Grafika rastrowa w HTML5

Element CANVAS

- Teksty
- Obrazy
- Animacje
- Interakcja
- Ramka formatowana za pomocą CSS

```
<canvas id="TheWall" width="600" height="400"
 style="border:1px solid #c3c3c3;"
</canvas>
```

Rysowanie na kanwie

- Współrzędne x, y; środek układu współrzędnych: górny, lewy narożnik

```
<script>
var canvas = document.getElementById("TheWall");
var ctx = canvas.getctx("2d");
ctx.fillStyle = "red"; // "#FF0000"
ctx.fillRect(10,50,50,30); //start -(1,50); szerokość, wysokość - (50,50)
</script>
```


Rysowanie napisów

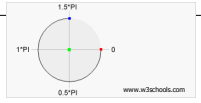
```
<script>
var canvas = document.getElementById("TheWall");
var ctx = canvas.getctx("2d");
ctx.font = "30px Comic Sans MS";
ctx.fillStyle = "gray";
ctx.textAlign = "center";
ctx.fillText("The Wall", canvas.width/2, 30);
</script>
```

```
ctx.strokeText("The Wall", canvas.width/2, 30);
```

The Wall

Rysowanie łuku i linii

```
<script>
var canvas = document.getElementById("TheWall");
var ctx = canvas.getctx("2d");
ctx.beginPath();
ctx.moveTo(10,50);
ctx.lineTo(10+50,50+0);
ctx.lineTo(10+50,50+30);
ctx.lineTo(10,50+30);
ctx.closePath();
//ctx.lineTo(10,50);
ctx.stroke();

ctx.beginPath();
ctx.arc(canvas.width/2,350,15,0,2*Math.PI); //x, y, r, start, stop
ctx.stroke();

</script>
```



Łączenie linii

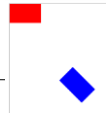
```
<script>
var canvas = document.getElementById("TheWall");
var ctx = canvas.getctx("2d");
ctx.lineWidth = 5;
ctx.lineJoin = 'miter';
//ctx.lineJoin = 'round';
//ctx.lineJoin = 'bevel';
ctx.beginPath();
ctx.moveTo(10,50);
ctx.lineTo(10+50,50+0);
ctx.lineTo(10+50,50+30);
ctx.lineTo(10,50+30);
ctx.closePath();
ctx.stroke();


</script>
```

Transformacje - przesunięcie

```
<script>
var canvas = document.getElementById("TheWall");
var ctx = canvas.getctx("2d");

ctx.fillStyle = "red";
ctx.fillRect(0,0,50,30);

ctx.translate(100, 100);

ctx.fillStyle = "blue";
ctx.fillRect(0,0,50,30);

</script>
```

Transformacje - obrót

```
<script>
var canvas = document.getElementById("TheWall");
var ctx = canvas.getctx("2d");

ctx.fillStyle = "red";
ctx.fillRect(0,0,50,30);

ctx.translate(100, 100);
ctx.rotate(Math.PI/4);

ctx.fillStyle = "blue";
ctx.fillRect(0,0,50,30);

</script>
```



Transformacje - skalowanie

```
<script>
var canvas = document.getElementById("TheWall");
var ctx = canvas.getctx("2d");

ctx.fillStyle = "red";
ctx.fillRect(0,0,50,30);

ctx.translate(100, 100);
ctx.scale(2, 0.5);

ctx.fillStyle = "blue";
ctx.fillRect(0,0,50,30);

ctx.scale(-1, 1);

ctx.font = "30px Comic Sans MS";
ctx.fillStyle = "gray";
ctx.textAlign = "center";
ctx.fillText("The Wall", 0, 0);

</script>
```

Transformacja własna

```
ctx.transform(a, b, c, d, e, f);
ctx.setTransform(a, b, c, d, e, f);
```

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

```
x' = a*x + c*y + e;
y' = b*x + d*y + f;
```

