JESUÏTES
educació

DAWM12
GP1 FRAMEWORKS AVANÇATS EN JS

GP1 6.1 HTML5 CANVAS 2D

CFGS Desenvolupament d'Aplicacions Web
M12. Projecte Final - Global Project
Fundació Jesuïtes Educació - Escola del Clot
Sergi Grau sergi.grau@fje.edu

OBJECTIUS

- Aprendre l'API de canvas 2D
- Aplicar tècniques de manipulació de bits a un element canvas
- Gestionar esdeveniments sobre el canvas

L'ELEMENT CANVAS

- Canvas és una API d'HTML5 que permet dibuixar gràfics de mapa de bits, utilitzant seqüències d'ordres en JavaScript. Pot servir, per exemple per dibuixar gràfics, fer composicions de fotos o per animacions.

- El concepte original va ser creat per Apple per al Mac OS X Dashboard i pel navegador Safari. Posteriorment Apple va donar les seves patents al W3C.

- L'element canvas forma part de les aplicacions web 1.0 de les especificacions WHATWG també conegut com a HTML5.

- L'ús de <canvas> requereix un coneixement d'HTML i JavaScript.

VERIFICANT EL NAVEGADOR

- Com que HTML5 és encara relativament nou és necessari un mitjà per proporcionar contingut quan el navegador no és compatible amb l'element.

- Això és molt senzill: simplement cal oferir contingut alternatiu dins de l'element. Els navegadors que no suporten canvas ignoren el contenidor i mostren el contingut alternatiu en el seu interior. Els navegadors que són compatibles amb canvas ignoren el contingut dins del contenidor, i funcionen normalment.

- Al Safari, canvas és un element sense etiqueta de tancament com <img>. Però en el Mozilla Firefox requereix una etiqueta de tancament (</ canvas>).

VERIFICANT EL NAVEGADOR

- Si el contingut alternatiu no és necessàri, una simple <canvas id="foo" ...> </ canvas> serà plenament compatible amb Safari i Mozilla – El Safari simplement s'ignorarà l'etiqueta de tancament.

BIBLIOTECA PER IE<9

- per a utilitzar-ho amb versions IE<9 es pot utilitzar alguna biblioteca JavaScript existent, per exemple explorercanvas

- http://code.google.com/p/explorercanvas

- http://msdn.microsoft.com/en-us/library/ms537512(v=vs.85).aspx

VERIFICANT EL NAVEGADOR

```
<head>
<!--[if IE lt 9]><script src="excanvas.js"></script><![endif]--
>
</head>
```

```
try {
  document.createElement("canvas").getContext("2d");
  document.getElementById("suport").innerHTML =
    "HTML5 Canvas suportat";
} catch (e) {
document.getElementById("suport").innerHTML = "HTML5 Canvas no
suportat";
}
```

L'ELEMENT CANVAS

- <canvas id="tutorial" width="150" height="150"></canvas>

- Aquestes etiquetes s'assemblen molt a l'element <img>, l'única diferència és que no té els atributs src i alt. L'element <canvas> només té dos atributs - amplada i alçada. Totes dos són opcionals i també es poden establir mitjançant les propietats DOM. Quan no s'especifiquen els atributs d'amplada i alçada, el llenç serà inicialment de 300 píxels d'amplada i 150 píxels d'alt.

- Canvas suporta la majoria de les operacions 2D dels actuals sistemes operatius en relació als gràfics.

- Quan no s'apliquen les regles d'estil al llenç serà plenament transparent.

L'ELEMENT CANVAS

- Actualment canvas només ofereix context de dibuix 2D. WebGL permet 3D amb OpenGL ES.

- El <canvas> està inicialment en blanc, i per mostrar alguna cosa un primer script ha d'accedir al context de representació i dibuixar sobre aquest, i finalment aplicar els canvis sobre el context.

- <canvas> actualitza el navegador! </canvas>

L'ELEMENT CANVAS

- L'element canvas té un mètode anomenat getContext, utilitzat per obtenir el context de representació i les seves funcions de dibuix. getContext () pren un paràmetre, el tipus de context.

