Teoria - Colores, Parte 1:

¿Qué es el color? El color es luz.

Colores Primarios de la computacion: Rojo – Verde – Azul. (RGB).

paletton.com //Paletas de colores que contrasten bien el uno con el otro.

Sistemas de representación:

(ALPHA == Transparencia)

RGB: RED-GREEN-BLUE (3 Bytes)

RGBA: RED-GREEN-BLUE-ALPHA (4 Bytes)

HSL: HUE-SATURATION-LIGTHNESS

HSLA: HUE-SATURATION-LIGTHNESS-ALPHA

HEXADECIMAL: #RRGGBBAA

En JavaScript:

let color = rgba(255,255,255,1);

let color2 = hsla(0,0,0,0);

Canvas?

<canvas id=”algo”>

Teoria - Colores, Parte 2:

Una imagen es una matriz en 2D de pixeles.

HTML5 provee un “ImageData”, este almacena los pixeles en un arreglo de 1d. La imagen se recorre en Ancho y Alto (for, for).

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);

Formula de conversión de matriz a arreglo:

Indice = (x + y \* ImageData.width) \* 4;

CREAR IMAGENES DESDE CERO:

let imageData = ctx.createImageData(width, height);

for (let x = 0; x < width; x++ ) {

for (let y = 0; y < he; 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;

}

La carga de la imagen es asincrónica , el script se ejecuta secuencialmente línea por línea. El tiempo de demora dependerá de la latencia del servidor y del tamaño de la imagen.

ASI ESTA MAL:

Let image1 = new Image();

Image1.src = “image.jpg”;

ctx.drawImage(image1, 0, 0);

ASI ESTA BIEN:

let image1 = new Image();

image1.src = “imagen.jpg”;

image1.onload = function() {

myDrawImageMethod(this);

}

Function myDrawImageMethod(Image) {

ctx.drawImage(image1, 0, 0);

}