CFGS DAW

Mòdul 6: Desenvolupament web en entorn client

INS Joan d'Àustria

Canvas avanzado

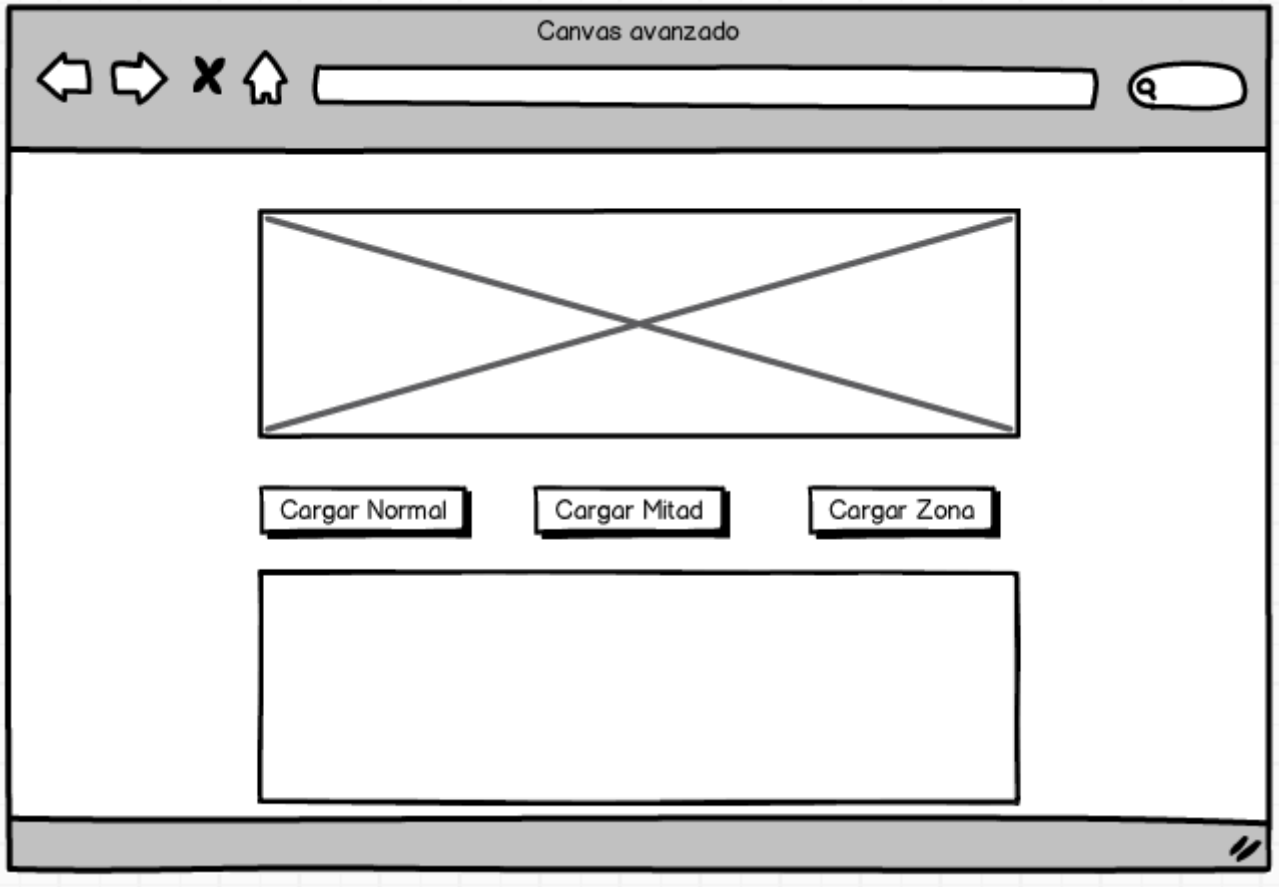
Imágenes dentro del canvas

Para poner imágenes dentro del canvas utilizamos la función drawImage con 3, 5 o 9 argumentos:

- context.drawImage(image, x, y): indicamos las coordenadas x,y donde se situará la imagen
- context.drawImage(image, x, y, ancho, alto): indicamos las coordenadas y también su anchura y altura
- context.drawImage(image, sx, sy, sw, sh, x, y, ancho, alto): Indicamos, además, la parte de la imagen que queremos coger mediante las coordenadas sx, sy y sw, sh (source x,y width, height)

Canvas avanzado

Ejemplo



Canvas avanzado

Para trabajar con la información utilizaremos el objeto ImageData que nos ofrece los siguientes métodos y propiedades:

Propiedades	Métodos
data	getImageData
width	putImageData
height	createImageData

Canvas avanzado

Obtener la información:

```
ImageData = context.getImageData(x,y,ancho,alto);
```

Una vez disponemos de esta información podemos tratarla como si fuera un array. Cada píxel de la imagen se almacenará en 4 posiciones del array, correspondiendo a los valores r,g,b,a

```
var ImageData = context.getImageData(0,0,1,1);
for (var i=0; i<ImageData.data.length; i+=4) {
        var r = ImageData.data[i];
        var g = ImageData.data[i+1];
        var b = ImageData.data[i+2];
        var a = ImageData.data[i+3];
        alert(r+" "+g+" "+b+" "+a);
}
```

Canvas avanzado

Este array que hemos obtenido se puede manipular, de forma que podemos sobreescribir información de la misma manera.