- La coordenada inicial (0,0) es troba a la part superior esquerra.

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

L'ELEMENT CANVAS

```
<html>
  <head>
    <title>canvas</title>
    <script type="text/javascript">
      function draw(){
        var canvas = document.getElementById('tutorial');
        if (canvas.getContext){
          var ctx = canvas.getContext('2d');
        }
      }
    </script>
    <style type="text/css">
      canvas { border: 1px solid black; }
    </style>
  </head>
  <body onload="draw();">
    <canvas id="tutorial" width="150" height="150"></canvas>
  </body>
</html>
```

L'ELEMENT CANVAS


```
<script>
  function dibuixarDiagonal() {

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

    context.beginPath();
    context.moveTo(70, 140);
    context.lineTo(140, 70);

    context.stroke(); // per a finalitzar i dibuixar
  }
   window.addEventListener("load", dibuixarDiagonal, true);
</script>
```

- La Graella: Normalment, 1 unitat a la graella correspon a 1 píxel del llenç. L'origen d'aquesta xarxa es col·loca en la cantonada superior esquerra (coordenades (0,0)). Tots els elements es posen en relació amb aquest origen. Més endavant veurem com podem convertir l'origen en una posició diferent, girar i reduir l'escala.

FORMES

- A diferència de SVG, l'element <canvas> només admet una forma primitiva.
- rectangles. Es dibuixen directament en el llenç, a diferència dels camins
- fillRect(x,y,width,height) : Dibuixa un rectangle ple
- strokeRect(x,y,width,height) : dibuixa la vora d'un rectangle
- clearRect(x,y,width,height) : neteja l'àrea especificada i la fa transparent.

- Camins: Representen qualsevol forma que es vulgui dibuixar. El primer pas per crear un camí és cridar al mètode beginPath. Cada vegada que aquest mètode és cridat, la llista es restableix i podem començar a dibuixar noves formes. El camí sempre utilitza el concepte de localització actual, per defecte el 0,0.

- El segon pas és cridar els mètodes que realment especifiquen els camins de dibuix, com ara moveTo(x,y)(mou la localització actual sense dibuixar) o lineTo(x,y)(mou dibuixant)

+ La tercera, i un pas opcional, seria cridar al mètode closepath. Aquest mètode intenta tancar la manera de dibuixar una línia recta des del punt actual de la sortida. Si la forma ja s'ha tancat o que només hi ha un punt en la llista, aquesta funció no fa res.

- L'últim pas serà trucar el mètode que dibuixa la vora o omple la figura. Es quan es dibuixa la forma al llenç. Quan s'utilitza el mètode d'omplir les formes obertes es tancaran automàticament i no cal utilitzar el mètode de closepath.

- Si utilitzem simultàniament stroke i fill, cal omplir abans que dibuixar el contorn de la forma per a evitar el fet que l'ompliment es fa centrat sobre la línia de contorn.

- beginPath(): inicia una forma i closePath(): tanca una forma amb l'origen, i informa que és una forma tancada

