai. |
| CP2Y | Number | Tọa độ y của điểm kiểm soát Bézier thứ hai. |
| x | Number | Tọa độ x của điểm để thêm vào đường dẫn hiện tại. |
| y | Number | Tọa độ y của điểm để thêm vào đường dẫn hiện tại. |

Lưu ý : Điểm đầu tiên trên đường cong là điểm cuối cùng trong đường con hiện tại. Nếu đường dẫn không tồn tại, hãy sử dụng phương thức beginPath và moveTo để tạo điểm bắt đầu.



* (0, 300) là điểm khởi đầu của đường cong.
* (150, 0) tức là (CP1X, CP1y) là vị trí điều khiển đầu tiên của đường cong.
* (350, 0) tức là (CP2X, CP2Y) là vị trí điều khiển thứ hai của đường cong.
* (500, 300) tức là. (x, y) là điểm kết thúc của đường cong.

// Vẽ đường cong Bézier

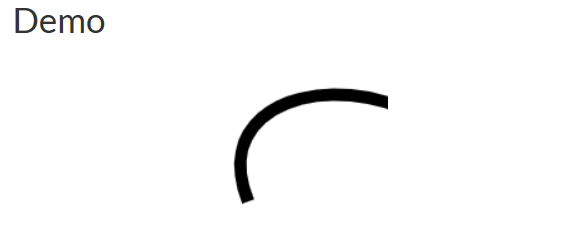
ctx.beginPath();

context.moveTo(188, 130);

context.bezierCurveTo(140, 10, 388, 10, 388, 170);

// line color

context.strokeStyle = 'black';

****

# Canvas phương thức drawImage()

Phương thức **CanvasRenderingContext2D.drawImage()** của API Canvas 2D cung cấp các cách khác nhau để vẽ hình ảnh lên canvas.

function draw() {

const ctx = document.getElementById("myCanvas").getContext("2d");

const img = new Image(); *// Create an image object*

img.onload = () => { *// Draw the image when it's loaded*

ctx.drawImage(img, 0, 0);

//ctx.drawImage(imageObj, x, y, width, height);

ctx.beginPath();

ctx.moveTo(30, 96);

ctx.lineTo(70, 66);

ctx.lineTo(103, 76);

ctx.lineTo(170, 15);

ctx.stroke();

};

img.src = "./image/img.jpg";

}

****

Save and restore()

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

// Vẽ một hình chữ nhật màu xanh

ctx.fillStyle = "green";

ctx.fillRect(10, 10, 50, 50);

// Lưu trạng thái hiện tại của canvas

ctx.save();

// Đổi màu và vẽ một hình chữ nhật màu đỏ ở vị trí khác

ctx.fillStyle = "red";

ctx.fillRect(100, 10, 50, 50);

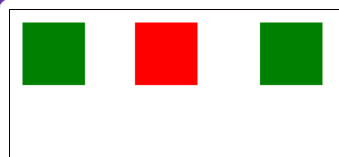
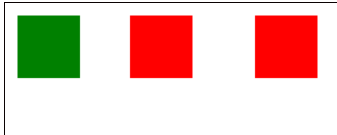
// Khôi phục lại trạng thái đã lưu (quay về hình chữ nhật màu xanh)

ctx.restore();

// Vẽ một hình chữ nhật màu xanh khác ở vị trí khác

ctx.fillRect(200, 10, 50, 50);

console.error("Không tìm thấy phần tử Canvas");

Download IMG in Canvas

// Create a temporary link element

var dataURL = c.toDataURL('image/png'); // Change 'image/png' to 'image/jpeg' if needed

var link = document.createElement('a');

link.href = dataURL;

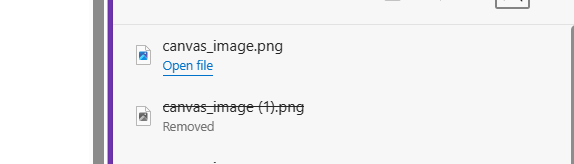
link.download = 'canvas\_image.png'; // Change the filename as needed

// Trigger the click event on the link to download the image

document.body.appendChild(link);

link.click();

document.body.removeChild(link);



