JavaScript – Let's Draw!

Use that canvas, play some games

Goal

- Learn to about coordinate system
- Learn to draw lines, rectangles and arc/circles
- Learn to use variables to control size etc.

- We'll cover all the drawing commands, then you'll have a chance to explore

Start with a Canvas

- **CANVAS** is an HTML *element*
- The canvas is a container for graphics (appears like a box)
- Specify the width and height and other properties of the canvas in HTML
- Actual drawing is done in JavaScript

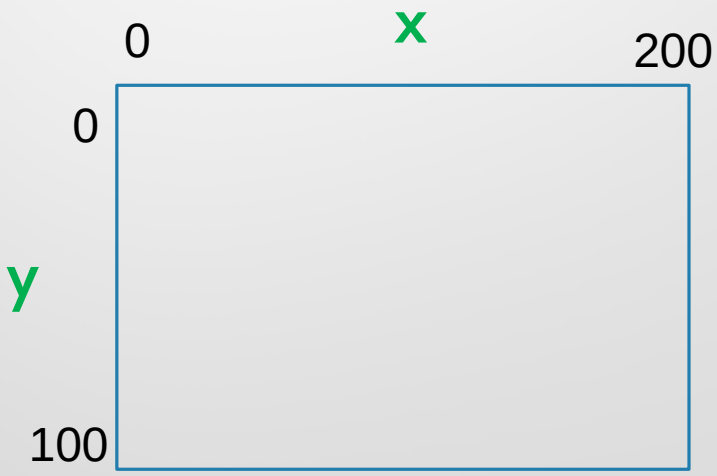
```
<canvas id="myCanvas"
   width="200" height="100"
   style="border:1px solid #000000;">
</canvas>
```

- id is needed so JavaScript can find it
- width/height specify size in pixels
- style is optional
- inside JavaScript, canvas is an **OBJECT** that contains values (soon we'll see width & height) and also methods (see below)

```
// Find the canvas
var canvas =
   document.getElementById("myCanvas");
// Create a drawing object (context)
var ctx = canvas.getContext("2d");
```

Canvas coordinates

- Drawing commands are based on x/y coordinates
- Unlike what you may be used to from a math class, canvas coordinates start with 0,0 in the top left corner
- X coordinates go horizontally from 0 to pixel width specified in HTML
- Y coordinates go vertically from 0 to pixel height specified in HTML
- Assume; `width="200" height="100"`



Draw a line

If you gave a person instructions to draw a line, you might put a mark at the start location and a mark at the end location, then tell the person to draw a line between those marks.

JavaScript commands

- `beginPath();` starts a new path (each path can have attributes like color)
- `moveTo(x,y);` tells JS the start location
- `lineTo(x,y);` tells JS the end location
- `stroke();` actually draws the line
- `strokeStyle = 'red';` optional, sets style (color, gradient, pattern)

Pedagogy sidebar: You might want small exercises so students can practice drawing one type of object (e.g., lines) before moving on to other shapes. We're going full speed ahead!

Draw Rectangle

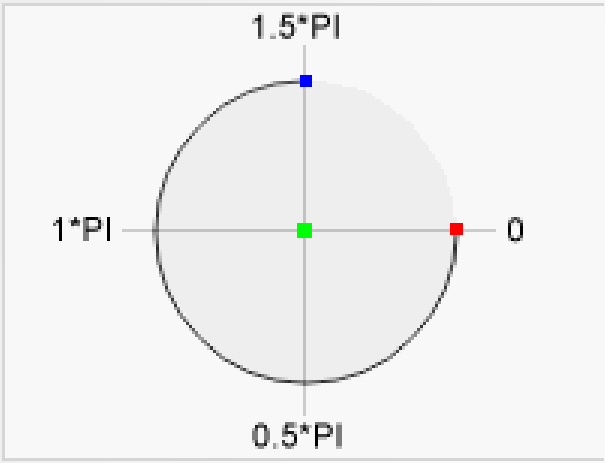
- Start with canvas and context
- Four parameters
  - Upper x coordinate
  - Upper y coordinate
  - Width
  - Height
- Can set strokeStyle/linewidth
- rect command draws

```
ctx.rect(5, 5, 290, 140);
```

- For a solid rectangle, set fillStyle
- fillRect command draws

```
ctx.fillStyle = "green";
ctx.fillRect(30, 30, 50, 50);
```

