P.PORTO

POLITÉCNICO
DO PORTO
ESMAD

COMPUTAÇÃO GRÁFICA
TSIW

Syllabus

- HTML element: Canvas
- Context 2D and coordinates
- Simplest shape: rectangle
- Styling

Why Canvas?

- Interactivity
- Animation
- Flexibility
- Browser/Platform Support
- Popularity
- Web standard
- Develop once, run everywhere
- Free and accessible development tools

HTML element: Canvas

- HTML5 specs includes new functionalities, one of them being the Canvas element

"resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, art, or other visual images on the fly"

- Drawing and animation is performed by scripting (**JS**)

Canvas (basic support) - LS

Usage: Global — % of all users: 98.4% + 1.21% = 99.61%

Method of generating fast, dynamic graphics using JavaScript.

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	Ka Bro
6-8		[1] 2-3.5		[1] 3.1-3.2				[1] 2.1-2.3								
9-10	12-93	3.6-92	4-93	4-14.1	10-79	3.2-14.8		3-4.4.4	12-12.1				4-14.0			
11	94	93	94	15	80	15	[2] all	94	64	94	92	12.12	15.0	10.4	7.12	2
		94-95	95-97	TP												

https://caniuse.com/?search=canvas

HTML element: Canvas

HTML

```
<!DOCTYPE html>
<html>
   <head>
       <title>Your First Canvas Application </title>
   </head>

   <body>
        <canvas>
            Your browser does not support HTML5 Canvas.
        </canvas>
   </body>
</html>
```

What do you see on your browser?

HTML element: Canvas

HTML

```
<!DOCTYPE html>
<html>
   <head>
       <title>Your First Canvas Application </title>
        <style>
            canvas {
                border: 3px solid orange;
            }
        </style>
   </head>

   <body>
        <canvas>
            Your browser does not support HTML5 Canvas.
        </canvas>
   </body>
</html>
```

CSS

And now?

HTML element: Canvas

- Canvas element has no content nor border
- There can be more than one Canvas element in a single webpage
- Default size is `300px` by `150px`
- Like other HTML elements, it has several attributes:
  `id`, `width`, `height`, `style`, …

```
<canvas id="canvas1" width="200" height="100" style="border:1px solid blue;">
    Your browser does not support HTML5 Canvas.
</canvas>
```

- Text between tags is the *fallback content*
- Given an `id`, it is **accessible using JavaScript**

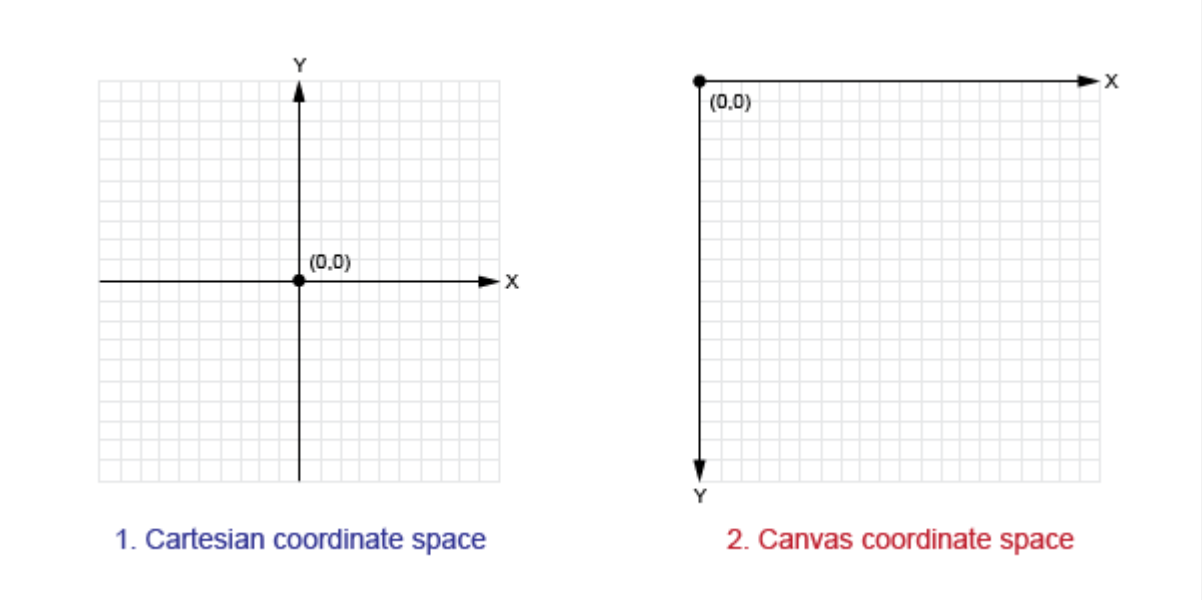
JS object: HTMLCanvasElement

JS

```
// Getting the DOM element
const canvas = document.querySelector("#canvas1");

// Getting its 2D rendering context
const ctx = canvas.getContext("2d");
```

- **canvas**: Canvas object in JavaScript
- Two properties: `width` and `height`
  - sets both the element's size and the size of the element's drawing surface
- Three methods, being the principal `getContext()`:
  - sets the **rendering context** bound to it: 2D, webgl, webgl2 (⚠),…