Transformacja własna

- przesunięcie

```
x' = a*x + c*y + e;
y' = b*x + d*y + f;
```

```
x' = x + e ;       // a=1, c=0
y' = y + f;        //b=0, d=1
ctx.transform(1, 0, 0, 1, e, f);
```

- skalowanie

```
x' = a*x;                 //c=0, e=0
y' = d*y;                 //b=0, f=0
ctx.transform(a, 0, 0, d, 0, 0);
```

- obrót

```
x' = cos(alpha)*x - sin(alpha)*y; //e=0
y' = sin(alpha)*x + cos(alpha)*y; //f=0
ctx.transform(cos(alpha), sin(alpha), -sin(alpha), cos(alpha), 0, 0);
```

- skrzywienie (shear)

```
x' = x + c*y;          //a=1, e=0; skrzywienie pionowe c=0
y' = b*x + y;          //d=1, f=0; skrzywienie poziome b=0
ctx.transform(1, b, c, 1, 0, 0);
```

Transformacja własna – skrzywienie poziome

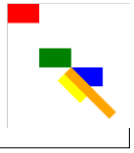
```
x' = a*x + c*y + e;
y' = b*x + d*y + f;
```

```
ctx.transform(1, 0, 0.75, 1, 0, 0);
ctx.font = "30px Comic Sans MS";
ctx.fillStyle = "gray";
ctx.textAlign = "center";
ctx.fillText("The Wall", canvas.width/2, 30);
```

The Wall

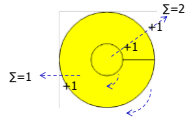
Stos stanu kontekstu

```
<script>
var canvas = document.getElementById("TheWall");
var ctx = canvas.getctx("2d");

ctx.fillStyle = "red";
ctx.fillRect(0,0,50,30);
ctx.save();
ctx.translate(100, 100);
ctx.fillStyle = "blue";
ctx.fillRect(0,0,50,30);
ctx.save();
ctx.rotate(Math.PI/4);
ctx.fillStyle = "yellow";
ctx.fillRect(0,0,50,30);
ctx.save();
ctx.scale(2, 0.5);
ctx.fillStyle = "orange";
ctx.fillRect(0,0,50,30);
ctx.restore();
ctx.restore();
ctx.fillStyle = "green";
ctx.fillRect(0,0,-50,-30);

</script>
```

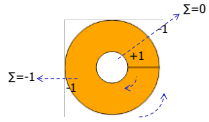


Reguła niezerowego skręcenia

```
ctx.fillStyle = 'yellow';
ctx.beginPath();
ctx.arc(75, 75, 75, 0, 2*Math.PI);
ctx.arc(75, 75, 25, 0, 2*Math.PI);
ctx.fill();
ctx.stroke();
```

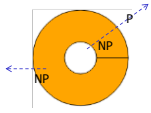



```
ctx.fillStyle = 'orange';
ctx.beginPath();
ctx.arc(75, 75, 75, 0, 2*Math.PI, true);
ctx.arc(75, 75, 25, 0, 2*Math.PI);
ctx.fill();
ctx.stroke();
```



Reguła parzysty-nieparzysty

```
ctx.fillStyle = 'orange';
ctx.beginPath();
ctx.arc(75, 75, 75, 0, 2*Math.PI);
ctx.arc(75, 75, 25, 0, 2*Math.PI);
ctx.fill("evenodd");
ctx.stroke();
```



Wypełnianie gradientem

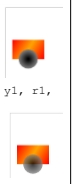
```
var grd = ctx.createLinearGradient(10,50,10+50,50+30);
grd.addColorStop(0,"red");
grd.addColorStop(0.5,"orange");
grd.addColorStop(1,"red");

ctx.fillStyle = grd;
ctx.fillRect(10,50,50,30);

grd = ctx.createRadialGradient(canvas.width/2,350, 1,
canvas.width/2,350, 15); //x1, y1, r1, x2, y2, r2
grd.addColorStop(0,"black");
grd.addColorStop(1,"gray");

ctx.fillStyle =grd;

ctx.beginPath();
ctx.arc(canvas.width/2,350,15,0,2*Math.PI); //x, y, r, start, stop
ctx.fill();
```

