P.PORTO

POLITÉCNICO
DO PORTO
ESMAD

COMPUTAÇÃO GRÁFICA
TSIW

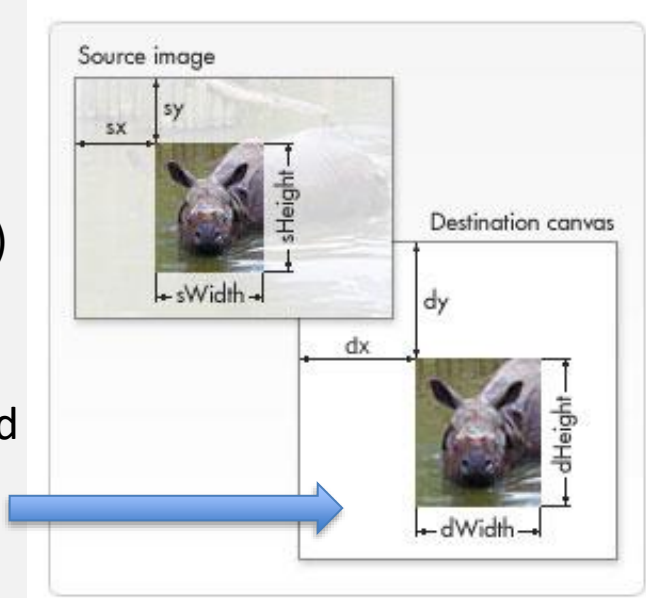
# Syllabus

- Images in Canvas
- Image Data
- Sprite animation

# Images in Canvas

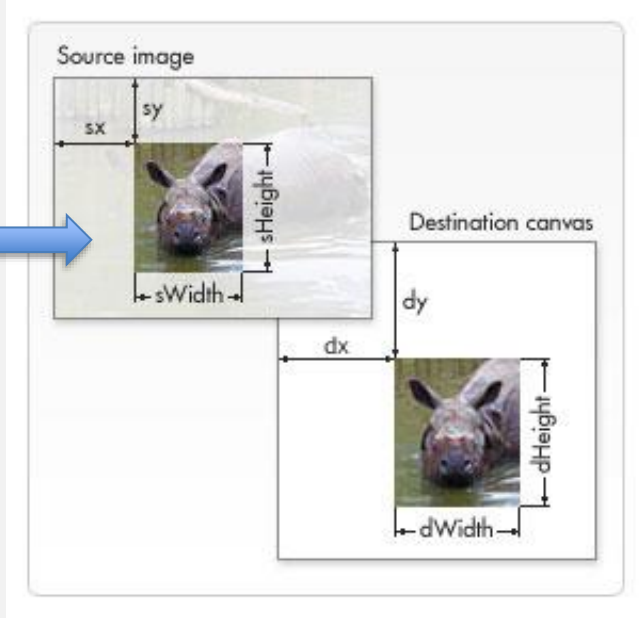
- To draw an image into the HTML Canvas element, use method **`drawImage(image, [sx, sy, sw, sh,] dx, dy, [dw, dh])`**

  - `image`: image source; a JavaScript object, that can be of type
    HTMLImageElement (e.g. image file)
    HTMLCanvasElement (e.g. some other Canvas)
    HTMLVideoElement (e.g. frames from a video)
  - `(dx,dy)`: top left coordinate of the Canvas rectangle where the image is going to be painted – mandatory parameters
  - `dw,dh`: size of the Canvas rectangle where the image is going to be draw
    – optional parameters: can cause image distortion, if not proportional with the image source



# Images in Canvas

- To draw an image into the HTML Canvas element, use method **`drawImage(image, [sx, sy, sw, sh,] dx, dy, [dw, dh])`**
  - `sx,sy,sw,sh`: define the clipped rectangle on the image source to be drawn on Canvas – optional parameters: if not defined, the entire image will be draw




- Examples:

```
//draw the full image, starting in Canvas at point (0,0)
ctx.drawImage(img, 0, 0);
//draw the full image, within a square in Canvas
ctx.drawImage(img, 0, 0, 100, 100);
//draw part of the image, within a square in Canvas
ctx.drawImage(img, 10, 10, 20, 40, 0, 0, 100, 200);
```

