More Canvas

Images

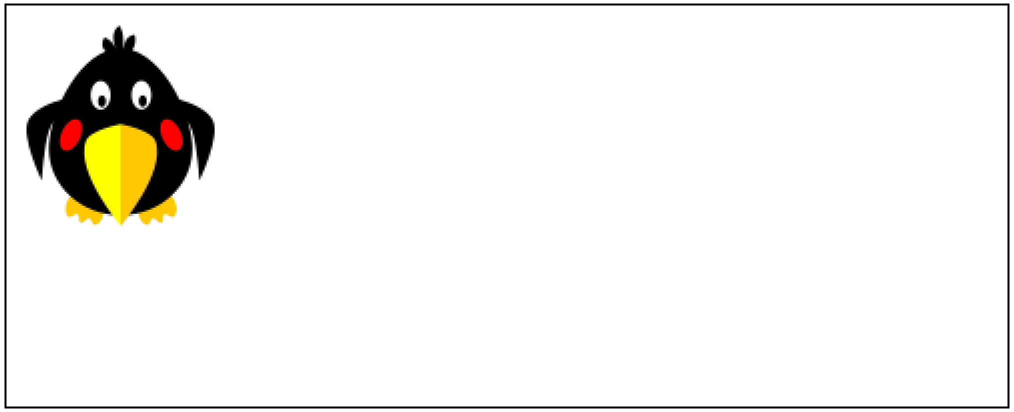
- In computers we talk about vector and bitmap graphics
- So far we've been making vector graphics in canvas
  - We give a logical description of shapes (arc, rect, lines, points)
- Bitmap graphics don't specify shapes, they work with pixels (rasters of colored dots)
- We can use the drawImage() method to draw pixel data onto canvas

drawImage()

- drawImage() takes the pixel data - from an image element or from another canvas element – and draws it onto the canvas
- What you need to be careful of is that your source image has fully loaded before you try to use it to draw from
  - In order to prevent this error from happening we can tell our JavaScript to wait for the image to load
- By default, drawImage will draw the image at its original size.
  - You can also give it two additional arguments to dictate a different width and height
  - `drawImage ( img , x , y, width, height);`

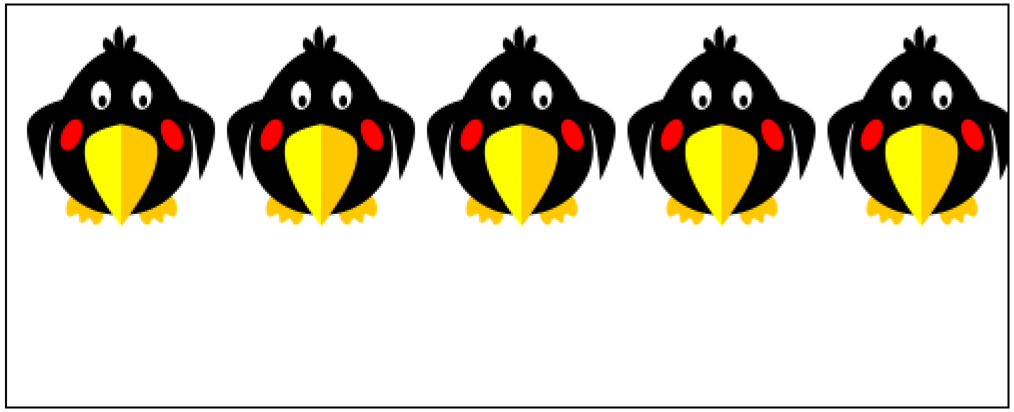
Example - one angry bird

```
var img = new Image();
img.src = "blackbird.png";
img.onload = function () {
   cx.drawImage(img , 10 , 10) ;
};
```



Example – many angry birds

```
var img = new Image();
img.src = "blackbird.png";
img.onload = function () {
  for (var x = 10; x < 500; x +=
100)
      cx.drawImage(img , x , 10) ;
  };
```



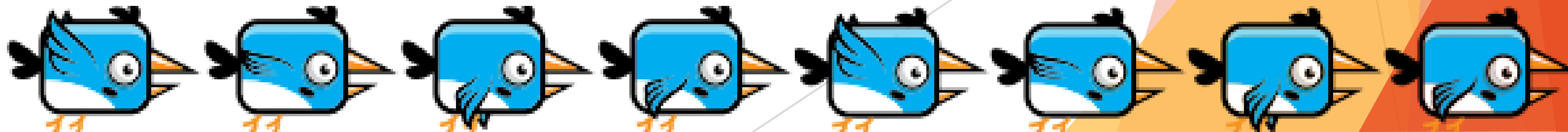
drawImage() cont.