Cień

```
var grd = ctx.createLinearGradient(10,50,10+50,50+30);
grd.addColorStop(0,"red");
grd.addColorStop(0.5,"orange");
grd.addColorStop(1,"red");

ctx.save();

ctx.shadowColor = 'gray';
ctx.shadowBlur = 5;
ctx.shadowOffsetX = 5;
ctx.shadowOffsetY = 5;

ctx.fillStyle = grd;
ctx.fillRect(10,50,50,30);

ctx.restore();

ctx.lineWidth = 2;
ctx.strokeRect(10,50,50,30);
```

Przezroczystość

```
var grd = ctx.createLinearGradient(10,50,10+50,50+30);
grd.addColorStop(0,"red");
grd.addColorStop(0.5,"orange");
grd.addColorStop(1,"red");

ctx.fillStyle = grd;
ctx.fillRect(10,50,50,30);
ctx.stroke();

grd = ctx.createRadialGradient(35, 80, 1, 35,80, 15); //x1, y1, r1,
x2, y2, r2
grd.addColorStop(0,"black");
grd.addColorStop(1,"gray");

ctx.fillStyle =grd;
ctx.globalAlpha = 0.75;
ctx.beginPath();
ctx.arc(35,80,15,0,2*Math.PI); //x, y, r, start, stop
ctx.fill();
```

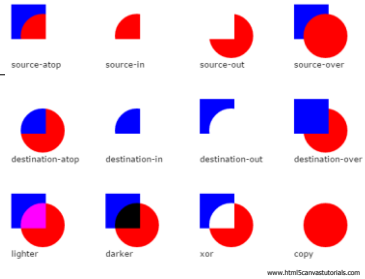


Przycinanie

```
ctx.save();

ctx.beginPath();
ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
ctx.clip();
//ctx.stroke();

ctx.beginPath();
ctx.arc(x - offset, y - offset, radius, 0, 2 * Math.PI, false);
ctx.fillStyle = 'yellow';
ctx.fill();

ctx.restore();
```

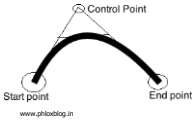
Operacje złożone

```
ctx.beginPath();
ctx.rect(0, 0, 50, 50);
ctx.fillStyle = 'blue';
ctx.fill();
ctx.globalCompositeOperation = "source-atop";
ctx.beginPath();
ctx.arc(50, 50, 35, 0,
        2 * Math.PI, false);
ctx.fillStyle = 'red';
ctx.fill();
```



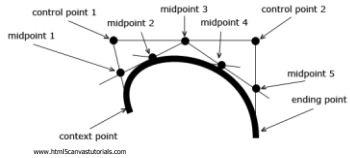
Parabola


```
ctx.beginPath();
ctx.moveTo(canvas.width/2-50, 350);          //start point
ctx.quadraticCurveTo(canvas.width/2, 400,    //control point
        canvas.width/2+50, 350);             //end point
ctx.stroke();
```



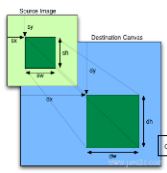
Krzywa Beziera


```
ctx.beginPath();
ctx.moveTo(188, 130);  //context point
      ctx.bezierCurveTo(140, 50, //control point 1
                        388, 10, //control point 2
                        388, 170);  //ending point
      ctx.lineWidth = 5;
      // line color
      ctx.strokeStyle = 'black';
      ctx.stroke();
ctx.stroke();
```



Wczytanie grafiki

```
drawbackground(drawforeground);

function drawbackground(onload){
        var imagePaper = new Image();
        imagePaper.onload = function(){
            ctx.drawImage(imagePaper, 0, 0); //x, y - z przesunięciem
                // x, y, w, h - ze skalowaniem
            onload();
        };
        imagePaper.src = "sea.jpg";
}

function drawforeground(){
--
}
```




```
drawImage(imagePaper, sx, sy, sw, sh, dx, dy, dw, dh)
```