# Images in Canvas

- Example 1: draw from image files
  - It is important to wait for the file to be loaded!

```
//create a generic JavaScript image object
let img = new Image();
//define the source path (local image file)
img.src = 'images/ESMAD_rave_1819.jpg';

//wait for the image load event
img.onload = function () {
        //draw the full image, without distortion
        ctx.drawImage(img, 0, 0);
};
```

# Images in Canvas

- Example 2: draw from other Canvas

```
const canvas = document.querySelector('#myCanvas'); //Canvas#1
const ctx = canvas.getContext("2d");
const canvas2 = document.querySelector('#myCanvas2'); //Canvas#2
const ctx2 = canvas2.getContext("2d");

const W = canvas.width; const H = canvas.height;

//Draw something on Canvas#1
ctx.fillRect(10, 10, W-20, H-20);

//Copy from Canvas#1 to Canvas#2, on click event
canvas2.addEventListener('click',function () {
        ctx2.drawImage(canvas, 0, 0);
});
```

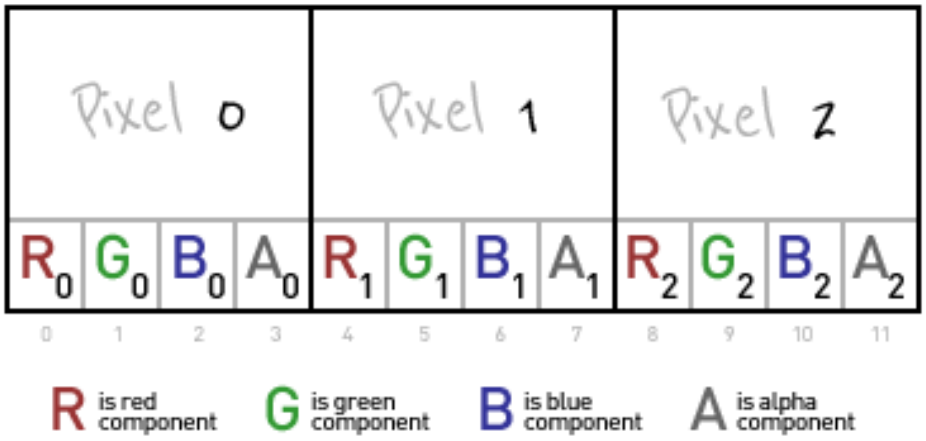
# Images in Canvas

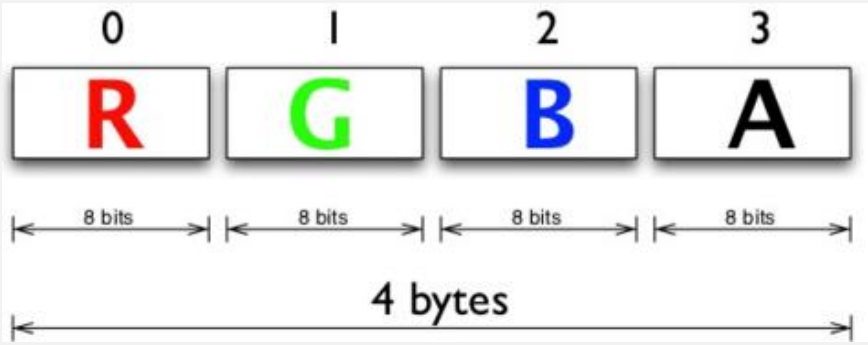
- Example 3: draw from video

```
const canvas = document.querySelector('#myCanvas’); //Canvas HTML element
const ctx = canvas.getContext("2d");

const W = canvas.width; const H = canvas.height;

//Copy frame from video, when paused
const video = document.querySelector("#myVideo"); //some video HTML element
video.addEventListener('pause',function () {
        ctx.drawImage(video, 10, 10, W - 20, H - 20);
});
```

# Image Data

- Canvas has a class to create, store and/or read pixel data: **`ImageData`**
- `ImageData` class has 3 properties:
  - `width`: image width in pixels
  - `height`: image height in pixels
  - `data`: array of pixel color values (of size `width x height x 4`)



