Tehnici Web
CURSUL 12

Semestrul I, 2019-2020

Carmen Chirita

https://sites.google.com/site/fmitehniciweb/

Punerea pe net a aplicației

Putem folosi Heroku: https://www.heroku.com/
Cum se adauga o aplicatie pe Heroku:
https://devcenter.heroku.com/articles/deploying-nodejs
Pasi principali:
- va inregistrati pe Heroku
- se instaleaza Heroku CLI
- se adauga in package.json versiunea de node si scriptul de start
- se face un git repository pentru proiect
- se da deploy la aplicatie pe Heroku

Cum facem un repository pe git

1) Ne facem un cont pe https://github.com/
2) In consola, **ne ducem in folderul aplicatiei** si dam urmatoarele comenzi:
git init
git config --global user.email "mailul_meu@ceva.com"
git add .

3) Facem un fisier .gitignore in folderul aplicatiei in care trecem:
/node_modules
Ce se trece in .gitignore va fi ignorat de git (nu va fi adaugat in repository)
4) Dam comanda
git commit -m "Adaugat aplicatie"

Socket.IO

https://socket.io/ https://socket.io/docs/

- folosim pentru a transmite in timp real evenimente de la server catre pagini - actualizand astfel pagina in timp real

Exemple de cazuri de utilizare:
- afisarea listei de utilizatori logati
- un chat
- un joc multiplayer (fiecare utilizator trebuie sa vada miscarile/mutarile celorlalti utilizatori)

Socket.IO creeaza o conexiune activa intre client si server, prin care se transmit mesaje de tip eveniment (este diferit de cazul clasic in care pagina odata trimisa la client, nu mai poate primi notificari de la el, ci poate doar cere informatii (de exemplu prin AJAX)

Socket.IO - server

```
const express = require('express');
var app = express();
const http=require('http')
const socket = require('socket.io');
var conexiune;
const server = new http.createServer(app);  //creez un obiect de tip server
                                                din serverul express
var  io= socket(server)  //initializez obiectul de tip socket io
io = io.listen(server);  //asculta pe acelasi port ca si serverul
io.on("connection", (socketC) => {  //la conectare (evenimentul
"connection")
    conexiune=socketC
    console.log("Conectare!");//afisez mesaj de confirmare a conectarii
    socketC.on('disconnect', () => conexiune=null;
console.log('Deconectare')});//deconectare
});
app.listen(8080);
```

Emitere eveniment pe socket

Emiterea evenimentului care va fi captat de client (pentru actualizarea dinamica a paginii)

```
app.post('/cerere_emitere', function(req, res) {
    if(conexiune){ // daca am deja realizata conexiunea pe socket
      //emit un eveniment custom, numit "notificare"
       conexiune.emit("notificare",’salut’);
    }
   res.render('cerere_emitere');

});
```

Emitere pentru toți clientii conectați:
io.sockets.emit(‘notificare’,’salut’);

Captare eveniment pe socket

```
<script src="../../node_modules/socket.io-client/dist/socket.io.js"></script>
<script>
    socketUrl = "url-ul aplicatiei";
    if(document.location.href.indexOf("localhost") != -1) {
       socketUrl = "http://127.0.0.1:8080";  //portul pe care ruleaza serverul
    }
    const socket = io(socketUrl, {reconnect: true});  //realizez conexiunea
    socket.on("notificare", (data) => {  //la captarea evenimentului emis de
server
       alert(data)
    });
</script>
```

Scalable Vector Graphics SVG

https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial

SVG

- componentele primitive ale imaginii sunt forme geometrice, care sunt descrise matematic; aceasta asigura independenta de rezolutie, scalabilitatea

- gramatica XML pentru grafica

Scalable Vector Graphics SVG

- fisierele SVG pot fi editate cu editoare specializate
  (Adobe Illustrator, Inkscape), dar si ca fisiere XML

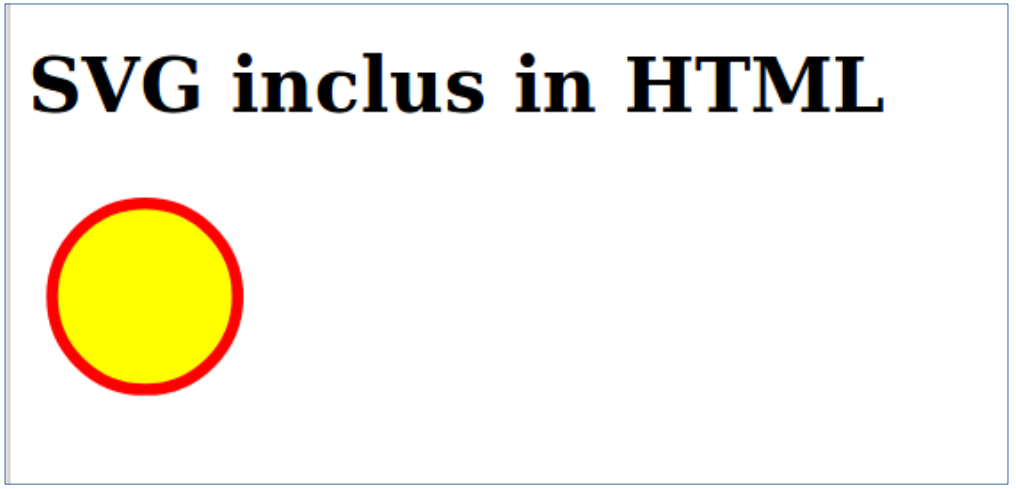
- fisierele SVG pot fi incluse in fisierele HTML folosind <img>
  <img src="star-svg.svg" id="imgsvg" alt="stea">

