



**Materia:** Tecnologías para la Web.  
**Tema:** Imágenes con **canvas**.

### PARTE III.

#### 1. Imágenes

- El método `drawImage` dibuja una imagen en un `canvas` mediante el uso de cinco argumentos. El primer argumento puede ser un elemento `image`, `canvas` o `video`. Los argumentos segundo y tercero son las coordenadas `x` y `y` del destino: éstas indican la posición de la esquina superior izquierda de la imagen en el `canvas`. Los argumentos cuarto y quinto son la anchura de destino y la altura de destino.

#### EJEMPLO 8.

##### Archivo `imagen.html`:

```
<!-- Dibuja una imagen en un canvas. -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Imágenes</title>
    <script>
      var imagen = new Image();
      imagen.src = "floresamarillas.png";
      function dibujar(){
        var canvas = document.getElementById("miimagen");
        var contexto = canvas.getContext("2d");
        contexto.drawImage(imagen, 0, 0, 175, 175);
      } // fin de la función dibujar
      window.addEventListener( "load", dibujar, false );
    </script>
  </head>
  <body>
    <canvas id = "miimagen" width = "200" height = "200"
      style = "border: 1px solid Black;">
    </canvas>
  </body>
</html>
```



#### 2. Manipulación de imágenes: procesamiento de los píxeles individuales de un `canvas`

- Es posible obtener los píxeles de un `canvas` y manipular sus valores rojo, verde, azul y alfa (RGBA).
- Podemos cambiar los valores RGBA con los elementos de entrada del tipo `range` definidos en el elemento `body`.
- El método `getImageData` obtiene un objeto que contiene los píxeles a manipular. El método recibe un rectángulo delimitador que representa una parte del `canvas` a obtener.
- El objeto devuelto contiene un arreglo llamado `data`, el cual almacena todos los píxeles en el área rectangular seleccionada como cuatro elementos en el arreglo. Los datos de cada píxel se almacenan en el orden rojo, verde, azul, alfa.
- Así, los primeros cuatro elementos en el arreglo representan los valores RGBA del píxel en la fila 0 y la columna 0, los siguientes cuatro elementos representan el píxel en la fila 0 y la columna 1, etcétera.

#### EJEMPLO 9.



Por cuestiones de seguridad, algunos navegadores permiten que una secuencia de comandos obtenga los píxeles de una imagen sólo si el documento se solicita desde un servidor Web y no si el archivo se carga desde el sistema de archivos de la computadora local. Por esta razón se puede probar este ejemplo en:

[http://test.deitel.com/iw3http5/ch14/fig14\\_12/imagenmanipulation.html](http://test.deitel.com/iw3http5/ch14/fig14_12/imagenmanipulation.html)

### Archivo manipulacionimagenes.html:

```
<!-- Manipulación de los píxeles de una imagen para cambiar colores y transparencia.-->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Manipulación de una imagen</title>
    <style>
      label { display: inline-block; width: 3em; }
      canvas { border: 1px solid black; }
      input[type="range"] { width: 600px; }
    </style>
    <script>
      var contexto;      // contexto para dibujar sobre el lienzo
      var rangoRojo;     // % del valor del pixel rojo original
      var rangoVerde;    // % del valor del pixel verde original
      var rangoAzul;     // % del valor del pixel azul original
      var rangoAlfa;     // valor de cantidad alfa
      var imagen = new Image(); // objeto imagen para almacenar la imagen cargada
      imagen.src = "floresrojas.png"; // Establecer el origen de la imagen
      function iniciar(){
        var canvas = document.getElementById("ellienzo");
        contexto = canvas.getContext("2d")
        contexto.drawImage (imagen, 0, 0); // imagen original
        contexto.drawImage(imagen, 250, 0); // imagen para las
modificaciones del usuario
original
        procesarEscalaGris(); // muestra la escala de grises de la imagen

        // configurar eventos de GUI
        rangoRojo = document.getElementById( "rangoRojo" );
        rangoRojo.addEventListener( "change",
          function() { procesarImagen( this.value, rangoVerde.value,
            rangoAzul.value ); }, false );
        rangoVerde = document.getElementById( "rangoVerde" );
        rangoVerde.addEventListener( "change",
          function() { procesarImagen( rangoRojo.value, this.value,
            rangoAzul.value ); }, false );
        rangoAzul = document.getElementById( "rangoAzul" );
        rangoAzul.addEventListener( "change",
          function() { procesarImagen( rangoRojo.value,
            rangoVerde.value, this.value ); }, false );
        rangoAlfa = document.getElementById( "rangoAlfa" );
        rangoAlfa.addEventListener( "change",
          function() { procesarAlfa( this.value ); }, false );
        document.getElementById( "botonReiniciar" ).addEventListener(
          "click", reiniciarImagen, false );
      } // fin de la función iniciar
      // establece el valor alfa para todos los pixeles
      function procesarAlfa( nuevoValor ){
        // obtiene el objeto ImageData que representa el contenido del
lienzo
        var datosImagen = contexto.getImageData(0, 0, 250, 250 );
```



