

TEHNICI WEB

CANVAS & SVG

Claudia Chiriță . 2024/2025

CANVAS

CANVAS

- `<canvas>` element HTML folosit pentru a desena via scripturi (de regulă cod JavaScript)
- permite desenarea de grafice, editare de imagini, creare de animații simple
- dimensiune default: 300px X 150px

```
<canvas id="tutorial" width="150" height="150">  
  fallback text  
</canvas>
```

RENDERING CONTEXT

- elementul <canvas> crează o suprafață de desen de dimensiune fixă, ce afișează unul sau mai multe contexte de randare
- contextele de randare sunt folosite pentru crearea și manipularea conținutului afișat
- ne concentrăm pe contextul de randare 2D

```
const canvas = document.getElementById("tutorial");  
const ctx = canvas.getContext("2d");
```

TEMPLATE DE BAZĂ

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <style>
      canvas {
        border: 5px solid pink;
      }
    </style>
    <script>
      window.onload = function() {
        draw();
      }

      function draw() {
        const canvas =
document.getElementById("tutorial");
        if (canvas.getContext) {
          const ctx = canvas.getContext("2d");
        }
      }
    </script>
  </head>
  <body>
    <canvas id="tutorial" width="150"
```



TEMPLATE DE BAZĂ

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        draw();
      }

      function draw() {
        const canvas =
document.getElementById("tutorial");
        if (canvas.getContext) {
          const ctx = canvas.getContext("2d");

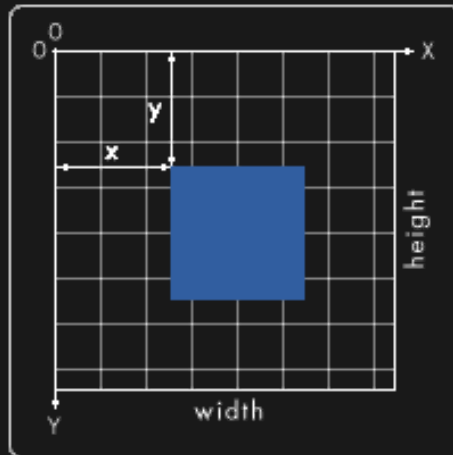
          ctx.fillStyle = "rgb(200, 0, 0)";
          ctx.fillRect(10, 10, 50, 50);

          ctx.fillStyle = "rgba(0, 0, 200,
0.5)";
          ctx.fillRect(30, 30, 50, 50);
        }
      }
    </script>
  </head>
```



GRID - SPAȚIU DE COORDONATE

- o unitate în grid corespunde unui pixel pe canvas
- originea gridului este în colțul din stânga sus, la coordonatele (0,0)



FIGURI PRIMITIVE

- rectangles
- paths

DREPTUNGHURI

```
fillRect(x, y, width, height)  
// desenează un dreptunghi plin
```

```
strokeRect(x, y, width, height)  
// desenează un contur de dreptunghi
```

```
clearRect(x, y, width, height)  
// șterge zona dreptunghiulară, făcând-o transparentă
```

DREPTUNGHIURI

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        draw();
      }

      function draw() {
        const canvas =
document.getElementById("tutorial");
        if (canvas.getContext) {
          const ctx = canvas.getContext("2d");

          ctx.fillStyle = "pink";
          ctx.fillRect(25, 25, 100, 100);
          ctx.clearRect(45, 45, 60, 60);
          ctx.strokeStyle = "pink";
          ctx.strokeRect(50, 50, 50, 50);
        }
      }
    </script>
  </head>
  <body>
```



PATHS

- liste de puncte conectate de segmente de linii care pot fi curbate, pot avea grosimi și culori diferite
- un path poate fi închis cu `closePath()`
- pentru a crea figuri folosind paths:
 1. creăm path-ul cu `beginPath()`
 2. folosim comenzi de desenat în path
 3. adăugăm contur cu `stroke()` ori umplem path-ul cu `fill()` pentru a îl randa

PATHS

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        draw();
      }

      function draw() {
        const canvas =
document.getElementById("tutorial");
        if (canvas.getContext) {
          const ctx = canvas.getContext("2d");

          ctx.beginPath();
          ctx.moveTo(75, 50);
          ctx.lineTo(100, 75);
          ctx.lineTo(100, 25);
          ctx.fillStyle = "pink";
          ctx.fill();
        }
      }
    </script>
  </head>
```