- imaginile (codul) SVG pot fi incluse direct in fisiere HTML

```
<svg  width="450" height="500"   id="elsvg">
...
</svg>
```

Exemplu

```
<body>
<h1>SVG inclus in HTML</h1>

<svg width="100" height="100"> //container pentru grafica
   <circle cx="50" cy="50" r="40" stroke="red" stroke-width="5" fill="yellow" />
</svg>

</body>
```



Elemente SVG predefinite

<rect>

<circle>

<ellipse>

<line>

<polyline>

<polygon>

<path>

<text>

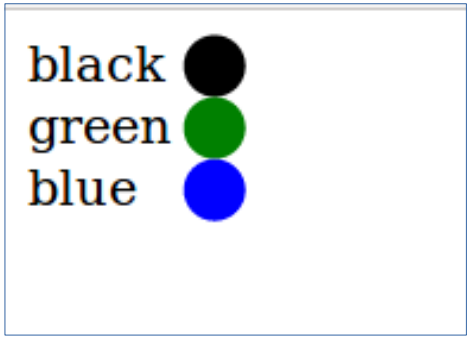
Elementele SVG <defs>, <use>

SVG permite ca obiectele grafice să fie definite pentru o reutilizare ulterioară.

Elementele de referință se vor defini în interiorul unui element <defs> și pot fi referite cu <use>.

De exemplu, <defs> poate fi utilizat la crearea gradientilor

```
<svg width="500" height="600">

 <defs>

    <circle id="Port" cx="10" cy="0" r="10"/>

 </defs>

  <text y="15">black</text>
  <use x="50" y="10" href="#Port" style="fill: black;"/>
  <text y="35">green</text>
  <use x="50" y="30" href="#Port" style="fill: green;"/>
  <text y="55">blue</text>
  <use x="50" y="50" href="#Port" style="fill: blue;"/>


</svg>
```



SVG Rectangle - <rect>

utilizat pentru a crea un dreptunghi și variații ale unei forme de dreptunghi

```
<body>

<svg width="400" height="200">
  <rect width="200" height="100" style="fill:green;  stroke-width:3;  stroke:red" />
</svg>

</body>
```

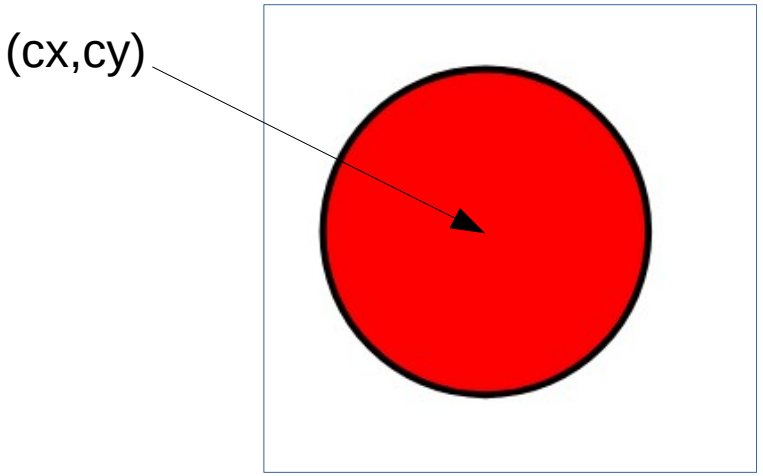
Atribute specifice: width , height, x, y
Atributele rx și ry //colturi rotunjite
Atributul style: defineste stil CSS

Proprietati CSS:

fill: culoarea continutului
stroke: culoarea conturului
stroke-width: grosimea conturului
fill-opacity:0-1, stroke-opacity:0-1,opacity:0-1

SVG Circle - <circle> (deseneaza un cerc)

```
<circle cx="100" cy="100" r="70"
 style="stroke:black; stroke-width:3; fill:red" />
```



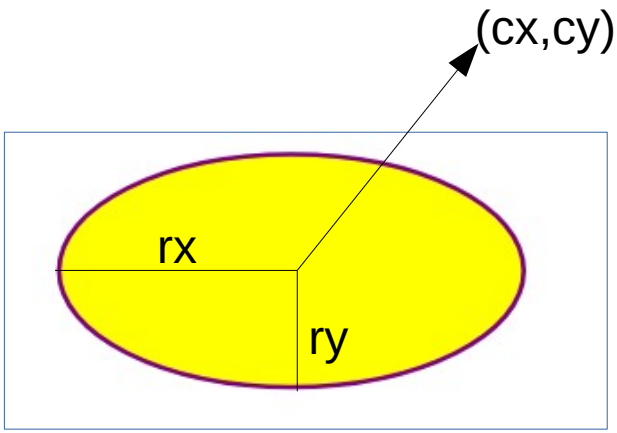
Atribute specifice:
cx, cy //coordonatele centrului
r //raza cercului

Proprietati CSS:

fill: culoarea continutului
stroke: culoarea conturului
stroke-width: grosimea conturului
fill-opacity:0-1, stroke-opacity:0-1,opacity:0-1

SVG Ellipse - <ellipse> (deseneaza o elipsa)

```
<ellipse cx="200" cy="80" rx="100" ry="50"
style="fill:yellow; stroke:purple; stroke-width:2" />
```