**Animation**Đầu tiên bạn nên vẽ nội dung của bạn vào canvas bằng cách sử dụng javaScript bước tiếp theo là nhanh chóng xóa khung vẽ , và vẽ lại rất nhanh nội dung ở vị trí mới của chúng để tạo hoạt ảnh (Animation)

## [Basic animation steps](https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_animations#basic_animation_steps)

Đây là các bước bạn cần thực hiện để vẽ khung:

1. **Clear the canvas** Trừ khi các hình dạng bạn sẽ vẽ lấp đầy khung vẽ hoàn chỉnh (ví dụ: hình ảnh phông nền), bạn cần xóa bất kỳ hình dạng nào đã được vẽ trước đó. Cách dễ nhất để làm điều này là sử dụng phương thức [**clearRect().**](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/clearRect)
2. **Save the canvas state** Nếu bạn đang thay đổi bất kỳ cài đặt nào (chẳng hạn như kiểu, chuyển đổi, v.v.) ảnh hưởng đến trạng thái canvas và bạn muốn đảm bảo trạng thái ban đầu được sử dụng mỗi khi vẽ khung, bạn cần lưu trạng thái ban đầu đó.
3. **Draw animated shapes** Bước mà bạn thực hiện kết xuất khung hình thực tế.
4. **Restore the canvas state** Nếu bạn đã lưu trạng thái, hãy khôi phục trạng thái đó trước khi vẽ khung mới.

### [Scheduled updates](https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_animations#scheduled_updates)

Đầu tiên là các hàm [**setInterval()**](https://developer.mozilla.org/en-US/docs/Web/API/setInterval)**,**[**setTimeout()**](https://developer.mozilla.org/en-US/docs/Web/API/setTimeout)**và**[**window.requestAnimationFrame(),**](https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame) có thể được sử dụng để gọi một hàm cụ thể trong một khoảng thời gian nhất định.

[**setInterval()**](https://developer.mozilla.org/en-US/docs/Web/API/setInterval)

Bắt đầu thực hiện liên tục chức năng được chỉ định bởi mỗi mili giây.functiondelay

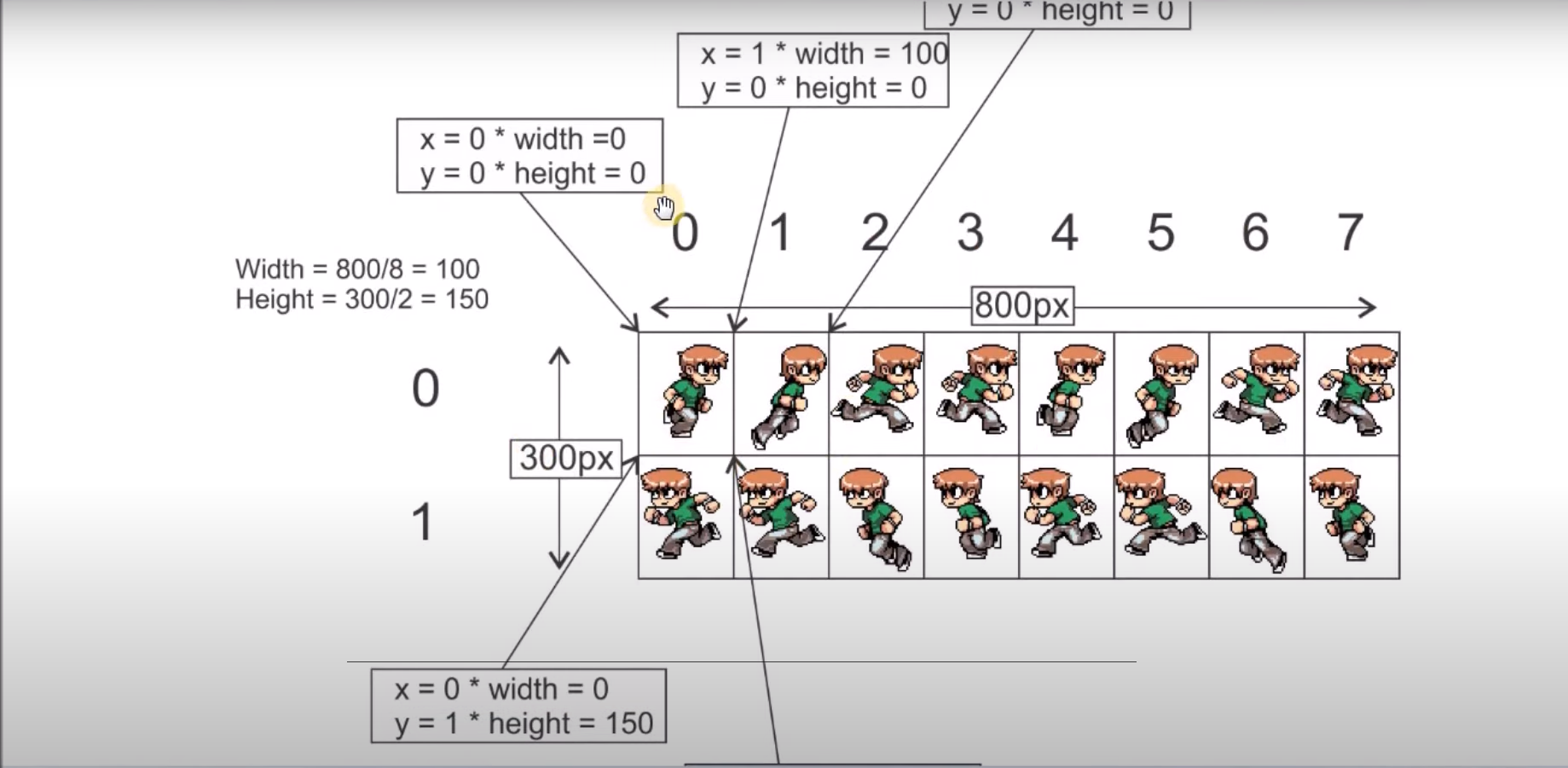
[**setTimeout()**](https://developer.mozilla.org/en-US/docs/Web/API/setTimeout)