LINII

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        draw();
      }

      function draw() {
        const canvas =
document.getElementById("tutorial");
        if (canvas.getContext) {
          const ctx = canvas.getContext("2d");

          ctx.beginPath();
          ctx.moveTo(25, 25);
          ctx.lineTo(105, 25);
          ctx.lineTo(25, 105);
          ctx.fillStyle = "pink";
          ctx.fill();

          ctx.beginPath();
          ctx.moveTo(125, 125);
          ctx.lineTo(125, 45);
```



ARCE DE CERC

```
arc(x, y, radius, startAngle, endAngle, counterclockwise)  
/* arc centrat în (x, y) cu raza r  
   începând de la startAngle și până la endAngle,  
   în direcția dată de counterclockwise */
```

```
arcTo(x1, y1, x2, y2, radius)  
/* arc cu punctele de control și raza date,  
   conectat la punctul anterior printr-o linie dreaptă */
```

ARCE DE CERC

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        draw();
      }

      function draw() {
        const canvas =
document.getElementById("tutorial");
        if (canvas.getContext) {
          const ctx = canvas.getContext("2d");
          ctx.fillStyle = "pink";
          ctx.strokeStyle = "pink";

          for (let i = 0; i < 4; i++) {
            for (let j = 0; j < 3; j++) {
              ctx.beginPath();
              const x = 25 + j * 50; // x
coordinate
              const y = 25 + i * 50; // y
coordinate
              const radius = 20; // arc
```



CURBE BÉZIER

```
quadraticCurveTo(cp1x, cp1y, x, y)  
/* curbă Bézier cuadratică de la poziția curentă  
   la punctul specificat de x și y,  
   folosind punctul de control (cp1x, cp1y) */
```

```
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)  
/* curbă Bézier cubică de la poziția curentă  
   la punctul specificat de x și y,  
   folosind punctele de control  
   (cp1x, cp1y) și (cp2x, cp2y) */
```

[citește mai multe despre curbe Bézier](#)

CURBE BÉZIER

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        draw();
      }

      function draw() {
        const canvas =
document.getElementById("tutorial");
        if (canvas.getContext) {
          const ctx = canvas.getContext("2d");
          ctx.fillStyle = "pink";

          ctx.beginPath();
          ctx.moveTo(75, 40);
          ctx.bezierCurveTo(75, 37, 70, 25,
50, 25);
          ctx.bezierCurveTo(20, 25, 20, 62.5,
20, 62.5);
          ctx.bezierCurveTo(20, 80, 40, 102,
75, 120);
          ctx.bezierCurveTo(110, 102, 130,
```



OBIECTE PATH2D

- folosim obiecte **Path2D** pentru a simplifica codul și îmbunătăți performanța, înregistrând comenzile de desen
- putem folosi toate metodele de path de mai devreme (moveTo, rect, arc etc.)
- putem combina path-uri:

```
Path2D.addPath(path [, transform])  
/* adaugă un path la path-ul curent  
   cu o matrice de transformare opțională */
```

OBIECTE PATH2D

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        draw();
      }

      function draw() {
        const canvas =
document.getElementById("tutorial");
        if (canvas.getContext) {
          const ctx = canvas.getContext("2d");
          ctx.fillStyle = "pink";
          ctx.strokeStyle = "pink";

          const rectangle = new Path2D();
          rectangle.rect(10, 10, 50, 50);

          const circle = new Path2D();
          circle.arc(100, 35, 25, 0, 2 *
Math.PI);

          ctx.stroke(rectangle);
```