- stroke(): dibuixa contorn

- fill(): ompliment de la forma

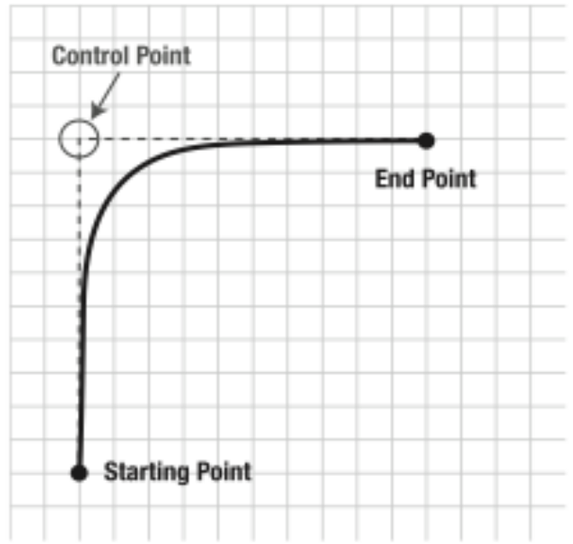
LÍNIES, RECTÀNGLES I ARCS

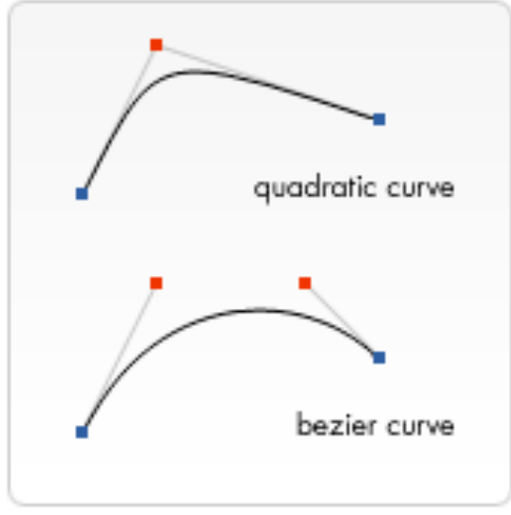

```
Línies: Depenen de com es tanca el camí, s'omplen les formes o
no lineTo(x, y)
ctx.beginPath();
ctx.moveTo(25,25);
ctx.lineTo(105,25);ctx.lineTo(25,105);
ctx.fill(); //dibuixa la forma plena

ctx.beginPath();
ctx.moveTo(125,125);
ctx.lineTo(125,45);ctx.lineTo(45,125);
ctx.closePath();
ctx.stroke(); //contorn

Rectangles: rect(x, y, width, height)
Arcs, mesurats en radians no en graus.
var radians = (Math.PI/180)*graus.
arc(x, y, radius, startAngle, endAngle, anticlockwise)
```

CORBES DE BEZIER I QUADRÀTIQUES

JESUÏTES educació
DAWM12 GP1 FRAMEWORKS AVANÇATS EN JS sergi.grau@fje.edu

```
quadraticCurveTo(cp1x, cp1y, x, y)
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)

diferències
ctx.beginPath();
ctx.moveTo(75,25);
ctx.quadraticCurveTo(25,25,25,62.5);
ctx.quadraticCurveTo(25,100,50,100);
ctx.quadraticCurveTo(50,120,30,125);
ctx.quadraticCurveTo(60,120,65,100);
ctx.quadraticCurveTo(125,100,125,62.5);
ctx.quadraticCurveTo(125,25,75,25);
ctx.stroke();
```





UTILITZANT IMATGES

- drawImage(`image, x, y`)
- drawImage(`image, x, y, width, height`)

- En primer lloc, necessitem una referència a un objecte o element d'imatge en JavaScript.
- En segon lloc dibuixem la imatge al llenç mitjançant la funció drawImage.

UTILITZANT IMATGES

- Vegem el pas primer. Hi ha bàsicament quatre opcions disponibles:

- 1. L'ús d'imatges que estan en la mateixa pàgina. Podem accedir a totes les imatges en una pàgina fent servir la col.lecció document.images, el mètode document.getElementsByTagName, o si sabem que l'atribut ID de la imatge, el mètode document.getElementById.

- 2. Ús d'altres elements <canvas> Igual que amb les imatges normals utilitzant el mètode document.getElementsByTagName o el mètode document.getElementById. Assegureu-vos que s'ha dibuixat alguna cosa en el llenç de la font abans d'utilitzar-la en el llenç de destinació.