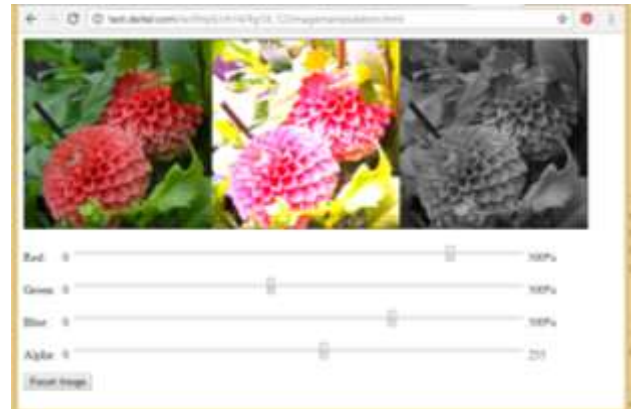
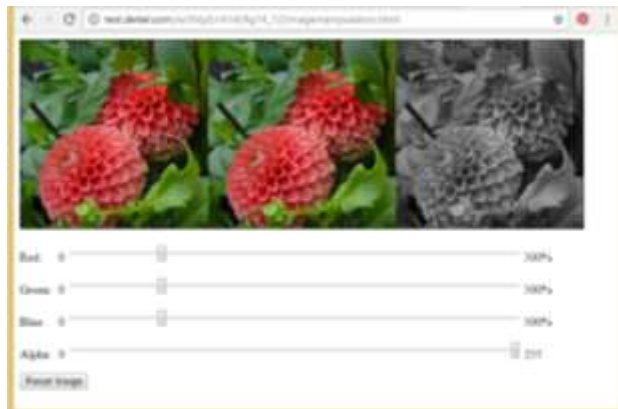
```
var pixeles = datosImagen.data; // información de pixeles de
ImageData
    // convierte cada pixel a escala de gris
    for ( var i = 3; i < pixeles.length; i += 4 ){
        pixeles [ i ] = nuevoValor;
    } // fin de for
    contexto.putImageData( datosImagen, 250, 0 ); // muestra escala de
gris
} // fin de la función procesarImagen
// establece los valores RGB para cada pixel
function procesarImagen( porcentajeRojo, porcentajeVerde, porcentajeAzul
){
    // obtener el objeto ImageData que representa el contenido del
lienzo
    contexto.drawImage(imagen, 250, 0);
    var datosImagen = contexto.getImageData(0, 0 , 250, 250);
    var pixeles = datosImagen.data; // información de pixeles de
ImageData
    // establecer porcentajes de rojo, verde y azul en cada pixel
    for ( var i= 0; i < pixeles.length; i += 4 ){
        pixeles[ i ] *= porcentajeRojo / 100;
        pixeles[ i + 1 ] *= porcentajeVerde / 100;
        pixeles[ i + 2 ] *= porcentajeAzul / 100;
    } // fin de for
    contexto.putImageData( datosImagen, 250, 0 ); // mostrar escala de
gris
} // fin de la función procesarImagen
// crea versión en escala de grises de la imagen original
function procesarEscalaGrises(){
    // obtiene el objeto ImageData que representa el contenido del
lienzo
    contexto.drawImage(imagen, 500, 0);
    var datosImagen = contexto.getImageData(0, 0 , 250, 250);
    var pixeles = datosImagen.data; // información de pixeles de
ImageData
    // convierte cada pixel en escala de gris
    for ( var i = 0; i < pixeles.length; i += 4 ){
        var promedio =
            (pixeles[ i ] * 0.30 + pixeles [ i + 1 ] * 0.59 +
            pixeles[ i + 2 ] * 0.11).toFixed(0);
        pixeles[ i ] = promedio;
        pixeles[ i + 1 ] = promedio;
        pixeles[ i + 2 ] = promedio;
    } // fin de for
    contexto.putImageData( datosImagen, 500, 0 ); // mostrar escala de
grises
} // fin de la función procesarEscalaGrises
// reinicia la imagen manipulada por el usuario y los controles deslizantes
function reiniciarImagen(){
    contexto.drawImage(imagen, 250, 0);
    rangoRojo.value = 100;
    rangoVerde.value = 100;
    rangoAzul.value = 100;
    rangoAlfa.value = 255;
} // fin de la función reiniciarImagen
window.addEventListener( "load", iniciar, false );
</script>
```



```

</head>
<body>
  <canvas id = "ellienzo" width = "750" height = "250" ></canvas>
  <p><label>Rojo:</label> 0 <input id = "rangoRojo"
    type = "range" max = "500" value = "100"> 500%</p>
  <p><label>Verde:</label> 0 <input id = "rangoVerde"
    type = "range" max = "500" value = "100"> 500%</p>
  <p><label>Azul:</label> 0 <input id = "rangoAzul"
    type = "range" max = "500" value = "100"> 500%</p>
  <p><label>Alfa:</label> 0 <input id = "rangoAlfa"
    type = "range" max = "255" value = "255"> 255</p>
  <p><input id = "botonReiniciar" type = "button"
    value = "Reiniciar imagen">
</body>
</html>