STILURI ȘI CULORI

- culori

```
fillStyle = color;  
strokeStyle = color;
```

- transparență

```
globalAlpha = transparencyValue; // valori în [
```

STILURI ȘI CULORI

- stiluri linie

```
lineWidth = value;  
lineCap = type; // butt, round, square  
lineJoin = type; // round, bevel, miter  
  
getLineDash(); // patternul de linie dash curent  
setLineDash(segments); // setează patternul de linie dash  
lineDashOffset = value; // unde să înceapă patternul dash
```

STILURI ȘI CULORI

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        const ctx =
document.getElementById("canvas").getContext("
2d");
        ctx.strokeStyle = "pink";
        let offset = 0;

        function draw() {
          ctx.clearRect(0, 0, canvas.width,
canvas.height);
          ctx.setLineDash([4, 2]);
          ctx.lineDashOffset = -offset;
          ctx.strokeRect(10, 10, 100, 100);
        }

        function march() {
          offset++;
          if (offset > 16) {
            offset = 0;
          }
        }
      }
    </script>
  </head>
  <body>
    <canvas id="canvas" width="300px" height="150px">
  </body>
</html>
```



STILURI ȘI CULORI

- gradienti

```
createLinearGradient(x1, y1, x2, y2)
```

```
// gradient liniar de la (x1,y1) la (x2,y2)
```

```
createRadialGradient(x1, y1, r1, x2, y2, r2)
```

```
/* gradient radial: un cerc cu centrul în (x1,y1) și raza r1  
și unul cu centrul în (x2, y2) și raza r2 */
```

```
createConicGradient(angle, x, y)
```

```
/* gradient conic cu unghi de angle radiani  
la poziția (x, y) */
```

STILURI ȘI CULORI

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        const ctx =
document.getElementById("canvas").getContext("
2d");

        // Create gradient
        const lingrad =
ctx.createLinearGradient(0, 0, 0, 150);
        lingrad.addColorStop(0, "#00ABEB");
        lingrad.addColorStop(0.95, "#fff");

        ctx.fillStyle = lingrad;
        ctx.fillRect(10, 10, 130, 130);
      }
    </script>
  </head>
  <body>
    <canvas id="canvas" width="150"
height="200"></canvas>
  </body>
```

>

STILURI ȘI CULORI

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {
        const ctx =
document.getElementById("canvas").getContext("
2d");

        // Create gradient
        const radgrad =
ctx.createRadialGradient(45, 45, 10, 52, 50,
30);
        radgrad.addColorStop(0, "#A7D30C");
        radgrad.addColorStop(0.9, "#019F62");
        radgrad.addColorStop(1, "rgba(1, 159,
98, 0)");

        ctx.fillStyle = radgrad;
        ctx.fillRect(0, 0, 150, 150);

      }
    </script>
  </head>
```



STILURI ȘI CULORI

- pattern

```
createPattern(image, type)  
/* image: sursa unei imagini  
   type: repeat, repeat-x, repeat-y, no-repeat */
```

STILURI ȘI CULORI

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {

        function draw() {
          const ctx =
document.getElementById("canvas").getContext("
2d");

          // create new image object to use as
pattern
          const img = new Image();
          img.src = "pattern.webp";
          img.onload = () => {
            // create pattern
            const ptrn = ctx.createPattern(img,
"repeat");
            ctx.fillStyle = ptrn;
            ctx.fillRect(0, 0, 400, 600);
          };
        }
        draw();
      }
    </script>
  </head>
  <body>
    <canvas id="canvas" width="400px" height="600px">
    </canvas>
  </body>
</html>
```



STILURI ȘI CULORI

- umbre

```
shadowOffsetX = float;  
// distanța orizontală de la obiect  
shadowOffsetY = float;  
// distanța verticală de la obiect  
shadowBlur = float;  
shadowColor = color;
```

TEXT

```
fillText(text, x, y [, maxWidth]);  
strokeText(text, x, y [, maxWidth]);
```

```
font = value; // default: 10px sans-serif  
textAlign = value; // start, end, left, right, center  
textBaseline = value; /* top, hanging, middle,  
                        alphabetic, ideographic, bottom */  
direction = value; // ltr, rtl, inherit
```