UTILITZANT IMATGES

- 3. Creació d'una imatge des del principi: `var img = new Image(); img.onload = function(){ ... } img.src = 'myImage.png';`

- 4. URL: var img_src = '';

UTILITZANT IMATGES

- Cal tenir cura amb la utilització d'imatges doncs es carreguen assíncronament, i pot ser que encara no hi siguin disponibles.

```
var imatge= new Image();
imatge.src = "imatge.jpg";
//esperem a la càrrega de la imatge
imatge.onload = function () {
  dibuixar();
}
```

ESTILS I COLORS

- Color és una cadena que permet especificar els colors en CSS3
- fillStyle = color
- strokeStyle = color

- globalAlpha = valor de transparència [0,1]
- lineWidth = valor
- lineCap = [butt, round o square], lineJoin = [round, bevel o miter] i miterLimit = valor

```
//augment la línia de gruix
context.lineWidth = 4;
// arrodonint les cantonades de la forma
context.lineJoin = 'round';
// canviant el color de la forma
context.strokeStyle = '#663300';

ctx.strokeStyle = "orange";
ctx.strokeStyle = "#FFA500";
ctx.fillStyle = "rgb(255,165,0)";
ctx.fillStyle = "rgba(255,165,0,1)";
```

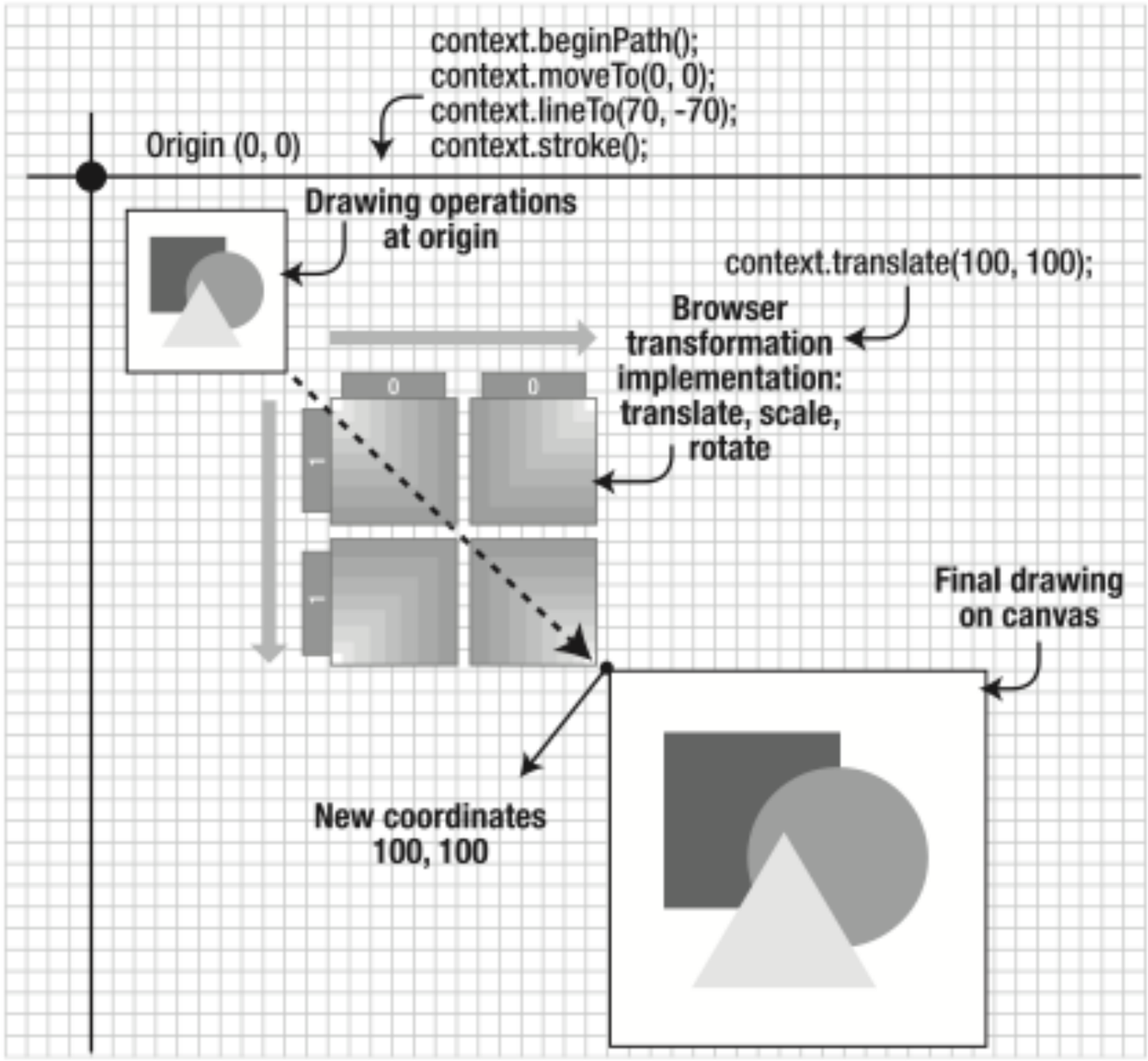
GRADIENTS, PATRONS I OMBRES

DAWM12 GP1 FRAMEWORKS AVANÇATS EN JS sergi.grau@fje.edu

```
createLinearGradient(x1,y1,x2,y2)
createRadialGradient(x1,y1,r1,x2,y2,r2)
Els colors s'afegeixen després
addColorStop(posició, color), de zero a 1
var lineargradient = ctx.createLinearGradient(0,0,150,150);
lineargradient.addColorStop(0,'white');
lineargradient.addColorStop(1,'black');
Ctx.fillStyle=lineargradient;
createPattern(image,type), els tipus són els de CSS (repeat,
repeat-x, ...)
shadowOffsetX = float
shadowOffsetY = float
shadowBlur = float
shadowColor = color
```

TRANSFORMACIONS

- Qualsevol transformació que s'apliqui afecta a tot el canvas.
- Utilitzem operacions lògiques de dibuix per a compensar el fet que tot es dibuixa en forma de píxels en el canvas.
- Les modificacions s'apliquen de manera seqüencial



TRANSFORMACIONS

- Passos:
- 1. Save(), desa el context actual a una pila, mantenint tot allò que està dibuixat de manera segura