# Image Data

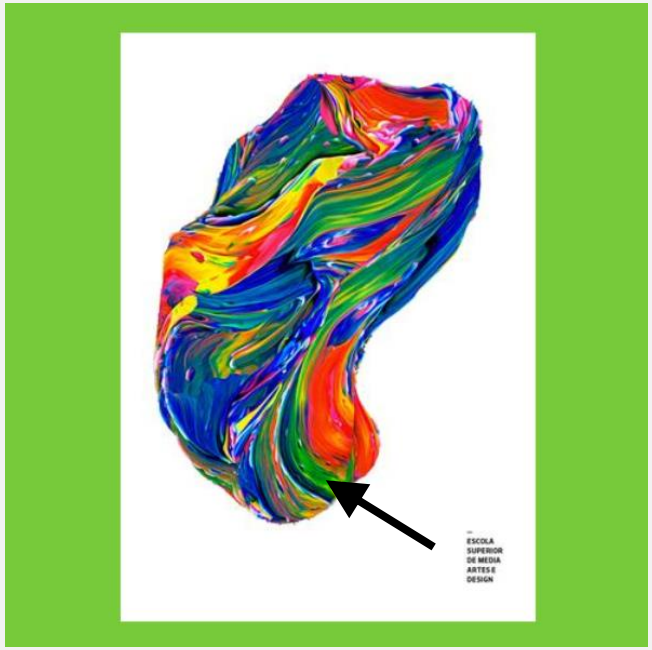
- `ImageData` class has 3 important methods:
  - `createImageData(w,h)`: creates a blank ImageData object
  - `getImageDta(x,y,w,h)`: returns an ImageData object representing the underlying pixel data for a specified portion of the Canvas
  - `putImageData(imageData,x,y[,rX,rY,rW,rH])`: paints data from a given ImageData object onto the Canvas
- Getting data from 1 pixel:

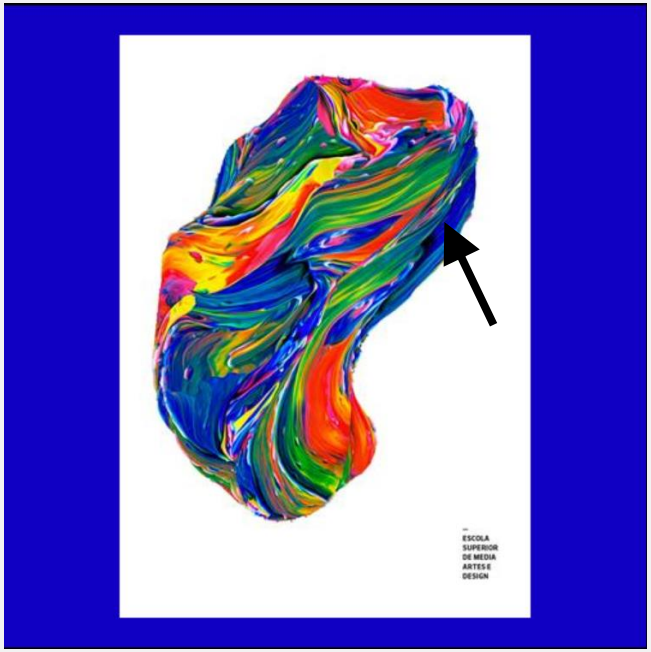
```
let pixel = ctx.getImageData(x,y,1,1)
pixel.data[0] //Red   pixel.data[1] //Green
pixel.data[2] //Blue  pixel.data[3] //Alpha
```

- Methods `createImageData` and `putImageData` are mostly used from pixel manipulation: read about it here and here