TEXT

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {

        function draw() {
          const ctx =
document.getElementById("canvas").getContext("
2d");
          ctx.font = "48px serif";
          ctx.textBaseline = "hanging";
          ctx.fillStyle = "pink";
          ctx.fillText("Hakuna Matata!", 10,
50);
        }
        draw();
      }
    </script>
  </head>
  <body>
    <canvas id="canvas" width="400"
height="300"></canvas>
  </body>
```

>

IMAGINI

```
const img = new Image(); // crează element img nou
img.addEventListener("load", () => {
    // execute drawImage statements here
}, false);
img.src = "myImage.png"; // source path
```

```
drawImage(image, x, y);
// desenează o imagine la poziția (x,y)
drawImage(image, x, y, width, height);
// + scalare
drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight);
/* ia din imaginea sursă un dreptunghi cu colțul din stânga
sus la (sx, sy) și cu lățimea sWidth și înălțimea sHeight
și o desenează în canvas la poziția (dx, dy), scalând-o */
```

CANVAS STATE

```
save(); // salvează starea canvasului  
restore(); // restaurează cel mai recent salvată stare
```

o stare de desen a canvasului constă din

- valorile atributelor: `strokeStyle`, `fillStyle`, `globalAlpha`, `lineWidth`, `lineDashOffset`, `shadowOffsetX`, `font` etc.
- transformările care au fost aplicate: `translate`, `rotate`, `scale`
- clipping path curent

TRANSFORMĂRI

```
translate(x, y);  
// mișcă canvasul și originea pe grid cu distanțele x și y  
rotate(angle); // rotirea canvasului în jurul originii  
scale(x, y); /* scalarea canvasului cu x și y unități;  
              valori nr. reale */  
transform(a, b, c, d, e, f);  
/* înmulțește matricea de transformare curentă cu  
   matricea dată prin argumente */
```

COMPOSITE

```
globalCompositeOperation = type;  
// tipul de compunere la desenarea unor figuri noi
```

- source-over, source-in, source-out
- destination-over, destination-in, destination-out
- lighter, copy, xor, multiply, screen, overlay, darken, lighten, color-dodge, color-burn, hard-light, soft-light
- difference, exclusion, hue, saturation, color, luminosity

CLIPPING

clipping path: mascarea unei zone din canvas

```
clip(); // path-ul construit devine clipping path-ul curent
```

- folosim clip() în loc de closePath() pentru a închide un path și a-l transforma într-o mască în loc să îl umplem (fill) sau conturăm (stroke)

CLIPPING

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {

        function draw() {
          const ctx =
document.getElementById("canvas").getContext("
2d");

          ctx.fillRect(0, 0, 150, 150);
          ctx.translate(75, 75);

          // Create a circular clipping path
          ctx.beginPath();
          ctx.arc(0, 0, 60, 0, Math.PI * 2,
true);
          ctx.clip();

          // draw background
          const lingrad =
ctx.createLinearGradient(0, -75, 0, 75);
          lingrad.addColorStop(0, "#232256");
          lingrad.addColorStop(1, "#143778");
```



ANIMAȚII

pași pentru desenarea unui frame:

1. clear canvas: putem folosi `clearRect()`
2. salvarea stării canvasului
3. desenarea figurilor de animat
4. restaurarea stării canvasului

ANIMAȚII

update-uri programate:

```
setInterval();  
setTimeout();  
requestAnimationFrame(callback);  
/* cere browserului să apeleze funcția callback pentru a  
   actualiza o animație înainte de următoarea redesenare */
```

