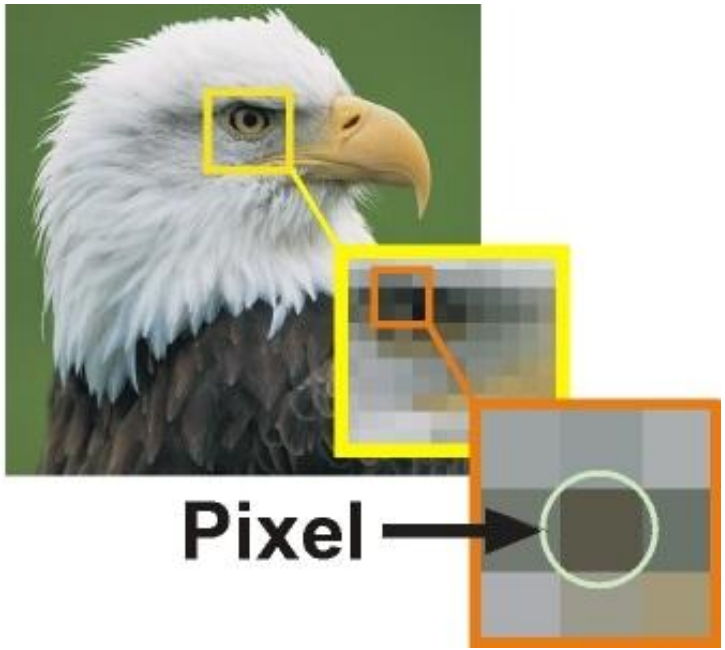
FILTROS

Imágenes

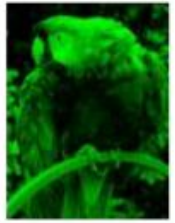
Una imagen digital es una matriz 2D (WxH) de píxeles.



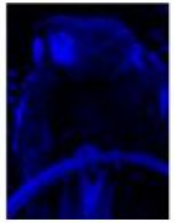
0	0	0	1	50	50	50	1	100	100	100	1

R

G



B



Representar imágenes

- HTML5 provee un “*ImageData*” .
- La imagen se recorre en *Ancho* y *Alto (for, for)*
- *“ImageData”* almacena los pixeles en un arreglo de 1d
- Los colores se almacenan en un *Array de enteros* y se pueden acceder como si fueran una matriz

- Para dibujar la Imagen en Pantalla: **ctx.putImageData (Imagen, x, y) ;**
- Para convertir de matriz a arreglo?

indice = …

indice = (x + y * imageData.width) * 4;

Crear imagen desde cero

```
var imageData = ctx.createImageData(width, height);

for (x=0; x<width; x++){
     for (y=0; y<height; y++){
         setPixel(imageData, x, y, r, g, b, a);
     }
}

ctx.putImageData(imageData, 0, 0);
```

```
function setPixel(imageData, x, y, r, g, b, a)
{
     index = (x + y * imageData.width) * 4;
     imageData.data[index+0] = r;
     imageData.data[index+1] = g;
     imageData.data[index+2] = b;
     imageData.data[index+3] = a;
}
```

Cómo cargar una imagen?

- La carga de imágenes es **asincrónica**
- El script se ejecuta secuencialmente línea por línea
- El tiempo de demora depende del tamaño de la imagen y de la latencia del servidor que la tiene

```
var image1 = new Image();

image1.src = "imagen.jpg";

ctx.drawImage(image1, 0, 0);
```

 Puede que la imagen no esté cargada en memoria al momento de dibujarla

Acceder a los pixeles

- ¿Cómo accedo a cada pixel?
- ¿Cómo obtengo las componentes R, G y B?

```
function getRed(imageData, x, y) {
    index = (x + y * imageData.width) * 4;
    return imageData.data[index+0];
}

function getGreen(imageData, x, y) {
    index = (x + y * imageData.width) * 4;
    return imageData.data[index+1];
}

function getBlue(imageData, x, y) {
    index = (x + y * imageData.width) * 4;
    return imageData.data[index+2];
}
```

