LECTURE 8

Clicks and Animation

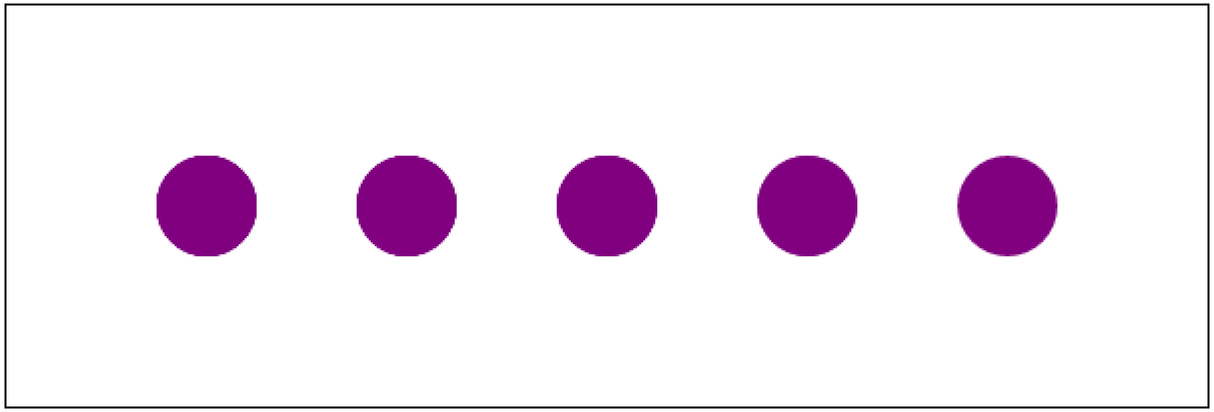
LOOPS AND CANVAS

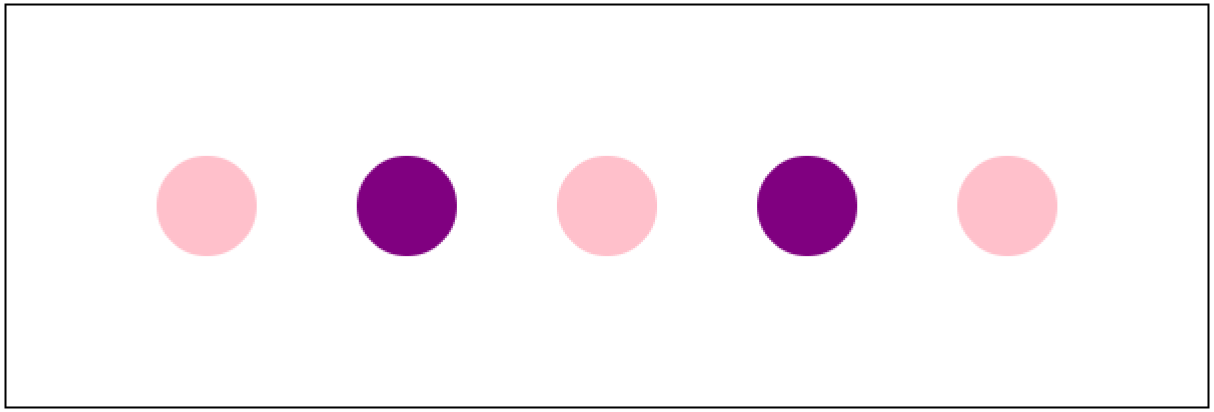
- Have you noticed yourself writing the same code over and over again when drawing on canvas?
- Why not use loops instead!

EXAMPLE

- What if I want to draw 5 purple balls on the canvas, and space them 100px apart?

```
let r = 25;
let x = 100, y=100;
ctx.fillStyle = "purple";
for(let i = 1; i <= 5; i++){
      ctx.beginPath();
      ctx.arc(x, y, r, 0, 2*Math.PI);
      ctx.fill();
      x += 100;
}
```