# Try yourself….

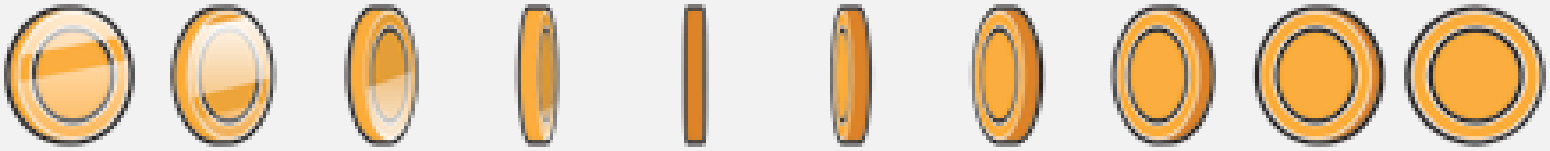
- Use image from Example 1 and draw it on a Canvas of size 500x500 (without distorting the image)
- Get pixel color from mouse cursor coordinates and paint the Canvas background with it

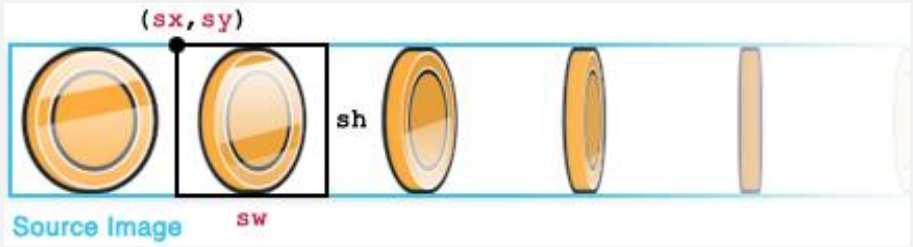


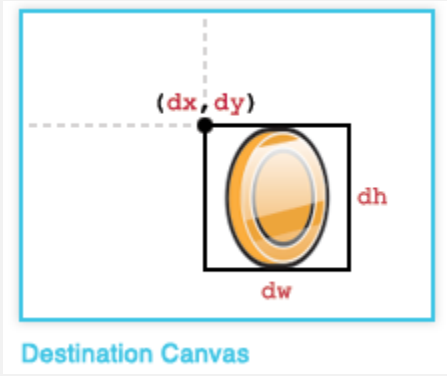



# Sprite Animation

- A **sprite** is an image that can be decomposed into several subimages
- Sprites are used in animation if the subimages are like frames from a video