ANIMAȚII

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script>
      window.onload = function() {

        function clock() {
          const now = new Date();
          const canvas =
document.getElementById("canvas");
          const ctx = canvas.getContext("2d");
          ctx.save();
          ctx.clearRect(0, 0, 150, 150);
          ctx.translate(75, 75);
          ctx.scale(0.4, 0.4);
          ctx.rotate(-Math.PI / 2);
          ctx.strokeStyle = "black";
          ctx.fillStyle = "white";
          ctx.fill();
          ctx.lineWidth = 8;
          ctx.lineCap = "round";

          // Hour marks
          ctx.save();
```



OPTIMIZĂRI



SVG

Scalable Vector Graphics

SVG

- limbaj de marcare bazat pe XML pentru descrierea de elemente grafice vectoriale 2D
- standard Web open bazat pe text, dezvoltat de W3C din 1999
- imaginile SVG sunt definite în fișiere XML și pot fi randate la orice dimensiune fără pierderea calității

SVG

- suportat de toate browserele principale
- bine integrat cu CSS, DOM, JavaScript
- interfață DOM; nu necesită extensii third-party
- dezvantaj: încărcarea unei imagini SVG poate fi lentă
- fișierele SVG pot fi create și cu aplicații de desenat vectoriale precum [Inkscape](#)

INGREDIENTE PRINCIPALE

- un document SVG constă din rădăcina `<svg>` și elemente pentru figuri de bază
- elemente pentru figuri geometrice (cercuri, dreptunghiuri) și curbe (simple și complexe)
- figurile pot fi grupate folosind elementul `<g>`
- SVG permite definirea de gradienti, rotații, efecte de filtrare, animații și interactivitatea cu JavaScript

SINTAXĂ

- fiind bazat pe XML, SVG e case-sensitive: atenție la majuscule în cazul elementelor și al atributelor
- valorile atributelor se scriu între ghilimele, chiar și cele numerice

EXEMPLU

```
<svg version="1.1" width="300" height="200" xmlns="http://www.  
    <rect width="100%" height="100%" fill="pink"></rect>  
    <circle cx="150" cy="100" r="80" fill="yellow"></circle>  
    <text x="150" y="125" font-size="60" text-anchor="middle"  
</svg>
```

EXEMPLU

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
  </head>
  <body>
    <svg version="1.1"
      width="300" height="200"
      xmlns="http://www.w3.org/2000/svg">

      <rect width="100%" height="100%"
fill="pink" />
      <circle cx="150" cy="100" r="80"
fill="yellow" />
      <text x="150" y="125" font-size="60"
text-anchor="middle" fill="black">SVG</text>
    </svg>
  </body>
</html>
```



PROPRIETĂȚI

- ordinea randării elementelor: elementele noi sunt adăugate peste elementele mai vechi
- codul SVG poate fi adăugat direct în pagini HTML

```

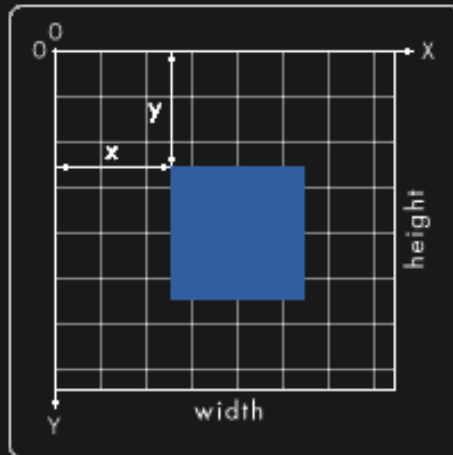
```

```
<iframe src="image.svg"></iframe>
```

```
<object data="image.svg" type="image/svg+xml"></object>
```


GRID - SPAȚIU DE COORDONATE

- o unitate în grid corespunde unui pixel
- originea gridului este în colțul din stânga sus, la coordonatele (0,0)
- pixeli? scalare?