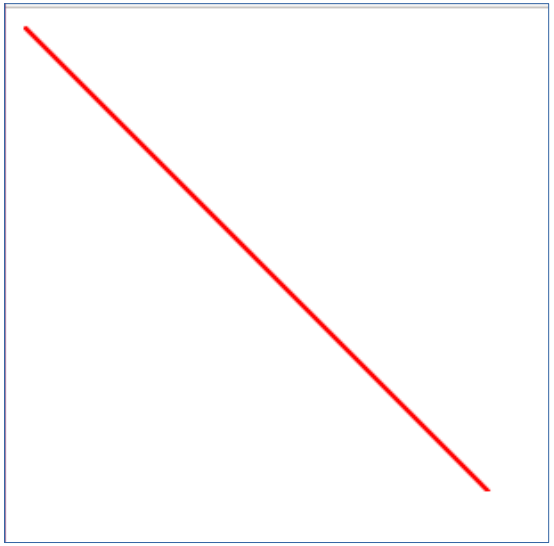
Atribute specifice:
cx, cy //coordonatele centrului elipsei
rx //raza orizontala
ry //raza verticala

Proprietati CSS:

fill: culoarea continutului
stroke: culoarea conturului
stroke-width: grosimea conturului
fill-opacity:0-1, stroke-opacity:0-1,opacity:0-1

SVG Line - <line> (deseneaza o linie)

```
<line x1="0" y1="0" x2="200" y2="200"
style="stroke:rgb(255,0,0);stroke-width:2" />
```



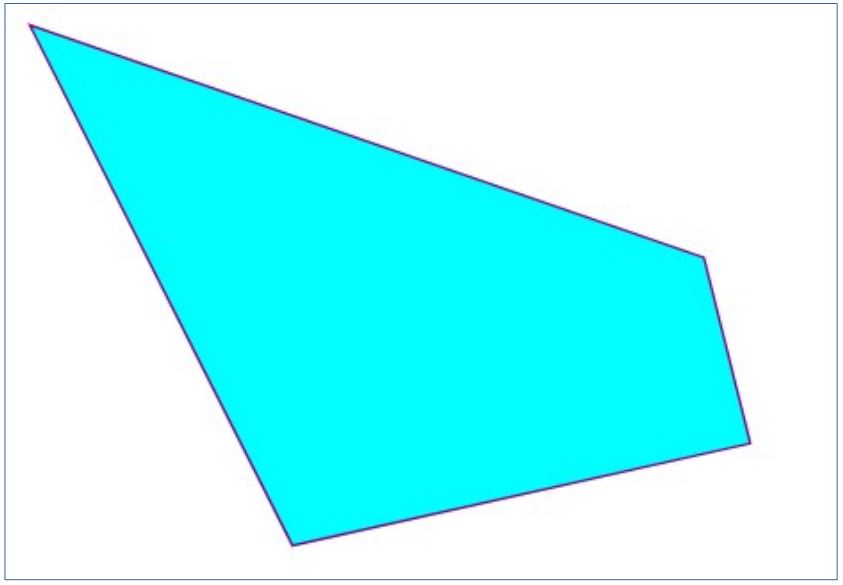
Atribute specifice:
x1, y1 //coordonatele de inceput
x2, y2 //coordonatele de sfarsit

Proprietati CSS:
stroke: culoarea liniei
stroke-width: grosimea liniei

SVG Polygon - <polygon>

este utilizat pentru a crea o imagine care conține cel puțin trei laturi.

```
<polygon points="10,10 300,110 320,190 123,234"
style="fill:cyan; stroke:purple; stroke-width:1" />
```

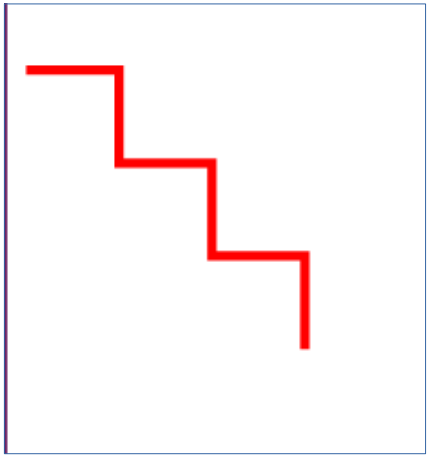


Atribute specifice:
points : //coordonatele varfurilor

SVG Polyline - <polyline>

este utilizat pentru a crea orice formă care constă numai din linii drepte

```
<polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160" style="fill:white; stroke:red; stroke-width:4" />
```

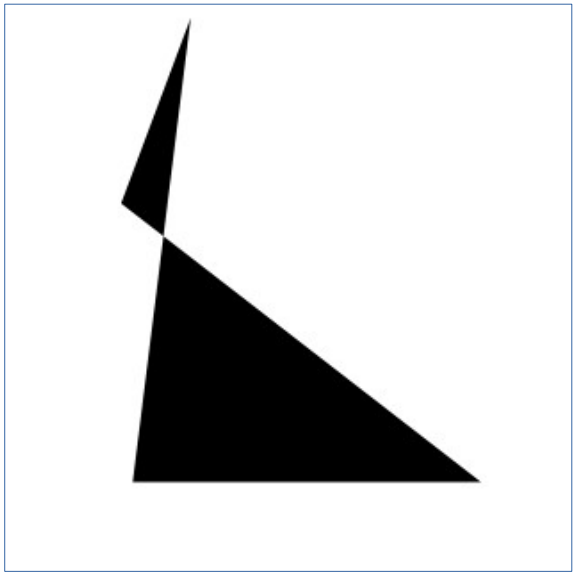


Atribute specifice:
points : //coordonatele varfurilor

SVG Path - <path>

este utilizat pentru a crea forme complexe combinand linii, arcuri, curbe, etc.

```
<path d="M100 0 L75 200 L225 200 L70 80 Z" />
```