Draw a Circle

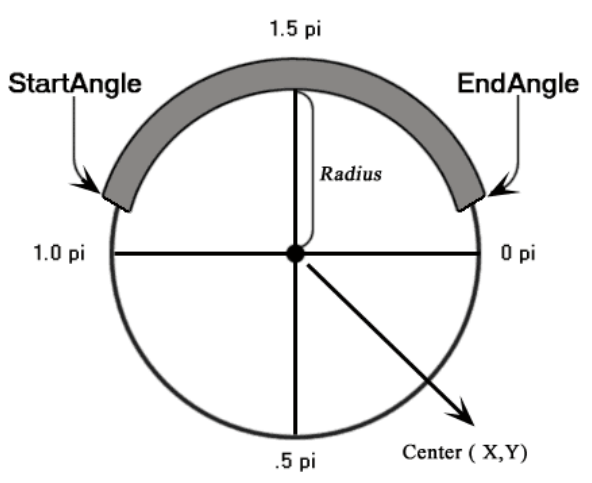


context.arc(*x*,*y*,*r*,*startAngle*,*endAngle*,
counterclockwise);

- x: x-coordinate of center
- y: y-coordinate of center
- r: radius of circle
- startAngle: start angle in radians, with 0 corresponding to 3:00 position
- endAngle: end angle in radians
- Counterclockwise: optional, false is default

To create a circle with arc(): Set start angle to 0 and end angle to 2*Math.PI.

Draw an Arc



- Same parameters, but you'll need to figure out angles and may want to change the direction

Pedagogy sidebar: You might want to provide example files (as I have), which provide similar function to TryItEditor of w3schools.

We're still moving full speed ahead, but you'll have a chance to try it in just a minute.

https://www.w3resource.com/html5-canvas/html5-canvas-arc.php

Use Variables to Draw

- For some drawings, elements will be positioned relative to each other
- Instead of hard-coding values, try to plan the drawing and use variables
- You may want to base sizing on width and height of canvas.

```
var canvas =
   document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle="lightGray";
// Create gray “background” for entire canvas
ctx.fillRect(0,0,canvas.width,canvas.height);
```

Recap

To draw in Javascript

1. Find the canvas (name match HTML)
2. Create the context (drawing object)
3. Set the style(s) as desired

- For a line
  - Create a path, determine start/end locations, stroke
- For a rectangle
  - Use draw or fill commands with desired coordinates
- For a circle
  - Determine parameters based on picture with radians

Relax and Play

- Experiment with the provided files
- Ask questions!

- We do have a more structured exercise coming up.

Pedagogy sidebar: Unstructured play may not work with your students. We have a lot to cover in a short time.

JavaScript in separate file

- For more complex programs, it's cleaner to have the JavaScript in a separate file
- File extension is .js
- As with CSS, we must tell the HTML file what type of file and where to find it:

```
<script type="text/javascript"
   src="facesPlan.js"></script>
```

- We've seen button clicks as one type of ***event***. Another type of event is when the program first starts. This event is named *onload*.

```
window.onload = function() {
   // alert("here");
   // draw the face when the window loads
   drawFace(happy);
}
```

Pedagogy sidebar: Challenge your assumptions. If it looks like function isn't doing anything, see if it's even being called.

Exercise: Draw a Smiley Face

Plan first!

- Plan the variables on paper - Worksheet
  - I will come around and share my design with you
- Use the provided facesPlan.js template
- Test incrementally (see examples)
- Read Pedagogy on next slide

Pedagogy

- Students often struggle with planning
- Specify program structure via comments (see facesPlan.js)
  - Comments identify sub-tasks
  - Easier to support if all students use similar approach
  - Students more likely to have success (blank page is intimidating)
- Specifying function parameters also helpful for novice programmers (see DrawCircle)
  - Think about what info will function need
  - Use meaningful names to get parameter order correct.
- Writing comments first is an actual technique used by some professionals
- Encourage students to use code from examples (remember, effective use of examples is critical)
- As students gain experience, require them to do this planning step

DO NOT write the entire program then try to run it!!!!! Model incremental testing. Comments can specify intermediate test points. If you can get students to do this, your job will be MUCH EASIER!

Optional Extension – more challenging

- Add a parameter to drawFace method. If parameter is set to `0`, draw happy face. If set to `1`, draw sad face.
- TEST – just hard-code function call to either 0 or `1` in the onload method.
- Add this button to your html (div just helps with spacing)

```
<div>
   <input type="button"
   id="toggle" value="Make me sad"
   onclick="toggleIt();" />
</div>
```

- Add a global var named `happy`
- Write the method named `toggleIt` which:
  - if `happy` = 0, set to `1` (and vice versa)
  - calls `drawFace` with `happy` as a parameter
  - sets the text of the button. Hint: you know how to find an element by id (in this case, "toggle"). Setting the button text is like the syntax for innerHTML, but instead of innerHTML you will set value. We covered innerHTML at the end of yesterday's JS lecture.