Thực hiện chức năng được chỉ định bởi tính bằng mili giây.functiondelay

[**requestAnimationFrame(callback)**](https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame)

Cho trình duyệt biết rằng bạn muốn thực hiện hoạt ảnh và yêu cầu trình duyệt gọi một chức năng được chỉ định để cập nhật hoạt ảnh trước khi sơn lại tiếp theo.

**Ví DỤ Sprite Animation**



<canvas id='canvas'></canvas><br />

<button onclick='moveLeft()'>Left</button>

<button onclick='moveRight()'>Right</button>

// chiều dài và chiều rộng của Canvas

var canvasWidth = 1500;

var canvasHeight = 300;

// chiều dài và chiều cao của ảnh spritesheet

var spriteWidth = 864;

var spriteHeight = 280;

// Có 2 hàng và 8 cột trong ảnh sprite sheet

var rows = 2;

var cols = 8;

// Hàng thứ 0 (đầu tiên) dành cho chuyển động bên phải

var trackRight = 0;

// Hàng thứ 1 (thứ hai) dành cho chuyển động sang trái (đếm chỉ số từ 0)

var trackLeft = 1;

// Để có được chiều rộng của một sprite chúng ta chia chiều rộng của sprite cho số cols

//vì tất cả các sprite đều có chiều rộng và chiều cao bằng nhau

var width = spriteWidth / cols;

// Tương tự với chiều cao ta chia chiều cao cho số hàng

var height = spriteHeight / rows;

// Mỗi hàng chứa 8 khung và khi bắt đầu chúng ta sẽ hiển thị khung đầu tiên (giả sử chỉ số từ 0)

var curFrame = 0;

// Tổng khung hình là 8

var frameCount = 8;

// tọa độ x và y để hiển thị sprite

var x = 0;

var y = 0;

// tọa độ x và y của khung vẽ để lấy khung đơn

var srcX = 0;

var srcY = 0;

//theo dõi chuyển động sang trái và viết

var left = false;

// Giả sử lúc đầu nhân vật sẽ di chuyển sang phải

var right = true;

//Tốc độ chuyển động

var speed = 12;