Comenzi

M = moveto

L = lineto

Z = closepath

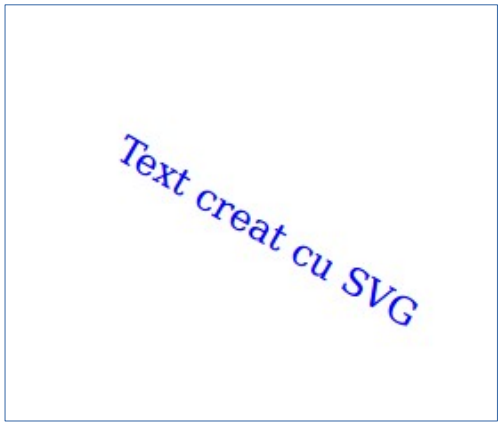
H = horizontal lineto

V = vertical lineto

A = elliptical Arc

SVG Text - <text> (defineste un text)

```
<text x="50" y="50" fill="blue" transform="rotate(30 20,40)">Text creat cu SVG</text>
```

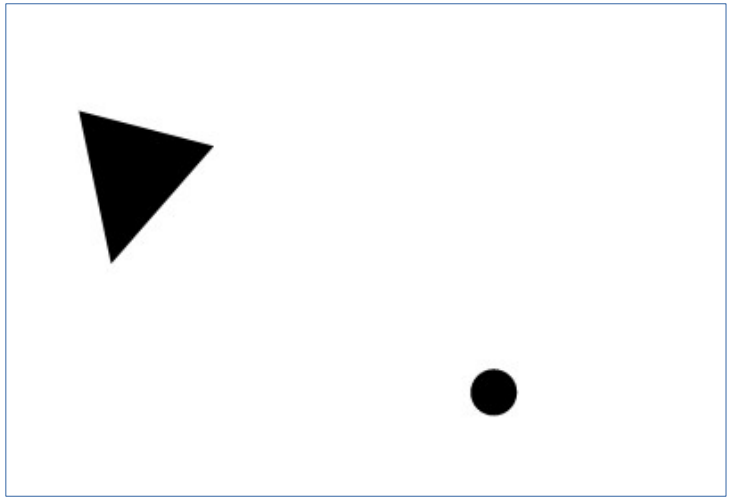


Atribute specifice:
x,y : //coordonatele de inceput

Animatie SVG

<animate>
<animateTransform>
<animateMotion>

```
<polygon points="60,30 90,90 30,90">
      <animateTransform attributeName="transform"
                        type="rotate"
                        from="0 60 70"
                        to="360 60 70"
                        dur="10s"
                        repeatCount="4"/>
   </polygon>
```



```
<circle cx="230" cy="32" r="10" >
<animateMotion
      path= "M0,0 L50,132 L-50, 132 L0,0"
       begin="3s" dur="10s"
       repeatCount="indefinite" />
</circle>
```

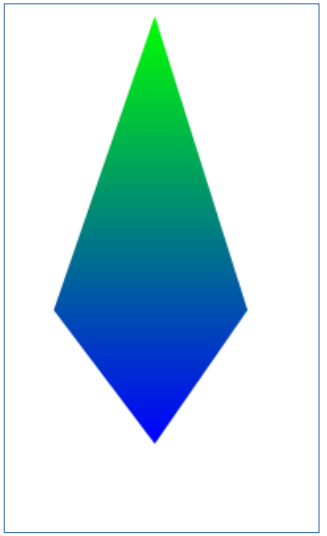
svg.html

Gradienti in SVG- <linearGradient>

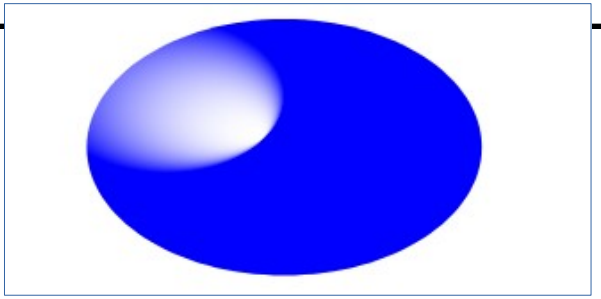
```
<svg>
<defs> //defineste elemente SVG care vor
fi utilizate ulterior

<linearGradient id="myLG“
   x1="0%" y1="0%"   x2="0%" y2="100%“
spreadMethod="pad">
<stop offset="0%"   stop-color="#00ff00" stop-opacity="1"/>
<stop offset="100%" stop-color="#0000ff" stop-opacity="1"/>
</linearGradient>

</defs>

<path d="M 230,32  L181,175 L 230,240
                  L 275,175 L 230,32 "
     style="fill:url(#myLG)“ />
</svg>
```

Gradienti:

Pe verticala: x1 = x2
Pe orizontala: y1 = y2
Unghiulari: x1≠x2, y1≠y2



Gradienti in SVG -<radialGradient>



```
<svg height="150" width="500">

 <defs>

    <radialGradient id="grad1" cx="20%" cy="30%" r="30%"
fx="50%" fy="50%">
      <stop offset="0%" style="stop-color:rgb(255,255,255);
stop-opacity:0" />
      <stop offset="100%" style="stop-color:rgb(0,0,255);
stop-opacity:1" />
    </radialGradient>

  </defs>

  <ellipse cx="200" cy="70" rx="85" ry="55" fill="url(#grad1)" />

</svg>
```

CANVAS

Elementul HTML <canvas> este folosit pentru a desena grafica, cu ajutorul JavaScript.

Elementul <canvas> este doar un container pentru grafică (este necesar un script pentru a desena grafica).