```



### 3. Patrones

- El método `createPattern` recibe dos argumentos. El primero es la imagen que usamos para el patrón, que puede ser un elemento `image`, un elemento `canvas` o un elemento `video`. El segundo especifica cómo se repetirá la imagen para crear el patrón y puede ser uno de cuatro valores: `repeat` (se repite en sentido horizontal y vertical), `repeat-x` (se repite en sentido horizontal), `repeat-y` (se repite en sentido vertical) o `no-repeat`.
- Use el valor `pattern` del atributo `fillStyle` y el método `fill` para dibujar el patrón en el `canvas`.

#### EJEMPLO 10.

##### Archivo `patrones.html`:

```

<!-- Creación de un patrón mediante el uso de una Imagen en un canvas. -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Patrones</title>
    <script>
      var imagen = new Image();
      imagen.src = "floresamarillas.png";
      function iniciar() {
        var canvas = document.getElementById("patron");
        var contexto = canvas.getContext("2d");
        var patron = contexto.createPattern(imagen, "repeat");
        contexto.rect(5, 5, 385, 200);
        contexto.fillStyle = patron;
        contexto.fill();
      } // fin de la función iniciar
    </script>
  </head>
  <body>
    <div>
      <img alt="Yellow flowers" data-bbox="115 325 460 415"/>
      <div>
        Red: 0 <input type="range" value="100" max="500"/> 500%
        Verde: 0 <input type="range" value="100" max="500"/> 500%
        Azul: 0 <input type="range" value="100" max="500"/> 500%
        Alfa: 0 <input type="range" value="255" max="255"/> 255
        <button id="botonReiniciar" value="Reiniciar imagen">
      </div>
    </div>
  </body>
</html>

```



```
        window.addEventListener( "load", iniciar, false );
    </script>
</head>
<body>
    <canvas id = "patron" width = "400" height = "200"
        style = "border: 1px solid black;">
    </canvas>
</body>
</html>
```



#### 4. Transformaciones

- Podemos cambiar la matriz de transformación (las coordenadas) en el `canvas` mediante el método `translate`, de modo que el centro del `canvas` se convierta en el punto de origen con los valores 0, 0 para `x`, `y`.
- El método `scale` puede estirar un círculo para crear una elipse. El valor `x` representa el factor de escala horizontal, el valor `y` el factor de escala vertical.
- El método `rotate` nos permite crear rotaciones animadas en un `canvas`.
- Para girar una imagen alrededor de su centro, cambie la matriz de transformación en el `canvas` usando el método `translate`.
- El método `rotate` recibe un argumento: el ángulo de la rotación en sentido de las manecillas del reloj, expresado en radianes.
- El método `setInterval` del objeto `window` recibe dos argumentos. El primero es el nombre de la función a llamar (girar) y el segundo es el número de milisegundos entre llamadas.
- El método `clearRect` borra los píxeles del rectángulo del `canvas` y los convierte de vuelta en transparentes. Este método recibe cuatro argumentos: `x`, `y`, `anchura` y `altura`.
- El método `transform` nos permite sesgar, escalar, girar y trasladar elementos sin necesidad de usar los métodos de transformación separados.
- El método `transform` recibe seis argumentos en el formato (a, b, c, d, e, f) con base en una matriz de transformación. El primer argumento (a) es la escala `x`: el factor por el cual se ajustará la escala del elemento en forma horizontal. El segundo argumento (b) es la inclinación `y`. El tercer argumento (c) es la inclinación `x`. El cuarto argumento (d) es la escala `y`: el factor por el cual se ajusta la escala del elemento en sentido vertical. El quinto argumento (e) es el traslado `x` y el sexto argumento (f) es el traslado `y`.