// Lấy id canvas

var canvas = document.getElementById('canvas');

//thiết lập chiều rộng và chiều cao của canvas

canvas.width = canvasWidth;

canvas.height = canvasHeight;

// Thiết lập bối cảnh cho canvas

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

//Tạo một đối tượng Image cho nhân vật của chúng ta

var character = new Image();

//Đặt nguồn cho file ảnh

character.src = '../image/character.png';

function draw() {

//Updating the frame

updateFrame();

//Drawing the image

ctx.drawImage(character, srcX, srcY, width, height, x, y, width, height);

}

setInterval(draw, 100);

//left right

function moveLeft() {

left = true;

right = false;

}

function moveRight() {

left = false;

right = true;

}

function updateFrame() {

//Updating the frame index

curFrame = ++curFrame % frameCount; 0% 8=0 , 1 %8 =1

//Tính tọa độ x cho spritesheet

srcX = curFrame \* width;

//Xóa khung vẽ

ctx.clearRect(x, y, width, height);

//nếu left là true và ký tự chưa tới cạnh trái

if (left && x > 0) {

//tính srcY

srcY = trackLeft \* height;

//giảm tọa độ x

x -= speed;

}

//nếu bên phải là đúng và ký tự chưa đến cạnh phải

if (right && x < canvasWidth - width) { // tính tọa độ y cho spritesheet

srcY = trackRight \* height;

// tăng tọa độ x

x += speed; }}



• ctx.drawImage(character, srcX, srcY, width, height, x, y, width, height);, các giá trị như srcX, srcY, width, và height được sử dụng để xác định phần cụ thể của spritesheet bạn muốn vẽ. Còn x, y, width, và height là vị trí và kích thước trên canvas mà bạn muốn vẽ hình ảnh.

• image: Đối tượng hình ảnh mà bạn muốn vẽ lên canvas.

• srcx: Vị trí x (hoặc cột) trên hình ảnh nguồn (spritesheet) từ nơi bạn muốn lấy hình để vẽ.

• srcy: Vị trí y (hoặc hàng) trên hình ảnh nguồn từ nơi bạn muốn lấy hình để vẽ.

• sWidth: Chiều rộng của phần hình ảnh mà bạn muốn lấy từ hình ảnh nguồn.

• sHeight: Chiều cao của phần hình ảnh mà bạn muốn lấy từ hình ảnh nguồn.

• dx: Vị trí x trên canvas nơi bạn muốn vẽ hình ảnh.

• dy: Vị trí y trên canvas nơi bạn muốn vẽ hình ảnh.

• dWidth: Chiều rộng của hình ảnh khi nó được vẽ lên canvas.

• dHeight: Chiều cao của hình ảnh khi nó được vẽ lên canvas.

**Transformations on Canvas**  
**Scale**

Scale là một phương thức biến đổi (transformation) được sử dụng để thay đổi kích thước của các phần tử trên canvas

**context.scale(scaleX, scaleY);**

**scaleX**: Hệ số tỷ lệ phóng to/thu nhỏ theo chiều ngang. Giá trị nhỏ hơn 1 sẽ làm cho phần tử thu nhỏ, còn giá trị lớn hơn 1 sẽ làm cho phần tử phóng to.

**scaleY**: Hệ số tỷ lệ phóng to/thu nhỏ theo chiều dọc. Tương tự như scaleX.

<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>

<script>

const c = document.getElementById("myCanvas");

const ctx = c.getContext("2d");

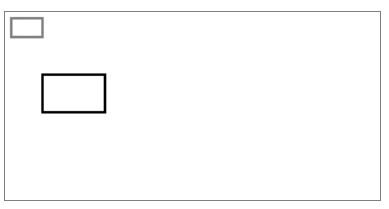
ctx.strokeRect(5, 5, 25, 15);

ctx.scale(2, 2);

ctx.translate(10, 20);

ctx.strokeRect(5, 5, 25, 15);

</script>

****

**Rotate**

Rotate là một phương thức biến đổi (transformation) được sử dụng để xoay các phần tử trên canvas. Phương thức này cho phép bạn thay đổi hướng của hình ảnh hoặc đối tượng theo một góc xác định.