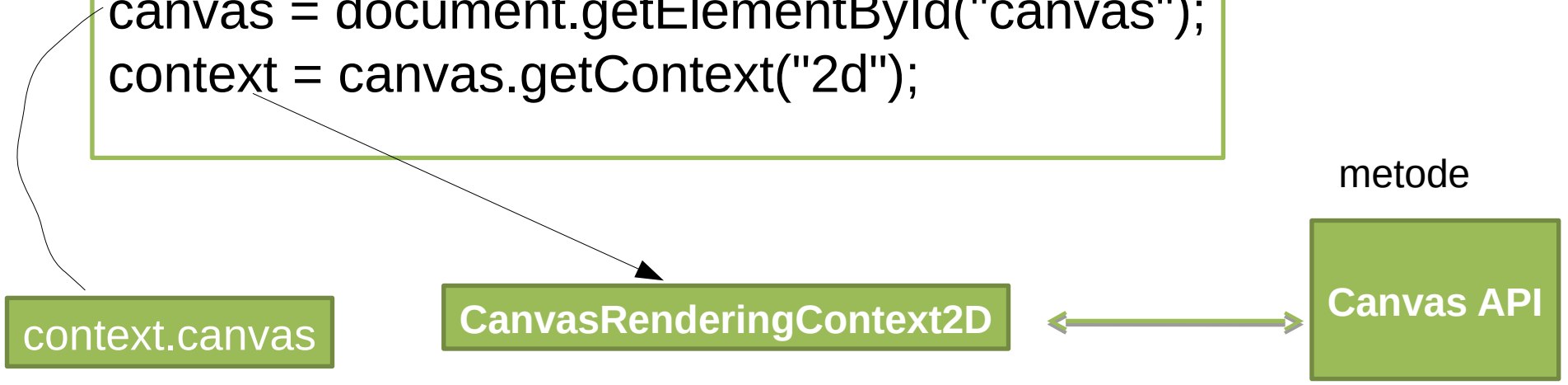
Canvas are mai multe metode pentru a desena linii, figuri, cercuri, text și inserare de imagini.

CANVAS

```
#canvas {border: 2px solid black
body {background-color: red;}
```

```
<canvas id=“canvas” width=“500” height=“400”>
 Continut alternativ </canvas>
```

```
canvas = document.getElementById("canvas");
context = canvas.getContext("2d");
```



https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement

http://www.w3.org/TR/2010/WD-html5-20100624/the-canvas-element.html

HTMLCanvasElement metoda toDataUrl

Returnează un URL de date ce conține o reprezentare a imaginii în formatul specificat de parametrul de tip (implicit png).

canvas.toDataURL(type, encoderOptions)

// type poate fi: image/png // “image/jpeg”, “image/webp”
//encoderOptions: calitatea imaginii (intre 0-1)

```
<body>
<div id="imgcanvas">
 <h1>Imagine creata cu canvas </h1>
 </div>
</body>
```

Canvas-ul poate fi folosit ca o imagine bitmap

```
var url = canvas.toDataURL();
 var newImg = document.createElement("img");
 newImg.src = url;
 var newImgParent = document.getElementById("imgcanvas");
 newImgParent.appendChild(newImg);
```

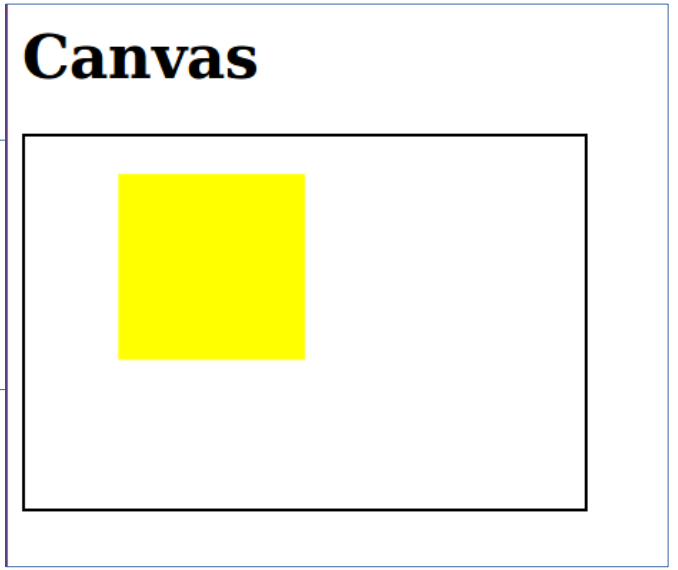
Canvas: fillStyle și fillRect()

fillStyle //stilul folosit în interiorul formelor
poate fi culoare /gradient/ patern
fillRect(x,y,width,height) //deseneaza un dreptunghi