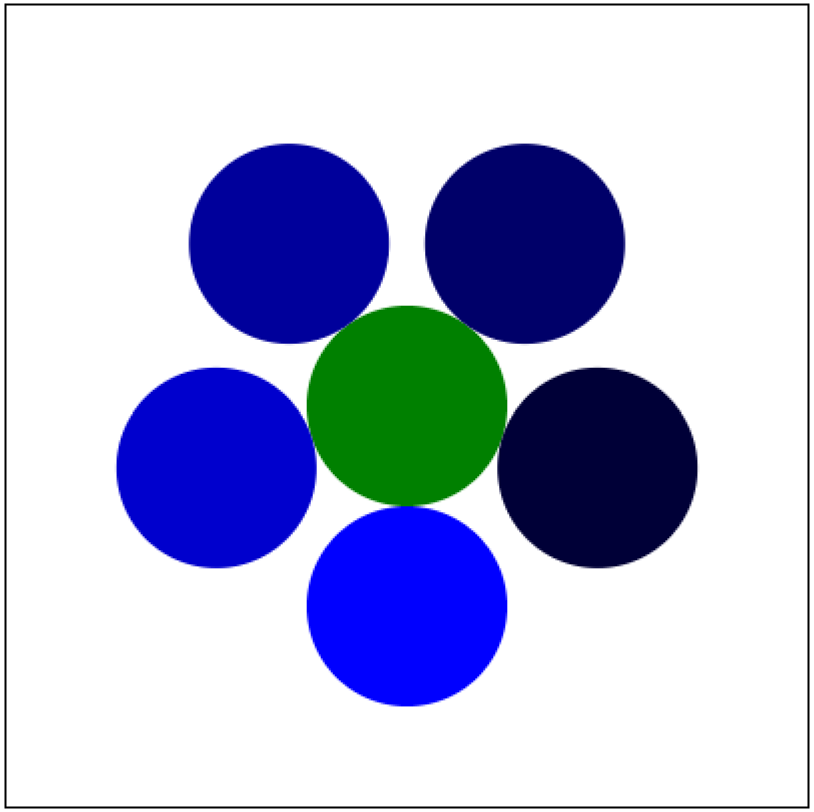


```
let r = 25;
let x = 100, y=100;

for(let i = 1; i <= 5; i++){
     if(i % 2 === 0)
          ctx.fillStyle = "purple";
     else ctx.fillStyle = "pink";

     ctx.beginPath();
     ctx.arc(x, y, r, 0, 2*Math.PI);
     ctx.fill();
     x += 100;
}
```

```
let blue = 255;
let radius = 50;
ctx.beginPath();
ctx.arc(200, 200, radius, 0, 2*Math.PI);
ctx.fillStyle = "green";
ctx.fill();
ctx.save();
ctx.translate(200,200);
for(let i = 0; i < 5; i++){
     ctx.beginPath();
     ctx.arc(0,100, radius,0, 2*Math.PI);
     ctx.fillStyle = "rgb(0,0,"+blue+")";
     ctx.fill();
     ctx.rotate(72*Math.PI/180);
     blue -= 50;
}
ctx.restore();
```



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 (collections of colored dots)
- We can use the drawImage() method to draw pixel data onto canvas

CTX.DRAWIMAGE()

- Takes the pixel data - from an image element or from another canvas element – and draws it onto the canvas

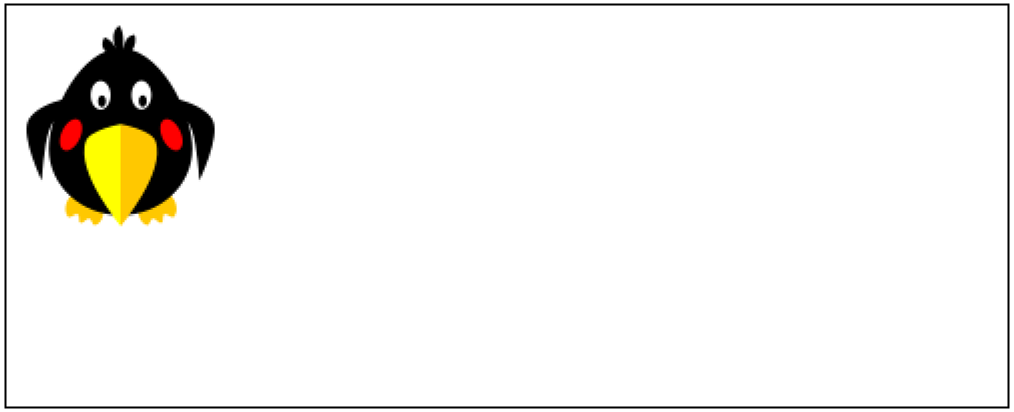
- What you need to be careful of is that your source image has fully loaded before you try to use it to draw from
    - In order to prevent this error from happening we can tell our JavaScript to wait for the image to load using onload