**context.rotate(angle);**

**angle**: Góc xoay được chỉ định theo radian. Để chuyển đổi từ độ sang radian, bạn có thể sử dụng công thức: radian = degree \* (Math.PI / 180).

<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>

<script>

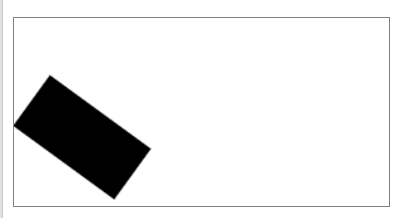
const c = document.getElementById("myCanvas");

const ctx = c.getContext("2d");

ctx.rotate(20 \* Math.PI / 100);

ctx.fillRect(50, 20, 100, 50);

</script>



**Translate**

**translate** là một phương thức biến đổi (transformation) được sử dụng để di chuyển vị trí của các phần tử trên canvas. Phương thức này thay đổi hệ tọa độ của canvas, làm cho các phần tử vẽ sau đó được hiển thị ở vị trí mới tính từ điểm gốc tọa độ đã được di chuyển.

**ctx.translate(x, y);**

**x**: Số lượng pixel cần di chuyển theo chiều ngang.

**y**: Số lượng pixel cần di chuyển theo chiều dọc.

<canvas id="myCanvas" width="300" height="150" style="border:1px solid grey"></canvas>

<script>

const c = document.getElementById("myCanvas");

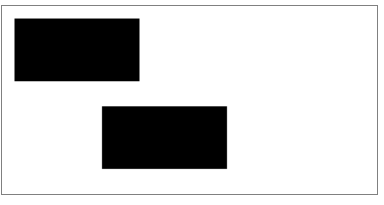
const ctx = c.getContext("2d");

ctx.fillRect(10, 10, 100, 50);

ctx.translate(70, 70);

ctx.fillRect(10, 10, 100, 50);

</script>



**Transform**

**transform** Phương thức transform cho phép bạn kết hợp nhiều biến đổi và xác định thứ tự của chúng trong một lệnh duy nhất.

**context.transform(a, b, c, d, e, f);**

**a** và **d** là hệ số tỷ lệ theo chiều ngang(a) và chiều dọc(d). **Scale (**thay đổi kích thước)

**b** và **c** là hệ số xoay và cắt theo chiều ngang(c) chiều dọc(b). **Shear**(nghiêng)

**e** và **f** là giá trị dịch chuyển theo chiều ngang(a) và chiều dọc(b). **Translate** (di chuyển)

<script>

ctx.fillStyle = "yellow";

ctx.fillRect(0, 0, 250, 100)

ctx.transform(1, 0.5, -0.5, 1, 30, 10);

ctx.fillStyle = "red";

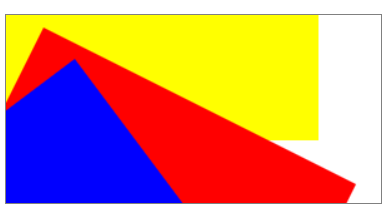
ctx.fillRect(0, 0, 250, 100);

ctx.transform(1, 0.5, -0.5, 1, 30, 10);

ctx.fillStyle = "blue";

ctx.fillRect(0, 0, 250, 100);

</script>



**Making Physics with Animations on Canvas:   
1. Vận tốc (Velocity):**

Vận tốc biểu thị tốc độ và hướng chuyển động của một đối tượng. Trong ngữ cảnh của các đoạn mã vẽ trên canvas, chúng ta có thể tạo một đối tượng di chuyển với vận tốc không đổi.