- 2. Realitzem les transformacions:
- Traslació de la graella: translate(x, y)
- Rotació de la graella: rotate(angle)
- Escalat de la graella: scale(x, y)
- Matriu de transformació: transform(m11, m12, m21, m22, dx, dy)
- setTransform(m11, m12, m21, m22, dx, dy)

- 3. Restore(), Quan hem acabat recuperem el context de la pila

- <canvas> també permet composició (organitzar en capes els elements) i “clipping”

TRANSFORMACIONS


```
<!DOCTYPE html>
<html>
 <head>
   <!-- dibuixa un arbre -->
   <meta charset="utf-8" />
   <title>Exercicis HTML5</title>
   <script type="text/javascript">
     function dibuixaRectangle(context) {
   context.beginPath();
   context.moveTo(0, 0);
   context.lineTo(100, 0);
   context.lineTo(100, 100);
   context.lineTo(0, 100);
   context.closePath();}

function dibuixar() {
   var canvas = document.getElementById('exercici');
   if(canvas.getContext) {
     var context = canvas.getContext('2d');
                 dibuixaRectangle(context);
```

```
        context.fill();
        //canviem l'origen
        context.save();
        context.translate(100, 100);
        context.rotate(Math.PI / 4); // gira tot el canvas
        dibuixaRectangle(context);

        context.stroke();
          context.restore();
        }
      }
    </script>

  </head>
  <body onload="dibuixar();">
    <canvas id="exercici" width="300" height="300">
      Actualitza el teu navegador!
    </canvas>
  </body>
</html>
```

TEXT

- El text segueix el mateixos principis dels camins
- fillText (text, x, y, maxwidth)
- strokeText (text, x, y, maxwidth)
- Maxwidth permet ajustar la font a la mida del text
- font, permet especificar la font de la mateixa manera que els CSS
- textAlign, permet alinear el text
- textBaseLine, per a definir la línia de base del text.

```
context.save();
context.font = "60px impact";
context.fillStyle = '#996600';
context.textAlign = 'center';
context.fillText('Canvas!', 200, 60, 400);
context.restore();
```