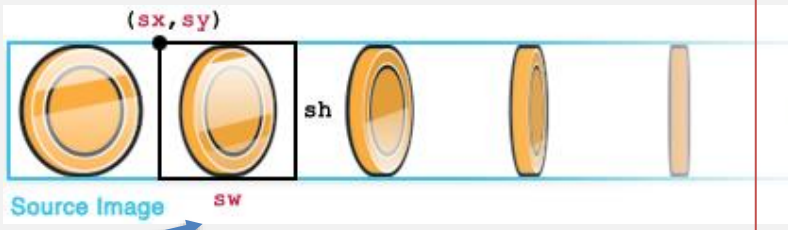


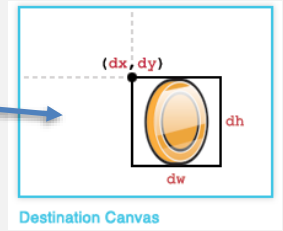


# Sprite Animation

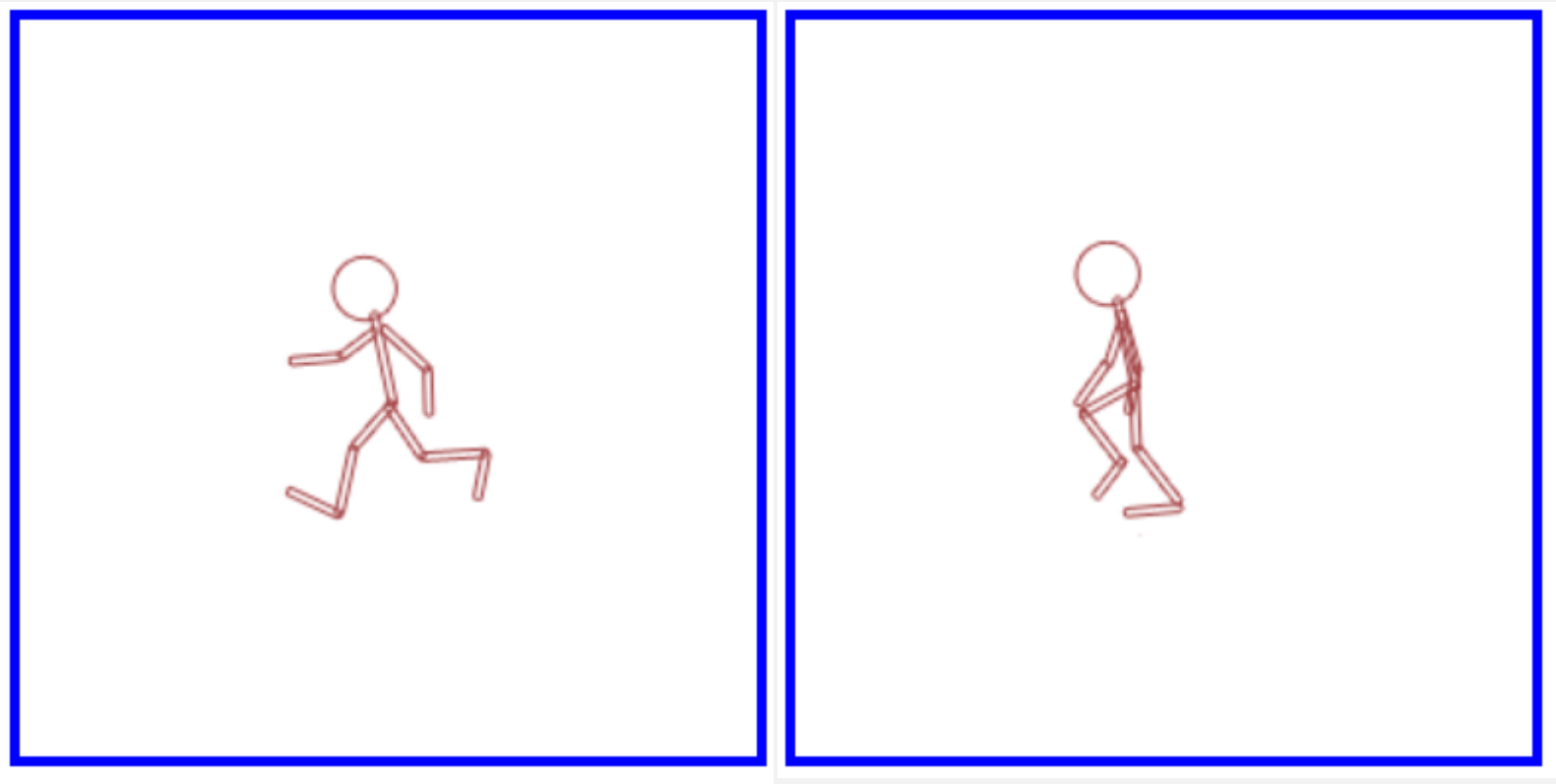
```
let coinImage = new Image();
coinImage.src = 'images/coin-sprite-animation.png';

coinImage.onload = function() {
        setInterval(render, 1000/15); //start animation AFTER image load! - 15 fps
};

//sprite frame counter
let frameIndex = 0;

function render() {
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        ctx.drawImage(coinImage, frameIndex*100, 0, 100, 100,
                                        0, 0, 100, 100);
        frameIndex++;
        if (frameIndex == 10)
                frameIndex = 0; //reset the number of frames counter
}
```