// Vận tốc

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

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

var viTriX = 50;

var vanTocX = 2; // Vận tốc không đổi theo hướng ngang

function chuyenDong() {

ctx.clearRect(0, 0, canvas.width, canvas.height);

viTriX += vanTocX;

ctx.fillRect(viTriX, 50, 30, 30); // Vẽ hình vuông ở vị trí đã cập nhật

requestAnimationFrame(chuyenDong);

}

chuyenDong();

**Acceleration**

**acceleration** (gia tốc) được sử dụng để mô phỏng tốc độ thay đổi của một đối tượng. Gia tốc được sử dụng trong phương trình chuyển động để biểu thị tốc độ thay đổi của vị trí theo thời gian.

var positionX = 50;

var velocityX = 0; // gia tốc = 0

var accelerationX = 0.1; // Gia tốc không đổi theo hướng ngang

function animate() {

ctx.clearRect(0, 0, canvas.width, canvas.height);

velocityX += accelerationX; //Tăng vận tốc của hình chữ nhật dựa trên gia tốc.

positionX += velocityX; //Cập nhật vị trí của hình chữ nhật dựa trên vận tốc.

ctx.fillRect(positionX, 50, 30, 30);

requestAnimationFrame(animate);

**Vertical Projection**

**Vertical Projection** thường được hiểu là quá trình chiếu các điểm dữ liệu từ chiều dọc của không gian 2D lên một đường thẳng hoặc một trục dọc. Trong ngữ cảnh của canvas HTML5, bạn có thể sử dụng JavaScript để thực hiện vertical projection để hiển thị hoặc xử lý dữ liệu.

const data = [50, 120, 80, 160, 100]; // Dữ liệu điểm chiều dọc

function drawData() { // Vẽ các điểm dữ liệu trên canvas

context.clearRect(0, 0, canvas.width, canvas.height);

context.beginPath();

for (let i = 0; i < data.length; i++) {

// Vertical projection: chiếu lên trục y

const x = i \* 80;

const y = canvas.height - data[i];

context.arc(x, y, 5, 0, 2 \* Math.PI);

context.fillStyle = '#0095DD';

context.fill();

context.stroke();

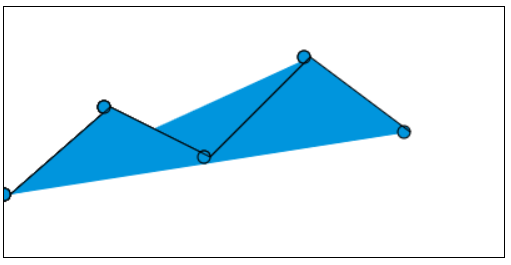
}

context.closePath();

}

// Gọi hàm để vẽ dữ liệu

drawData();



**Horizontal Projection and Angular Projection**

**Horizontal Projection** trong canvas sử dụng HTML và JavaScript để tạo một hiệu ứng giả lập vật lý, trong đó quả bóng được phóng ngang và rơi xuống dưới tác động của trọng lực:

var ball = {

x: 50,

y: canvas.height - 30, // Đặc điểm ban đầu của quả bóng

radius: 15,

color: '#0095DD',

speedX: 5, // Tốc độ ngang

gravity: 0.5 // Trọng lực

};

function drawBall() { // Hàm vẽ quả bóng

ctx.beginPath();

ctx.arc(ball.x, ball.y, ball.radius, 0, Math.PI \* 2);

ctx.fillStyle = ball.color;

ctx.fill();

ctx.closePath();

}

function updateBall() { // Hàm cập nhật vị trí của quả bóng

ball.x += ball.speedX; // Cập nhật vị trí ngang của quả bóng

ball.y += ball.gravity; // Áp dụng trọng lực để quả bóng rơi xuống

ctx.clearRect(0, 0, canvas.width, canvas.height); // Vẽ lại canvas với vị trí mới

drawBall();

requestAnimationFrame(updateBall); // Lặp lại hàm cập nhật để tạo hiệu ứng liên tục

}

// Gọi hàm cập nhật để bắt đầu vòng lặp

updateBall();

**Angular Projection** trong ngữ cảnh của vật lý và đồ họa canvas, có thể bạn đang nghĩ đến một khái niệm liên quan đến chuyển động ném hoặc chuyển động góc. Trong vật lý, chuyển động ném liên quan đến chuyển động của một vật thể được ném lên không trung, thường dưới tác động của trọng lực.