TREBALLANT A NIVELL DE PÍXEL

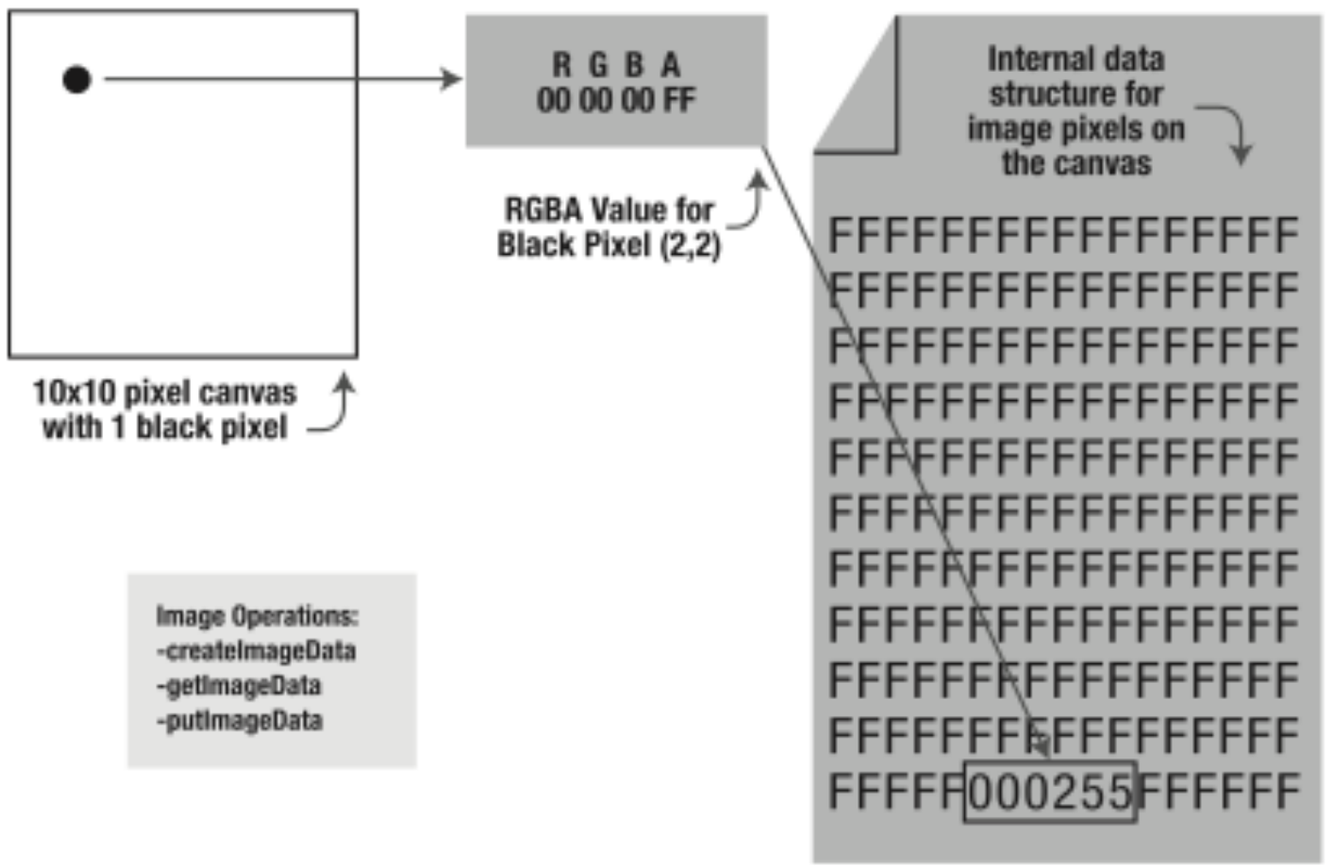
- Canvas pot treballar a nivell de píxel amb el mètode context.getImageData(sx,sy,sw,sh) que retorna una representació del canvas en forma de enters. Retorna width, el nombre de pixels en cada fila del “pixel data”, height, el nombre de pixels en cada columna del “pixel data”, i finalment data, una matriu RGBA amb els valors de cadascun dels píxels.