FIGURI: RECT

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
  </head>
  <body>
    <svg width="200" height="250" version="1.1"
xmlns="http://www.w3.org/2000/svg">
      <rect x="10" y="10" width="100"
height="100" stroke="white" fill="transparent"
stroke-width="5"/>
      <rect x="60" y="100" rx="10" ry="10"
width="75" height="75" stroke="pink"
fill="transparent" stroke-width="5"/>
    </svg>
  </body>
</html>
```



FIGURI: CIRCLE

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
  </head>
  <body>
    <svg width="200" height="250" version="1.1"
xmlns="http://www.w3.org/2000/svg">
      <circle cx="100" cy="75" r="50"
stroke="pink" fill="transparent" stroke-
width="5"/>
      <ellipse cx="150" cy="75" rx="30" ry="15"
stroke="red" fill="transparent" stroke-
width="5"/>
    </svg>
  </body>
</html>
```



FIGURI: LINE

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
  </head>
  <body>
    <svg width="200" height="250" version="1.1"
xmlns="http://www.w3.org/2000/svg">
      <line x1="10" x2="50" y1="50" y2="90"
stroke="orange" stroke-width="5"/>
      <polyline points="60 110 65 120 70 115 75
130 80 125 85 140 90 135 95 150 100 145"
stroke="yellow" fill="transparent" stroke-
width="5"/>
    </svg>
  </body>
</html>
```



FIGURI: POLYGON

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
  </head>
  <body>
    <svg width="200" height="250" version="1.1"
xmlns="http://www.w3.org/2000/svg">
      <polygon points="50 60 55 80 70 80 60 90
65 105 50 95 35 105 40 90 30 80 45 80"
      stroke="yellow" fill="transparent" stroke-
width="5"/>
    </svg>
  </body>
</html>
```

>

PATHS

- elementul <path> - forma pathului este dată de atributul "d" ce conține o serie de comenzi (specificate prin câte o literă) și parametri pentru acele comenzi
- fiecare comandă are două variante: o literă majusculă pentru coordonate absolute în pagină, și una minusculă pentru coordonate relative

PATHS

comenzi pentru linii:

- Move To: M x y
- Line To: L x y
- Horizontal Line: H x. Vertical Line: V y
- Close Path: Z

PATHS

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
  </head>
  <body>
    <svg width="200" height="250" version="1.1"
xmlns="http://www.w3.org/2000/svg">
      <path d="M 10 10 h 80 v 80 h -80 Z"
fill="transparent" stroke="white"/>
    </svg>
  </body>
</html>
```

>

PATHS

comenzi pentru curbe:

- Curbe Béziere cubice: C x1 y1, x2 y2, x y
- Curbe Béziere cuadratice: Q x1 y1, x y
- Arce de cerc:

A rx ry x-axis-rotation large-arc-flag sweep-flag x y

FILLS & STROKES

folosind attributele

- *fill* și *stroke*
- *stroke-width*, *stroke-linecap*, *stroke-linejoin*,
stroke-dasharray, *stroke-dashoffset*

GRADIENȚI

folosind elementele

- `<linearGradient>` împreună cu noduri `<stop>` și attributele *offset*, *stop-color*, *stop-opacity*
- `<radialGradient>` împreună cu noduri `<stop>` și attribute pentru setarea **punctelor de centru și focale**
- se adaugă într-un element `<defs>` și trebuie să aibă un atribut *id*

PATHS

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
  </head>
  <body>
    <svg width="120" height="240"
version="1.1"
xmlns="http://www.w3.org/2000/svg">
      <defs>
        <radialGradient id="RadialGradient1">
          <stop offset="0%" stop-color="red" />
          <stop offset="100%" stop-color="yellow"
/>
        </radialGradient>
        <radialGradient id="RadialGradient2"
cx="0.25" cy="0.25" r="0.25">
          <stop offset="0%" stop-color="red" />
          <stop offset="100%" stop-color="yellow"
/>
        </radialGradient>
      </defs>
```



ALTE ELEMENTE

- patterns
- text
- clip paths și masks
- imagini

TRANSFORMĂRI

atributul *transform* cu valorile:

- `translate()`
- `rotate()`
- `skewX()` și `skewY()`
- `scale()`
- transformări complexe cu `matrix()`

CANVAS VS. SVG