USING DRAWIMAGE()

- **`ctx.drawImage (img, x, y);`**
  - By default, drawImage will draw the image at its original size with the top left corner of the image located at position (x,y) on the canvas
- **`ctx.drawImage (img, x, y, w, h);`**
  - You can also give it two additional arguments to dictate a different width and height

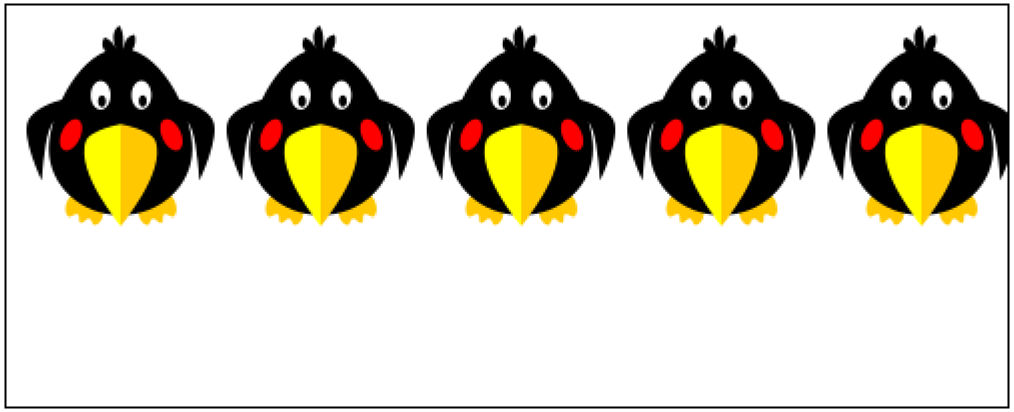
EXAMPLE – ONE ANGRY BIRD

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



EXAMPLE – MANY ANGRY BIRDS

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

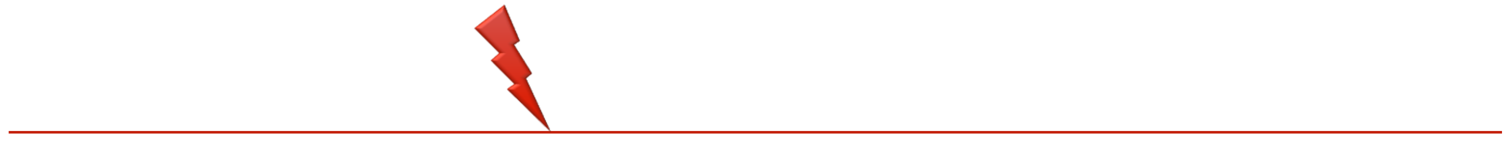


REVIEW: EVENTS

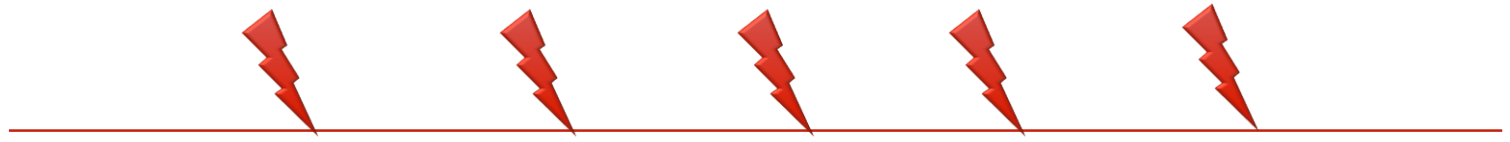
- Events are any thing that happens on the webpage, that our program can respond to
- We've seen Key, Click, Load
- Event handlers are the code that runs as a result of the event
- In our examples so far, we had multiple buttons on the page.
- Each button can have a different event handler associated.
- So far our events have been relatively simple so far

TIMERS

- One time timers
  - Generates an event after a fixed delay



- Periodic timers
  - Generate an event periodically, after a fixed delay
  - Does not stop generating events till you stop it.