- Per a modificar aquesta matriu utilitzem putImageData(imageData, dx, dy).

- Si volem crear una matriu de dades en blanc podem utilitzar createImageData(sw, sh)

- Per motius de seguretat només es poden modificar a nivell de píxel aquelles imatges generades pel propi canvas

TREBALLANT A NIVELL DE PÍXEL



Red component: ((width * y) + x) * 4

Green component: ((width * y) + x) * 4 + 1

Blue component: ((width * y) + x) * 4 + 2

Alpha component: ((width * y) + x) * 4 + 3

EXEMPLE DE MANIPULACIÓ A NIVELL DE BITS

EXEMPLE DE MANIPULACIÓ A NIVELL DE BITS


```
<head>
   <meta charset="utf-8" />
   <title>exercici7</title>
   <meta name="author" content="sergi grau" />
   <link href='http://fonts.googleapis.com/css?family=Telex'
rel='stylesheet' type='text/css'>
   <link rel="stylesheet" href="estils.css">
   <script type="text/javascript" src="Exercici7.js"></script>
 </head>
 <body>
 <canvas id="exercici" width="300" height="300">
     Actualitza el teu navegador!
   </canvas>
 </body>
</html>
```

EXEMPLE DE MANIPULACIÓ A NIVELL DE BITS


```
var imatge = new Image();
imatge.src = "images/logo.png";
imatge.onload = function() {
 dibuixar();
};

try {
 document.createElement("canvas").getContext("2d");
 //window.addEventListener("load", dibuixar, true);
} catch (e) {
 alert("HTML5 Canvas no suportat.");
}

function dibuixar() {
 var canvas = document.getElementById('exercici');
 if (canvas.getContext) {
   var ctx = canvas.getContext('2d');
   ctx.drawImage(imatge, 0, 0);
   ferNegatiu(imatge, ctx, canvas);
```

EXEMPLE DE MANIPULACIÓ A NIVELL DE BITS

```
 }
}
 function ferNegatiu(imageObj, context, canvas){
    var destX = 0;
    var destY = 100;

    context.drawImage(imageObj, destX, destY);
    var imageData = context.getImageData(0, 100, canvas.width,
canvas.height);
    var pixels = imageData.data;
    for (var i = 0; i < pixels.length; i += 4) {
        pixels[i]   = 255 - pixels[i];   // red
        pixels[i+1] = 255 - pixels[i+1]; // green
        pixels[i+2] = 255 - pixels[i+2]; // blue
        // i+3 es alpha
    }
    // modifiquem original
    context.putImageData(imageData, 0, 100);
}
```

```
<head>
    <meta charset="utf-8" />
     <script type="text/javascript">
         window.addEventListener("load", inici, true);

        function obtenirCoordenades(canvas, evt, ctx) {
            var rect = canvas.getBoundingClientRect();
            var x = evt.clientX - rect.left;
            var y = evt.clientY - rect.top;
            console.log("x: " + x + " y: " + y);
            ctx.lineTo(x+1, y+1);
            ctx.stroke();
        }
```

```
        function inici() {
            if (canvas.getContext) {
                var ctx = canvas.getContext('2d');
                ctx.beginPath();
                canvas.addEventListener('mousemove', function
(evt) {

obtenirCoordenades(document.getElementById('canvas'), evt,
ctx);
                }, false);
            }
        }
    </script>
</head>
<body>
    <canvas id="canvas">
    </canvas>
</body>
</html>
```

BIBLIOGRAFIA

+ https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
+ http://www.w3schools.com/tags/ref_canvas.asp
+ http://www.w3.org/TR/2dcontext/
+ http://www.whatwg.org/specs/web-apps/current-work/#the-canvas-element
+ http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(HTML5)