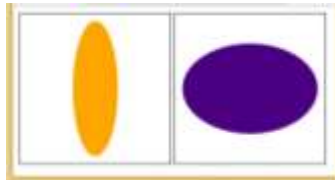
#### EJEMPLO 11.

##### Archivo `ellipse.html`:

```
<!-- Dibujar una elipse sobre un canvas. -->
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Ellipse</title>
    </head>
    <body>
        <!-- elipse vertical -->
        <canvas id = "dibujarElipse" width = "200" height = "200"
            style = "border: 1px solid black;">
        </canvas>
        <script>
```



```
var canvas = document.getElementById("dibujarElipse");
var contexto = canvas.getContext("2d");
contexto.translate(canvas.width / 2, canvas.height / 2);
contexto.scale(1, 3);
contexto.beginPath();
contexto.arc( 0, 0, 30, 0, 2 * Math.PI, true);
contexto.fillStyle = "orange";
contexto.fill();
</script>
<!-- elipse horizontal -->
<canvas id = "dibujarElipse2" width = "200" height = "200"
  style = "border: 1px solid black;">
</canvas>
<script>
  var canvas = document.getElementById("dibujarElipse2");
  var contexto = canvas.getContext("2d");
  contexto.translate(canvas.width / 2, canvas.height / 2);
  contexto.scale( 3, 2);
  contexto.beginPath();
  contexto.arc( 0, 0, 30, 0, 2 * Math.PI, true );
  contexto.fillStyle = "indigo";
  contexto.fill();
</script>
</body>
</html>
```



**Naranja:** escala entre ancho y alto es 1,3.      **Índigo:** escala entre ancho y alto es 3,2.

## EJEMPLO 12.

### Archivo girar.html:

```
<!-- Uso del método rotate para girar un rectángulo en un canvas. -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Girar</title>
  </head>
  <body>
    <canvas id = "girarRectangulo" width = "200" height = "200"
      style = "border: 1px solid black;">
    </canvas>
    <script>
      var canvas = document.getElementById("girarRectangulo");
      var contexto = canvas.getContext("2d");
      function iniciarRotacion(){
        contexto.translate(canvas.width/ 2, canvas.height / 2);
        setInterval(girar, 10);
      }
      function girar(){
        contexto.clearRect(-100, -100, 200, 200);
        contexto.rotate(Math.PI / 360);
        contexto.fillStyle = "lime";
```



```
        contexto.fillRect( -50, -50, 100, 100);
    }
    window.addEventListener( "load", iniciarRotacion, false );
</script>
</body>
</html>
```



### EJEMPLO 13.

#### Archivo **inclinarse.html**:

```
<html>
  <head>
    <meta charset = "utf-8">
    <title>Inclinarse</title>
  </head>
  <body>
    <!-- inclinar a la izquierda -->
    <canvas id = "transform" width = "320" height = "150"
      style = "border: 1px solid Black;">
    </canvas>
    <script>
      var canvas = document.getElementById("transform");
      var contexto = canvas.getContext("2d");
      var anchuraRectangulo = 120;
      var alturaRectangulo = 60;
      var escalaX = 2;
      var inclinacionY = 0;
      var inclinacionX = 1;
      var escalaY = 1;
      var trasladoX = -10;
      var trasladoY = 30;
      contexto.translate(canvas.width / 2, canvas.height / 2);
      contexto.transform(escalaX, inclinacionY, inclinacionX, escalaY,
        trasladoX , trasladoY);
      contexto.fillStyle = "red";
      contexto.fillRect(-anchuraRectangulo / 2, -alturaRectangulo / 2,
        anchuraRectangulo, alturaRectangulo);
    </script>
    <!-- inclinar a la derecha -->
    <canvas id = "transform2" width = "220" height = "150"
      style = "border: 1px solid Black;">
    <script>
      var canvas = document.getElementById("transform2");
      var contexto = canvas.getContext("2d");
      var anchuraRectangulo = 120;
      var alturaRectangulo = 60;
      var escalaX = 1;
      var inclinacionY = 0;
      var inclinacionX = -1.5;
      var escalaY = 2;
      var trasladoX = 0;
      var trasladoY = 0;
```



```
contexto.translate(canvas.width / 2, canvas.height / 2);
contexto.transform(escalaX, inclinacionY, inclinacionX, escalaY,
    trasladoX, trasladoY);
contexto.fillStyle = "blue";
contexto.fillRect(-anchuraRectangulo / 2, -alturaRectangulo / 2,
    anchuraRectangulo, alturaRectangulo);
</script>
</body>
</html>
```



**Rojo inclinado a la izquierda y escala horizontal. Azul inclinado a la derecha y con ajuste de escala vertical.**