- You can even give drawImage **NINE** pieces of information
- When you do, drawImage uses this information to only draw a portion of the source image
- `drawImage(img, SX, SY, SW, SH, DX, DY, DW, DH)`
- The second through fifth arguments indicate the rectangle (x, y, width, and height) in the source image that should be copied
- The sixth to ninth arguments give the rectangle (on the canvas) into which it should be copied

Why on Earth do we need 9?

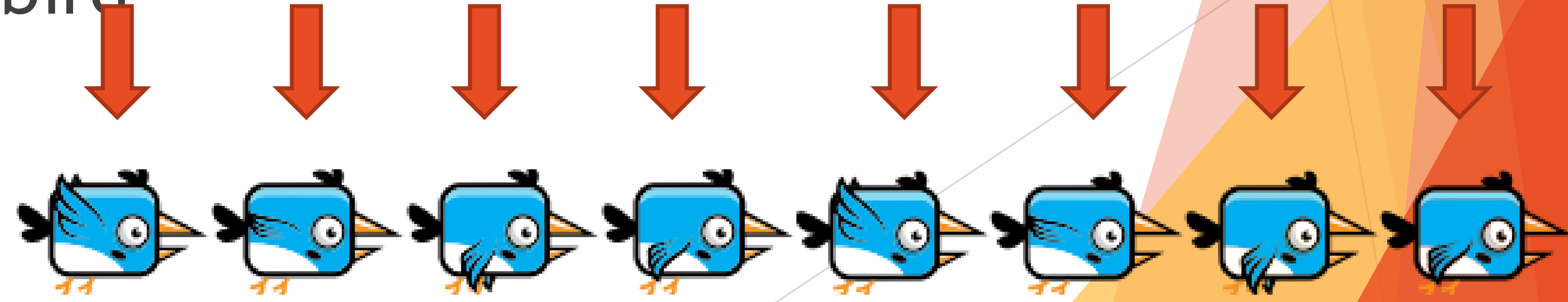
- This is useful for when we are trying to use sprites
- A sprite is a single image file that contains multiple smaller images
- This version of drawImage() is useful because it allows us to draw only the part of image that we need

Epic flappy bird sprite containing a bird character in multiple poses



More Flappy Bird

- By alternating which pose we draw, we can show an animation that looks like a flying character
- We know that this whole image is 600px wide and 100px high
- So each individual sprite (bird) on the sheet is 50px tall and 75px wide
- Knowing these dimensions allows us to animate the bird



But First…

- We need to talk about two more methods
- `clearRect(x,y,w,h)`
  - Is just like fillRect, but instead of coloring the rectangle, it makes it transparent, removing the previously drawn pixels
  - Like a square eraser
- `window.setInterval()`
  - This method calls a function or evaluates an expression at specified intervals (in milliseconds)
  - Ex. To alert "Hello" every 3 seconds (3000 milliseconds):

  -
    ```
    setInterval(function(){ alert("Hello");
    }, 3000);
    ```

Make Flappy Bird Flap

```
var img = new Image();
img.src = "flappy-bird.png";

var spriteW = 75 , spriteH = 50;
img.onload = function () {
   var cycle = 0;
   setInterval ( function () {
          cx.clearRect (0 , 0 , spriteW , spriteH ) ;
          cx.drawImage (img, cycle * spriteW, 0, spriteW, spriteH,0, 0, spriteW,spriteH);
      cycle = ( cycle + 1) % 8;
   }, 120) ;
} ;
```

How Does It Work?

- The cycle variable tracks our position in the animation
- Each frame, it is incremented and then clipped back to the 0 to 7 range by using the remainder operator
- This variable is then used to compute the x-coordinate of the current sprite in the source image

Let’s add a background!

```
var bg = new Image();
bg.src = "city300.png";

var img = new Image();
img.src = "flappy-bird.png";

var spriteW = 75 , spriteH = 50;
img.onload = function () {
     var cycle = 0;
     setInterval ( function () {
          cx.clearRect (0 , 0 , spriteW , spriteH ) ;
          cx.drawImage(bg, 0,0);
          cx.drawImage ( img , cycle * spriteW , 0 , spriteW , spriteH ,0, 0 , spriteW , spriteH ) ;
          cycle = ( cycle + 1) % 8;
     }, 120) ;
} ;
```