Para colocar la nueva información utilizaremos el método putImageData

```
for (var i=0; i<ImageData.data.length; i+=4) {
ImageData.data[i] = parseInt(Math.random()*255);
ImageData.data[i+1] = parseInt(Math.random()*255);
ImageData.data[i+2] = parseInt(Math.random()*255);
}
context.putImageData(ImageData, x, y);
```

Canvas avanzado

También se puede crear un nuevo array vacio en lugar de obtener una porción de la imagen mediante el método createImageData. Este método recibirá como parámetros la anchura y la altura. Todo el array estará inicializado con el valor negro transparente rgba(0,0,0,0).

```
var imagedata = context.createImageData(ancho, alto);
for (var i=0; i<ImageData.data.length; i+=4) {
        imagedata.data[i] = parseInt(Math.random()*255);
        imagedata.data[i+1] = parseInt(Math.random()*255);
        imagedata.data[i+2] = parseInt(Math.random()*255);
        imagedata.data[i+3] = 255;
}
context.putImageData(imagedata,0,0);
```

Canvas avanzado

Con ayuda de algunas funciones matemáticas podemos construir nuestros propios filtros de tratamiento de imágenes. Utilizaremos el siguiente patrón tipo:

```
var modified = context.createImageData(ancho,alto);
var imagedata = context.getImageData(0,0,ancho,alto);
for (var i=0; i<imagedata.data.length; i+=4) {
        var rgba = filtro (imagedata.data[i+0],imagedata.data[i+1],
                imagedata.data[i+2], imagedata.data[i+3] );
        modified.data[i+0] = rgba[0];
        modified.data[i+1] = rgba[1];
        modified.data[i+2] = rgba[2];
        modified.data[i+3] = rgba[3];
}
context.putImageData(modified,0,0);
```

Canvas avanzado

De esta manera podemos aplicar los filtros típicos de un programa de retoque fotográfico como por ejemplo:

- Escala de grises
- Tono sepia
- Invertir
- Intercambiar canales
- Color monocromático

Canvas avanzado

Escala de grises:

El código del programa Gimp, nos ofrece tres alternativas de escala de grises (http://docs.gimp.org/es/gimp-tool-desaturate.html), así como las fórmulas para obtenerlos

```
var claridad = function(r,g,b,a) {
        var val = parseInt((Math.max(r,g,b)+Math.min(r,g,b))*0.5);
        return [val,val,val,a];
};
```

```
var luminosidad = function(r,g,b,a) {
        var val = parseInt( (r*0.21)+(g*0.71)+(b*0.07) );
        return [val,val,val,a];
};
```

```
var media = function(r,g,b,a) {
        var val = parseInt( (r+g+b)/3.0 );
        return [val,val,val,a];
};
```

Canvas avanzado

Tono sepia:

Un artículo de Zach Smith (http://www.techrepublic.com/blog/howdoi/how-do-i-convert-images-to-grayscale-and-sepia-tone-using-c/120?tag=content;siu-container) nos ofrece la manera:

```
var sepiaTone = function(r,g,b,a) {
        var rS = (r*0.393)+(g*0.769)+(b*0.189);
        var gS = (r*0.349)+(g*0.686)+(b*0.168);
        var bS = (r*0.272)+(g*0.534)+(b*0.131);
        return [ (rS>255) ? 255 : parseInt(rS),
                 (gS>255) ? 255 : parseInt(gS),
                 (bS>255) ? 255 : parseInt(bS),
                 a ];
};
```

Canvas avanzado

Invertir colores:

Para invertir colores simplemente tenemos que restar al máximo (255) la cantidad de cada color. Si estaba al máximo obtendremos el mínimo, si estaba al mínimo el máximo, etc.

```
var invertColor = function(r,g,b,a) {
        return [ (255-r), (255-g), (255-b), a ];
};
```

Canvas avanzado

Intercanviar canales:

El proceso consiste en indicar qué orden queremos aplicar, de forma que podemos pasar por ejemplo de RGBA a BRGA. Para ello deberíamos añadir un parámetro más a la función que indique este nuevo orden (por ejemplo order=[2, 0, 1, 3])

```
var swapChannels = function(r,g,b,a,order) {


};
```

Canvas avanzado

Color monocromático:

El proceso consiste en poner cada pixel de los componentes RGB en un color particular, usando el tono de gris del píxel inicial como componente para el canal alfa. Para esto, necesitamos un parámetro más que indique el color monocromático que queremos, por ejemplo, para monocromo azul, podríamos indicar color = [0,0,255]

```
var monoColor = function(r,g,b,a,color) {
        return [ color[0], color[1], color[2], 255-(parseInt((r+g+b)/3.0)) ];
};
```

Canvas avanzado

Más información:

En la página web:

http://www.pixastic.com/lib/

podemos encontrar muchos filtros javascript libres.