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

ctx.fillStyle = "yellow";
ctx.fillRect(50, 20, 100, 100);
```

```
<style type="text/css" >
#canvas {border: 2px solid black;}
</style>
```

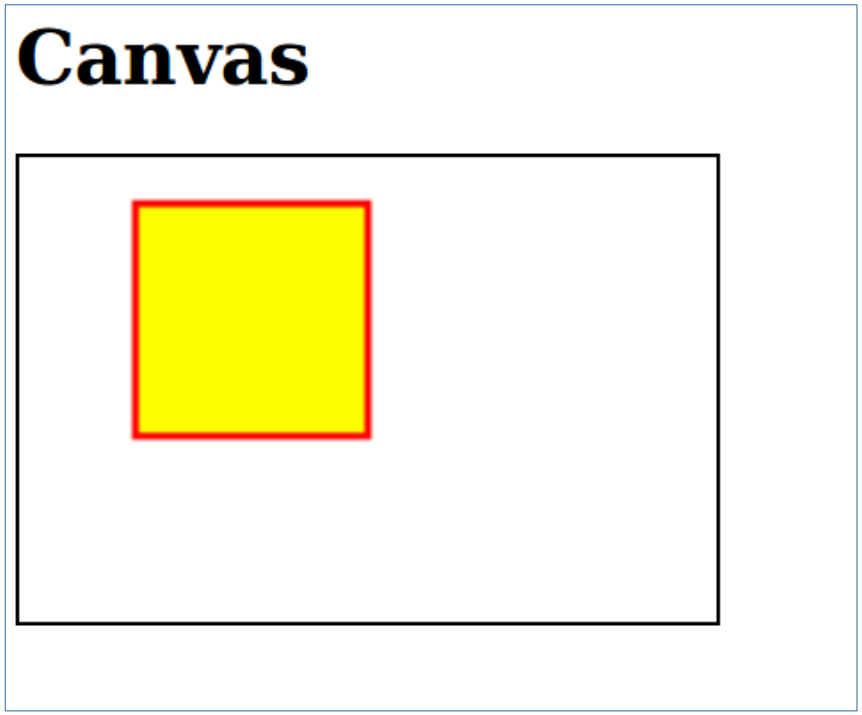
```
<body>
<h1> Canvas </h1>
<canvas id="canvas" width="300" height="200"> alt</canvas>
</body>
```



Stroke (conturul) si Fill (continutul)

```
ctx.strokeStyle = "red";
ctx.lineWidth = 3;
ctx.strokeRect(50, 20, 100, 100);

ctx.fillStyle = "yellow";
ctx.fillRect(50, 20, 100, 100);
```

strokeStyle
lineWidth
fillStyle



strokeRect(x,y,width,height)
fillRect(x,y,widht,height)
clearRect(x,y,widht,height)

(x,y) coltul stanga sus

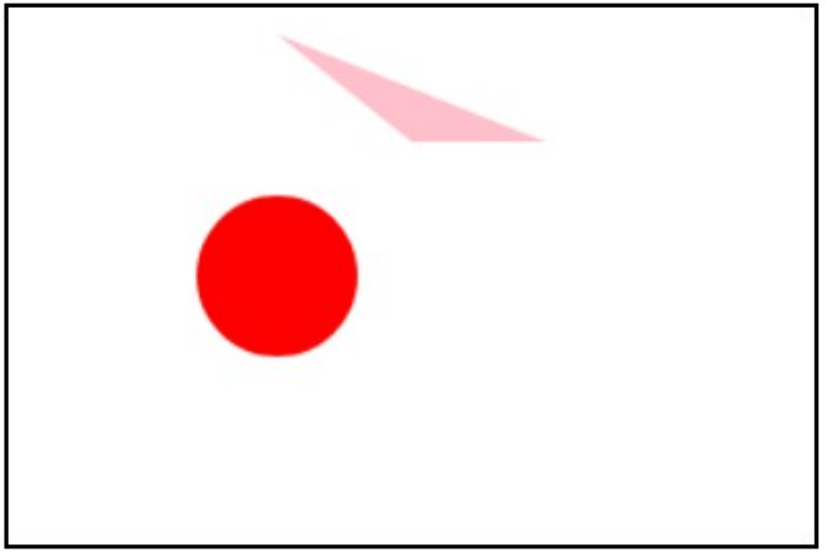
Desenarea figurilor

beginPath() //incepe desenarea unei noi figuri
closePath() //inchide figura de la punctul curent
la punctul de plecare

Desenarea unei linii
moveTo(x,y) //coordonatele de inceput
lineTo(x,y) // coordonatele de sfarsit
lineCap //stilul capetelor liniei ("butt|round|square")
stroke() //deseneaza efectiv linia

Desenarea unei cerc
beginPath()
arc(x,y,r,startangle,endangle) //creeaza un cerc (arc de cerc)
stroke() /* fill() //deseneaza doar conturul sau
coloreaza si continutul

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

ctx.beginPath();
ctx.moveTo(100,10);
ctx.lineTo(150,50);
ctx.lineTo(200,50);
ctx.lineTo(100,10);
ctx.lineCap = "round";
ctx.fillStyle = "pink";
ctx.fill();
ctx.closePath();


ctx.beginPath();
ctx.arc(100, 100, 30,  0, 2* Math.PI);
ctx.fillStyle = "red";
ctx.fill();
```

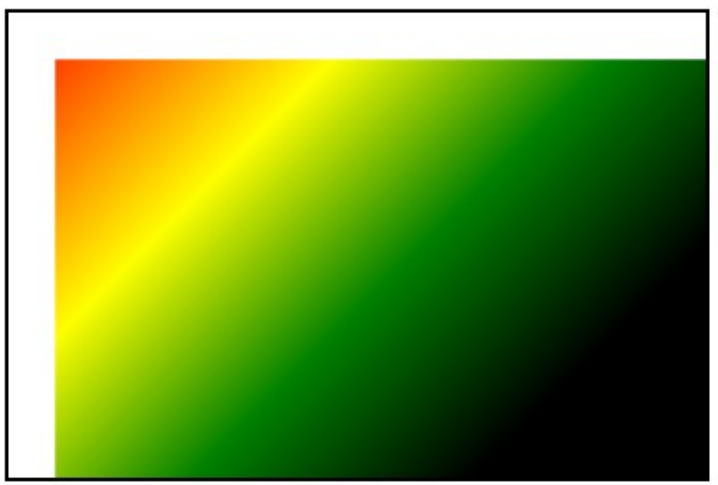


```
<body>
<h1> Canvas </h1>
<canvas id="canvas" width="300" height="200"> alt</canvas>
</body>
```