Przekształcanie grafiki

```
<script>
function drawImage(imageObj) {
  var x = 69; var y = 50;

  ctx.drawImage(imageObj, x, y);

  var imageData = ctx.getImageData(x, y, imageObj.width, imageObj.height);
  var data = imageData.data;

  for(var i = 0; i < data.length; i += 4) {
     data[i] = 255 - data[i]; // red
     data[i + 1] = 255 - data[i + 1]; // green
     data[i + 2] = 255 - data[i + 2]; // blue
  }

  // overwrite original image
  ctx.putImageData(imageData, x, y);
}

var imageObj = new Image();
imageObj.onload = function() {
  drawImage(this);
};
imageObj.src = 'sea.jpg';
</script>
```

Animacja

```
function drawBall(ball, ctx) {
  ctx.arc(ball.x, ball.y, ball.r, 0,2*Math.PI); //x, y, r, start,
  ctx.fill();
}
var linearSpeed = -50; // pixels / second
function animate(ball, canvas, ctx, prevTime) {
        var time = (new Date()).getTime();
        var dY = linearSpeed * (time-prevTime)/ 1000;
        if(ball.y+dY >  ball.r && ball.y+dY<canvas.height-ball.r) {
          ball.y += dY;
        }
        else
           linearSpeed*=-1;
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        drawBall(ball, ctx);
        drawforeground();
        requestAnimationFrame(function() {
           animate(ball, canvas, ctx, time);
        });
      }
var ball = { x: canvas.width/2, y: 350, r: 15};

var startTime = (new Date()).getTime();
animate(ball, canvas, ctx, startTime);
```

Obsługa zdarzeń myszy

- Zdarzenie generowane przez mysz:
  - click
  - mousedown
  - mousemove,
  - mouseup,
  - mouseover,
  - mouseout

```
canvas.onmousedown = function (e) {
// reakcja na przyciśnięcie klawisza myszy
// e.clientX, e.clientY
};
```

```
canvas.addEventListener('mousedown', function (e) {
// reakcja na przyciśnięcie klawisza myszy
// e.clientX, e.clientY
});
```

Przeliczenie współrzędnych myszy na współrzędne kanwy

- Kanwa ma dwa rodzaje wymiarów:
  - obszar do rysowania, ustawiany za pomocą atrybutów: `height i width`
  - rozmiar elementu, ustawiany za pomocą CSS
  - jeżeli się różnią, to przeglądarka skaluje kanwę

```
<!DOCTYPE html>
<head>
  <title>Rozmiar elementu: 600 x 300, rozmiar kanwy: 300 x 150</title>
  <style>
    #canvas {width: 600px; height: 300px;}
  </style>
</head>
<body> <canvas id='canvas'></canvas> </body>
</html>
```

```
function windowToCanvas(canvas, x, y) {
  var bbox = canvas.getBoundingClientRect();
  return { x: x - bbox.left * (canvas.width / bbox.width),
           y: y - bbox.top * (canvas.height / bbox.height)
  };
}
```

Obsługa klawiatury

- Zdarzenie generowane przez klawiaturę:
  - keydown
  - keypress - tylko dla drukowalnych znaków
  - keyup
- Własności obiektu zdarzenia:
  - keyCode (unsigned long)
  - altKey (boolean)
  - ctrlKey (boolean)
  - metaKey (boolean)
  - shiftKey (boolean)
  - which (unsigned long) – można bezpiecznie użyć: `String.fromCharCode(event.which)`

Obsługa dotknięć

- Zdarzenie generowane przez interfejs dotykowy:
  - touchstart
  - touchmove
  - touchend
  - touchcancel
- Własności obiektu zdarzenia:
  - touches (TouchList) – bieżące dotknięcia
  - changedTouches (TouchList) – dotknięcia zmienione od poprzedniego zgłoszenia zdarzenia
  - targetTouches (TouchList) – dotknięcia, które pozostały w elemencie, w którym się rozpoczęły
  - altKey, ctrlKey, metaKey, shiftKey (boolean)