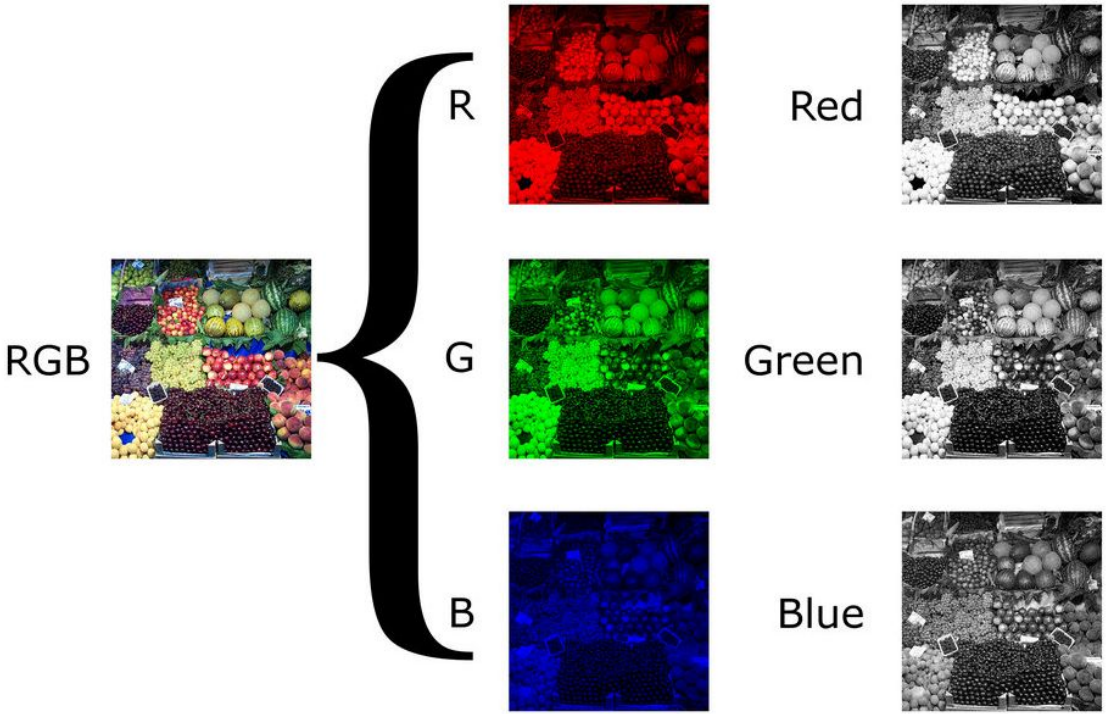
Filtros

- Modifica el valor de un pixel dado una ecuación matemática
- Este valor puede ser simplemente un coeficiente

$$Gris = (R + G + B) / 3$$

Escala de Grises

- R = G = B



BLUR

- Promedio de la Matriz vecina

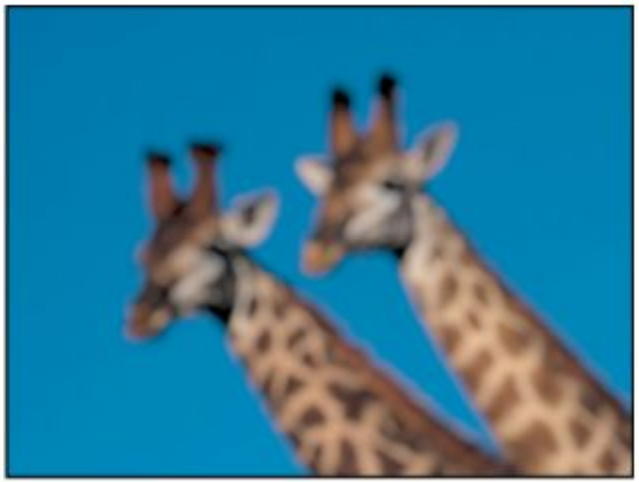
=(C5+D5+E5+E6+D6+C6+C7+D7+E7)/9

original

	0	1	2	3	4
0					
1					
2		20	33	36	70
3		20	50	66	90
4		55	55	55	99

copia

	0	1	2		
0					
1					
2					
3			43.333	61.5	69.3



Filtro de Sobel

- Se utiliza una Matriz (Gx) para multiplicar con la imagen original
- El resultado es 1 numero que será el valor del pixel original
- Utilizado principalmente para detección de bordes

https://es.wikipedia.org/wiki/Operador_Sobel

Filtro de Sobel

- Cálculo utilizando Gx
- Recuerden calcularlo para cada color
- Y con los extremos de la Matriz?

Original Image

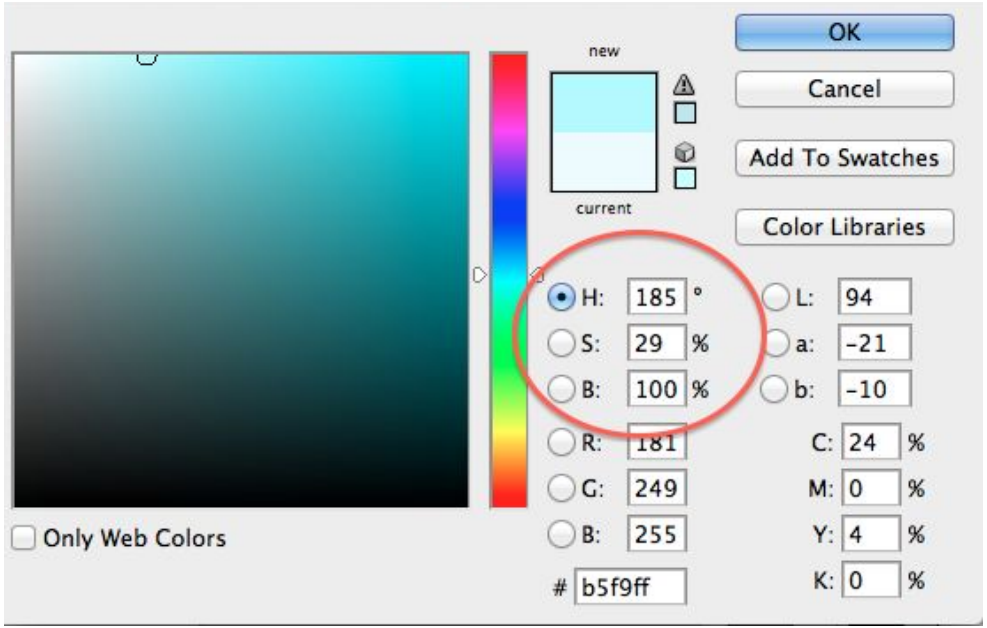
Gx

a1 x a2 = 50
b1 x b2 = 100
c1 x c2 = 50
d1 x d2 = 0
e1 x e2 = -100
f1 x f2 = -200
g1 x g2 = -100
h1 x h2 = 0
TOTAL = -200

Saturación y Brillo

- Si tenemos imagen en RGB. Entonces convertir el color a HSB, hacer el calculo y luego nuevamente pasarlo a RGB

    1. RGB to HSB
    2. Calcular nuevo valor
    3. HSB to RGB
    4. Guardar nuevo valor en la imagen



https://en.wikipedia.org/wiki/HSL_and_HSV