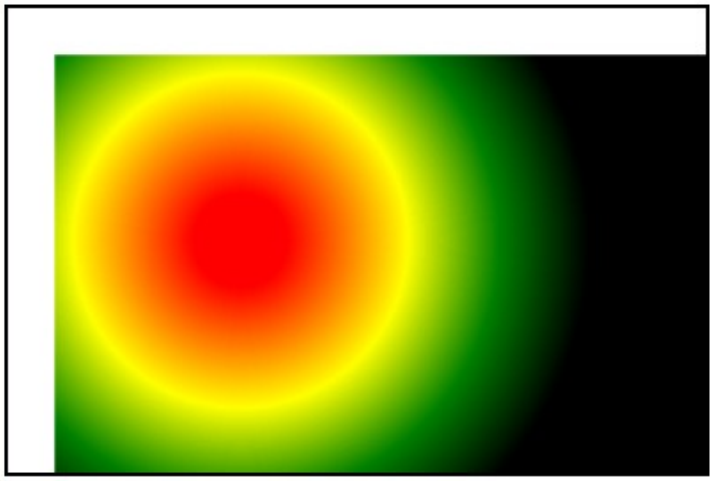
linearGadient, radialGradient

createLinearGradient(x0,y0,x1,y1)
createRadialGradient(x0,y0,r0,x1,y1,r1)
addColorStop()

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

var fade = ctx.createLinearGradient(0, 0, 200, 200);
fade.addColorStop(0, "red");
fade.addColorStop(0.4, "yellow");
fade.addColorStop(0.7,"green");
fade.addColorStop(1.0, "black");

ctx.fillStyle = fade;
ctx.fillRect(20, 20, 300, 300);
```



```
var fade = ctx.createRadialGradient(100,100,20,100,100,150);
```



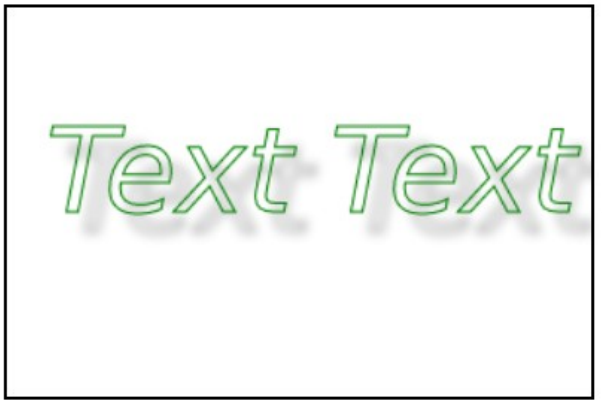
Text și shadow

font
strokeText(text,x,y)
fillText(text,x,y)

shadowOffsetX
shadowOffsetY
shadowBlur
shadowColor

```
ctx.shadowOffsetX = 10;
ctx.shadowOffsetY = 10;
ctx.shadowBlur    = 4;
ctx.shadowColor   = "#666666";
ctx.fillStyle ="green";
//ctx.strokeStyle = "green";

ctx.font = "italic 60px sans-serif";
ctx.lineWidth = 1;
ctx.fillText("Text Text",20, 105);
//ctc.strokeText("Text Text",20, 105);
```

fill text



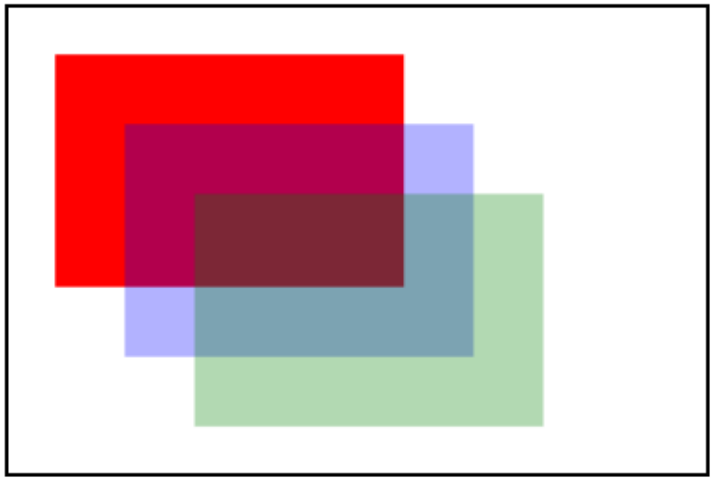
stroke text

CANVAS: globalAlpha si globalCompositeOperation

globalAlpha()
transparenta (intre 0-1=opac)

globalCompositeOperation
modul de amestecare a culorilor

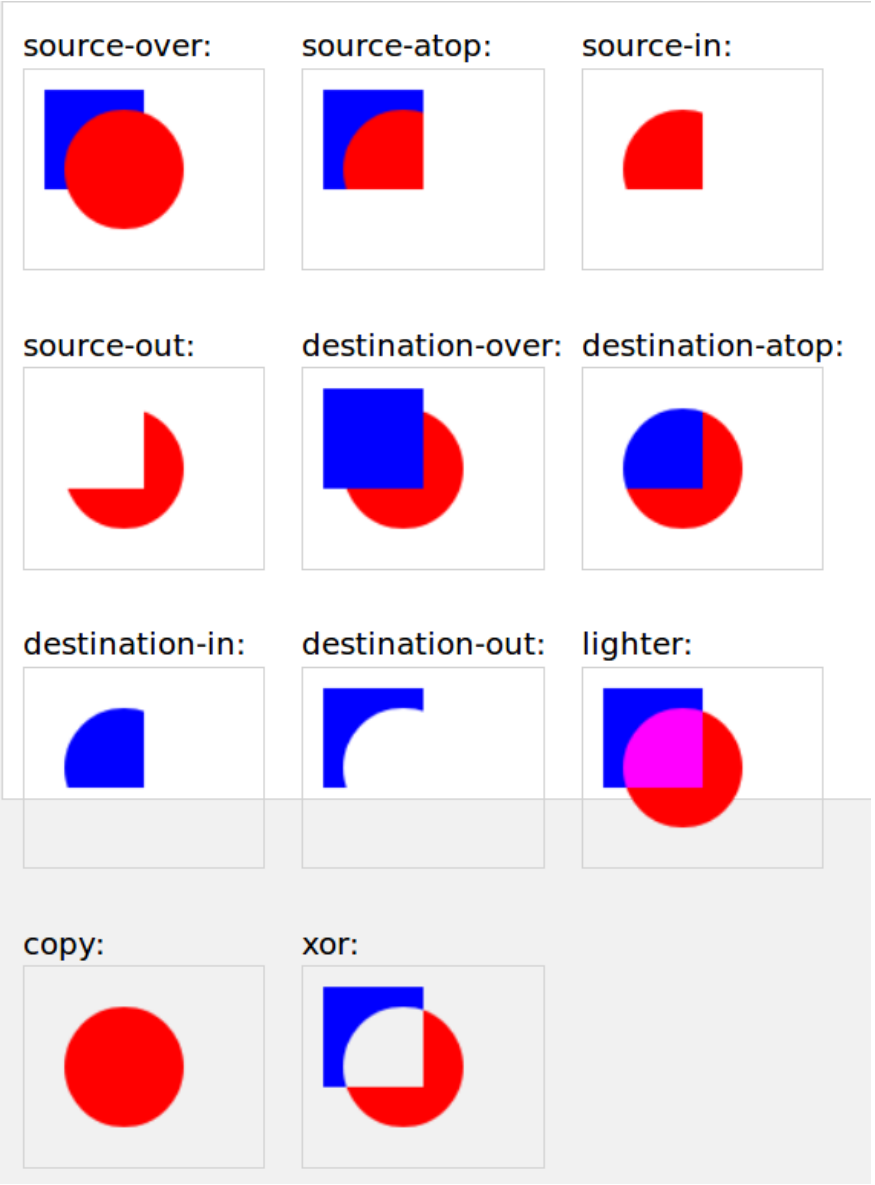
```
ctx.fillStyle = "red";
ctx.fillRect(20, 20, 150, 100);