# Try yourself....

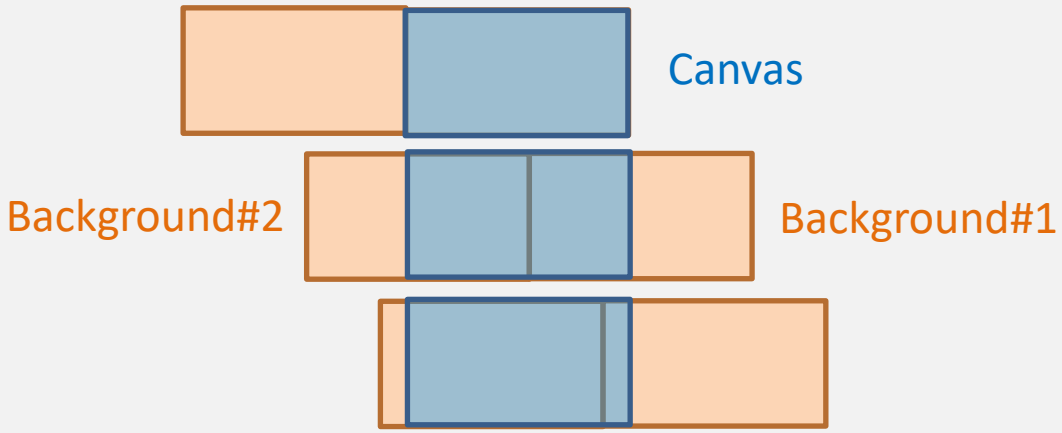
1. Using the image `sprite.png` , animate a walking character, centered in a Canvas of 250x250. The sprite has 5 subimages, each one with 100x100 pixels. Set a delay between frames of 100ms.



# Try yourself....

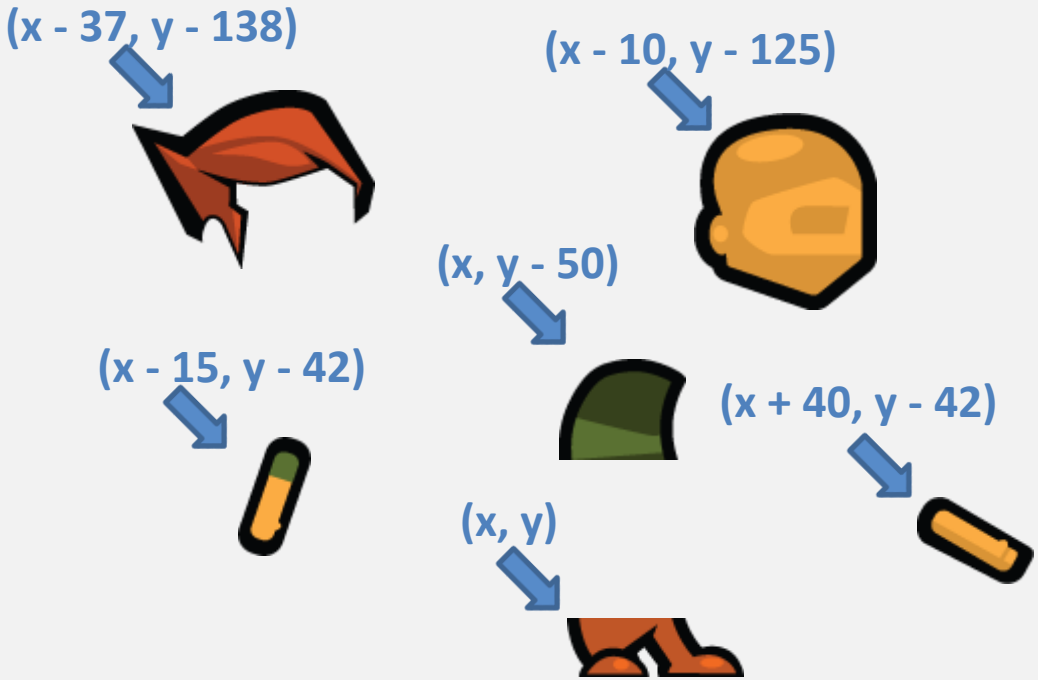
2. To the previous exercise, had a moving background image (**bg.png**). Adjust the character position so that he “walks” over the “grass”. Add a scroll effect to the background, improving the motion illusion of the character.

HINT: draw the image background twice, like the graphics below, making both of them move smoothly to the left




# Try yourself....

3. Using the images and base file **Sprite - Ex3.html** provided in Moodle, draw the following character (do not forget its eyes). The coordinates given below are refered to the legs position (x,y). Use the base files provided in Moodle.



# Try yourself....

3. Make the character “breathe”: the legs and torso do not move; the remaining elements bounce vertically, 4 pixels up and 4 pixels down, from their initial position.