<script>

var canvas = document.getElementById('canvasChuyenDongGoc');

var ctx = canvas.getContext('2d');

var vatTheNem = {

x: 50,

y: canvas.height - 30,

vanTocX: 5,

vanTocY: -10,

trọngLuc: 0.5,

bánKính: 10

};

function veVatTheNem() {

ctx.beginPath();

ctx.arc(vatTheNem.x, vatTheNem.y, vatTheNem.bánKính, 0, Math.PI \* 2);

ctx.fillStyle = '#0095DD';

ctx.fill();

ctx.closePath();

}

function capNhatVatTheNem() {

vatTheNem.x += vatTheNem.vanTocX;

vatTheNem.y += vatTheNem.vanTocY;

// Áp dụng trọng lực để mô phỏng chuyển động ném

vatTheNem.vanTocY += vatTheNem.trọngLuc;

// Xóa canvas để vẽ lại

ctx.clearRect(0, 0, canvas.width, canvas.height);

// Vẽ lại vật thể

veVatTheNem();

// Lặp lại hàm cập nhật để tạo hiệu ứng hoạt hình

requestAnimationFrame(capNhatVatTheNem);

}

// Gọi hàm cập nhật để bắt đầu hoạt hình

capNhatVatTheNem(); </script>

**Momentum on One Axis**

Động lượng là một khái niệm trong vật lý, thường được ký hiệu là p, và được định nghĩa là tích của khối lượng (m) của một đối tượng và vận tốc (v) của nó.

Công thức toán học để tính động lượng là:

**p=m⋅v**

Trong đó:

**p** là động lượng,

**m** là khối lượng của đối tượng,

**v** là vận tốc của đối tượng.   
  
**momentum** trên một trục trong canvas HTML5, bạn có thể sử dụng JavaScript để mô phỏng chuyển động của một đối tượng di chuyển với đàng lượng trên trục ngang.

var object = {

x: 50,

y: canvas.height - 30,

velocityX: 2, // Vận tốc trên trục x

mass: 1, // Khối lượng

};

function drawObject() {

ctx.beginPath();

ctx.arc(object.x, object.y, 15, 0, Math.PI \* 2);

ctx.fillStyle = '#0095DD';

ctx.fill();

ctx.closePath();

}

// var momentumX = object.mass \* object.velocityX;

function updateObject() {

object.x += object.velocityX;

//object.x += momentumX;

// Clear canvas

ctx.clearRect(0, 0, canvas.width, canvas.height);

// Draw object

drawObject();

// Repeat the update function for animation

requestAnimationFrame(updateObject);

}

// Call the update function to start the animation

updateObject();

**Momentum on Two Axes**

Động lượng trên Hai Trục" (Momentum on Two Axes) thường ám chỉ việc mô phỏng hoặc biểu diễn động lượng của một đối tượng trên cả hai chiều hoành độ (trục x) và chiều tung độ (trục y) trong hệ thống tọa độ.

var object = {

x: 50,

y: canvas.height - 30,

velocityX: 2,

velocityY: -2, // Thêm thành phần vận tốc theo trục y

mass: 1,

};

function drawObject() {

ctx.beginPath();

ctx.arc(object.x, object.y, 15, 0, Math.PI \* 2);

ctx.fillStyle = '#0095DD';

ctx.fill();

ctx.closePath();

}

function updateObject() {

object.x += object.velocityX;

object.y += object.velocityY;

ctx.clearRect(0, 0, canvas.width, canvas.height);

drawObject();

requestAnimationFrame(updateObject);

}

updateObject();

[HTML Canvas (w3schools.com)](https://www.w3schools.com/graphics/canvas_intro.asp)  
[HTML5 Canvas Element Tutorial (html5canvastutorials.com)](https://www.html5canvastutorials.com/tutorials/html5-canvas-element/)  
[HTML5 Canvas Tutorial for beginners - w3resource](https://www.w3resource.com/html5-canvas/)  
[Phần tử Canvas trong HTML5 - QuanTriMang.com](https://quantrimang.com/hoc/phan-tu-canvas-trong-html5-160093)  
[JavaScript Tutorial (tutorialspoint.com)](https://www.tutorialspoint.com/javascript/index.htm)  
[HTML Canvas Tutorial - GeeksforGeeks](https://www.geeksforgeeks.org/html-canvas-tutorial/?ref=lbp)

https://www.youtube.com/watch?v=RMUxP3ZNTEI