ctx.globalAlpha = 0.3;
ctx.fillStyle = "blue";
ctx.fillRect(50, 50, 150, 100);
ctx.fillStyle = "green";
ctx.fillRect(80, 80, 150, 100);
```



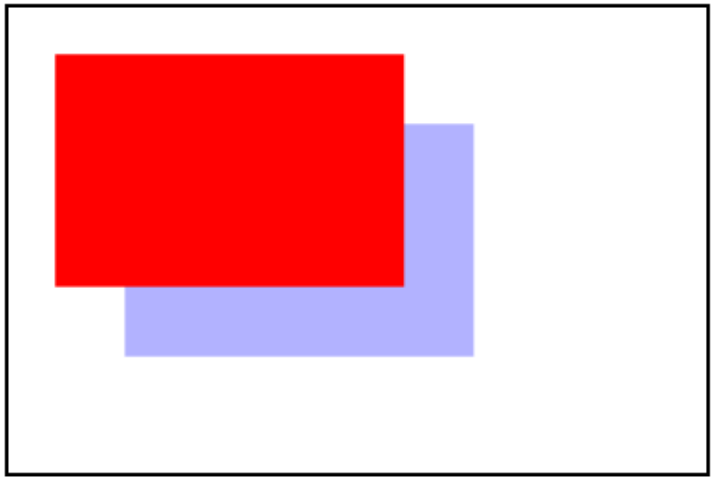
globalCompositeOperation
modul de amestecare a culorilor

specifica modul în care o imagine sursă (nouă)
este plasata pe o imagine destinație (existentă).

implicit:source-over



```
ctx.fillStyle = "red";
ctx.fillRect(20, 20, 150, 100);

ctx.globalAlpha = 0.3;
ctx.globalCompositeOperation="destination-over";
ctx.fillStyle = "blue";
ctx.fillRect(50, 50, 150, 100);
```

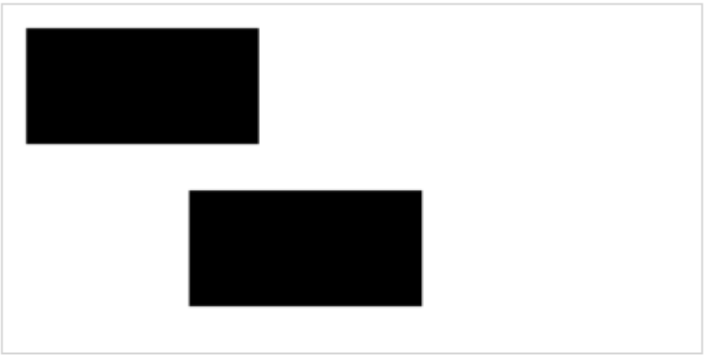


Transformari si salvarea contextului

```
ctx.translate(x,y)
ctx.rotate(degrees*Math.PI/180)
ctx.scale(factor)

se aplica tuturor
elementelor desenate
```

```
ctx.save()
ctx.restore()
```

```
ctx.fillRect(10, 10, 100, 50);
ctx.translate(70, 70);
ctx.fillRect(10, 10, 100, 50);
```



Desenarea imaginilor

drawImage(image,x0,y0,width,height)

fisier.html

```
<canvas id="canvas" width="300" height="200"> alt</canvas>
<p style = "display:none">
<img id="imagine" src="iarna.jpg" width="150" height="90">
</p>
```

fisier.js

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var image = document.getElementById('imagine');
 ctx.drawImage(image, 0, 30);
```