- Other stuff can happen in between timer events

PERIODIC TIMERS

- We use the command setInterval() to create periodic timers

```
let timerId = setInterval(eventHandler, interval);
```

Argument	Explanation
eventHandler	A function that will run every time the timer event occurs
interval	The time interval between each timer event, the period of the event. 1/frequency
timerID	The ID to identify the timer event to javaScript. Needed to stop the timer

ONE TIME EVENT

- we can use the setTimeout to create one time events

```
let timerID = setTimeout(eventHandler, delay);
```

Argument	Explanation
eventHandler	A function that will run after the specified delay
delay	The amount of delay before the eventHandler will run.
timerID	The ID to identify the timer event to javaScript. Needed to stop the timer

EXAMPLE - DEMO

- The following example will create a timer that counts down from 10 to 0 and displays the count down on the HTML page

```
let time = 10;
let timer = setInterval(myTimer, 1000);

function myTimer(){
     time = (time - 1);
     timeDisplay.innerHTML = time;
     if(time === 0)
          clearInterval(timer);
}
```

DRAW A PORTION OF AN IMAGE

- 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

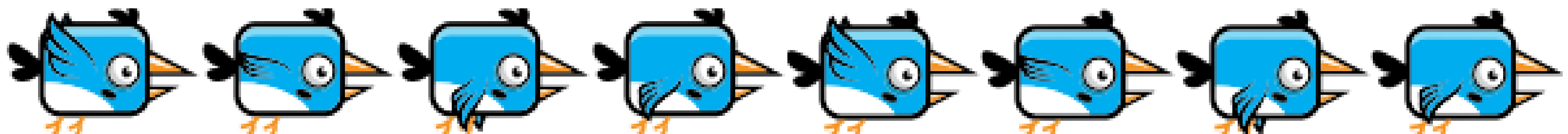
- `drawImage(img,SX,SY,SW,SH,DX,DY,DW,DH)`

- 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?

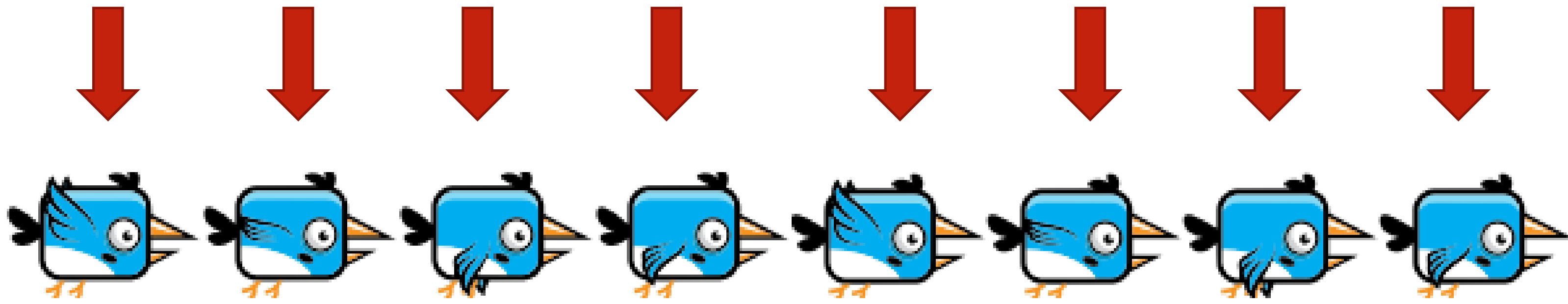
- This is useful for when we are trying to use sprites
- A sprite is a single image file that contains multiple smaller images
- This version of drawImage() is useful because it allows us to draw only the part of image that we need

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 50px high
- So each individual sprite (bird) on the sheet is 50px tall and 75px wide
- Knowing these dimensions allows us to animate the bird



MAKE FLAPPY BIRD FLAP

```
let img = new Image();
img.src = "flappy-bird.png";
let flapping;
let sW = 75 , sH = 50;
img.onload = function () {
     var cycle = 0;
     flapping=setInterval ( function () {
                    cx.clearRect (0 , 0 , sW , sH ) ;
                    cx.drawImage (img, cycle*sW,0,sW,sH,
0,0,sW,sH);
          cycle = ( cycle + 1) % 8;
     }, 120) ;
} ;
```

HOW DOES IT WORK?

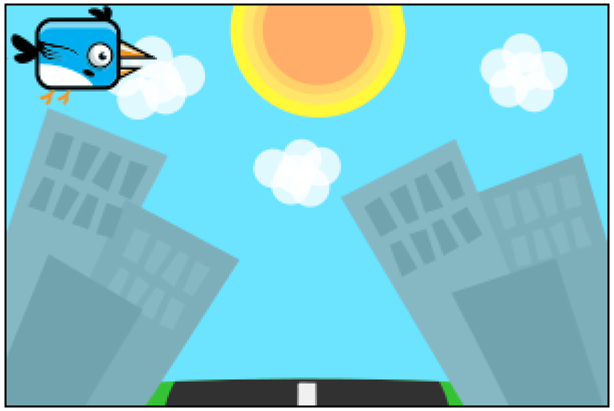
- The cycle variable tracks our position in the animation
- Each frame, it is incremented and then clipped back to the 0 to 7 range by using the remainder operator
- This variable is then used to compute the x-coordinate of the current sprite in the source image

LET'S ADD A BACKGROUND!

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

let img = new Image();
img.src = "flappy-bird.png";
let flapping;
var sW = 75 , sH = 50;

img.onload = function () {
      var cycle = 0;
      flapping=setInterval ( function () {
            cx.clearRect (0 , 0 , sW , sH ) ;
            cx.drawImage(bg, 0,0);
            cx.drawImage ( img , cycle*sW,0,sW,sH
,0,0,sW,sH) ;
            cycle = ( cycle + 1) % 8;
      }, 120) ;
} ;
```



MOUSE EVENT OBJECTS

- Mouse Event objects contains information about mouse events.
- Mouse Events:
  - onclick: When the user clicks and element
  - onmoousemove: occurs when mouse moves while over an element
  - onmouseenter: When the mouse moves over and element
  - onmouseover: when the pointer moves onto an element or one of it's children.
  - onmouseout: When the pointer moves out of an element or one of it's children.

MOUSE EVENT OBJECT

- You can get the location of the mouse cursor, when a mouse event occurs

```
function mouseEventHandler(event){
      console.log(event.clientX);
      console.log(event.clientY);
}
```

Properties	
event.clientX event.clientY	horizontal/vertical coordinate (within current browser window) of the mouse pointer during mouse event
event.offsetX event.offsetY	retrieves the coordinate of the mouse pointer relative to the top-left corner of the parent element of the element that fires the event

EXAMPLE - DEMO

```
<canvas id="canvas1“></canvas>
<h3 id="output">Find Me!</h3>


<script>
let output = document.getElementById("output");
let canvas =
document.getElementById("canvas1");
canvas.onclick = function(){
      output.innerHTML = "X: " + event.offsetX;
      output.innerHTML +=" Y: "+ event.offsetY;
}
</script>
```

EXERCISE

- **How would we detect if a click has occurred inside a square on the canvas?**

```
ctx.fillRect(50,50,100,100);

<script>

let output = document.getElementById("output");

let canvas =
document.getElementById("canvas1");

canvas.onclick = function(){

      output.innerHTML = "X: " + event.offsetX;
      output.innerHTML +=" Y: "+ event.offsetY;

}

</script>
```

EXERCISE

- Write the JavaScript to change the background color of the body of the page every ten seconds
- You should change the color to be one of three **random** colors (red, blue, green)
- Hint:

```
document.body.style.backgroundColor
```

SOLUTION

```
let change = setInterval(setColor, 1000);

function setColor(){
     let random = Math.random();

     if(random < 0.3)
          document.body.style.backgroundColor="red";
     else if(random < 0.6)
          document.body.style.backgroundColor="blue";
     else

     document.body.style.backgroundColor="green";
}
function stopColor(){
     clearInterval(change);
}
```

DEMOS!