Canvas grid coordinates

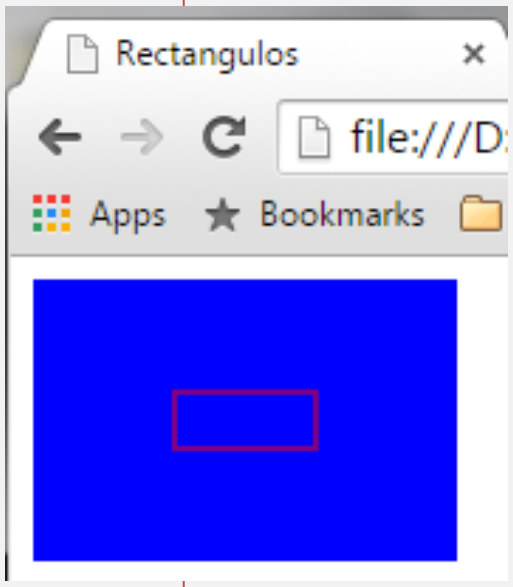


- Points outside the Canvas grid boundaries (`width` and `height`) are not drawn
- It is possible to modify the Canvas coordinate system using transformations (will be learned later…)

Simplest shape: rectangle

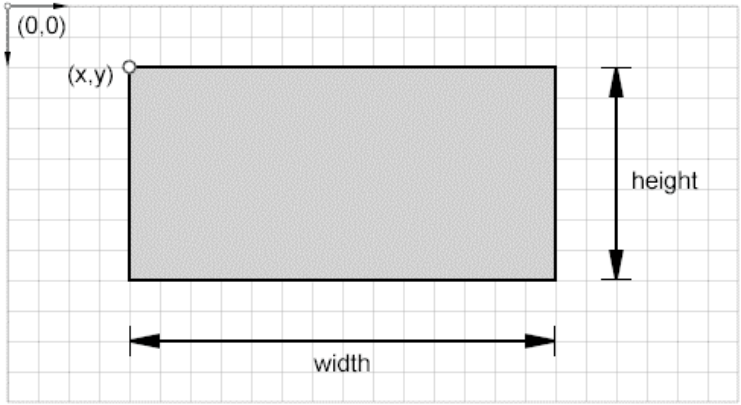
JS

```
// Getting the DOM element
const canvas = document.querySelector("#canvas1");

// Getting its 2D rendering context
const ctx = canvas.getContext("2d");

// Drawing a blue solid rectangle
ctx.fillStyle = 'blue';
ctx.fillRect(0, 0, 150, 100);

// Drawing a smaller red rectangle
ctx.strokeStyle = 'red';
ctx.strokeRect(50, 40, 50, 20);
```



- All Canvas drawings must be performed using JS **in its rendering context** - in the example: **`ctx`**

Simplest shape: rectangle



`fillRect(x,y,width,height)`

paints the given rectangle, using the current fill style

`strokeRect(x,y,width,height)`

paints the box that outlines the given rectangle, using the current stroke style

`clearRect(x,y,width,height)`

clears all pixels in the given rectangle to transparent black

`rect(x,y,width,height)`

draws the given rectangle has part of a path (will be learned later…)

Styling

COLORS

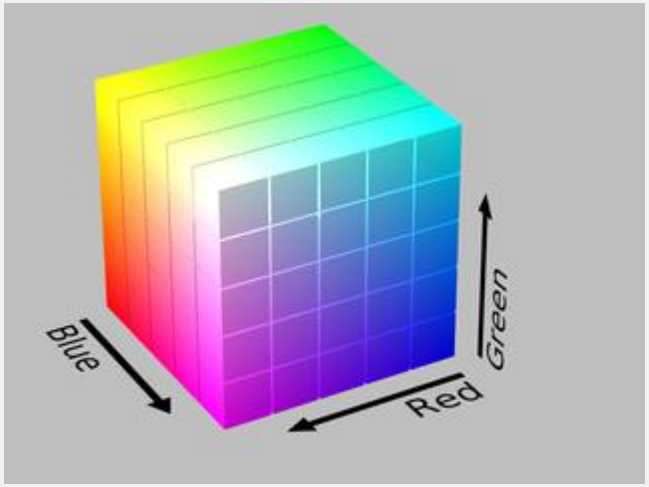
- Use properties **`fillStyle`** and **`strokeStyle`** to define an object's color
- Possible values:
  - Color name (EN) from CSS (https://developer.mozilla.org/en-US/docs/Web/CSS/named-color)
  - Hexadecimal values
  - Decimal values, using function `rgb()`

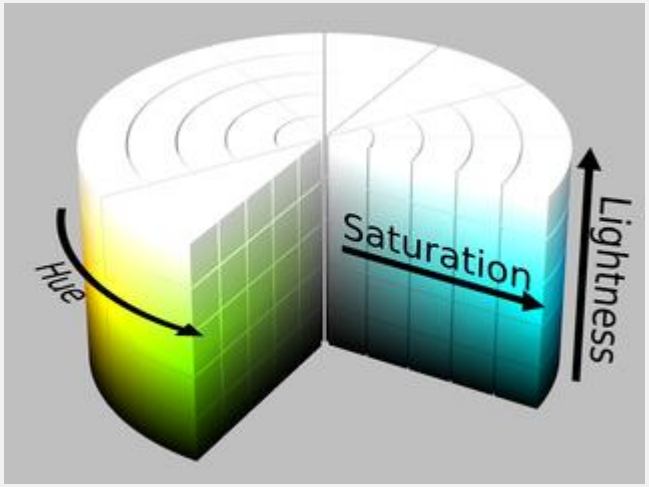
Color	Name	HEX	HEX (abreviated)	RGB
	`red`	`#FF0000`	`#F00`	`rgb(255,0,0)`
	`green`	`#00FF00`	`#0F0`	`rgb(0,255,0)`
	`blue`	`#0000FF`	`#00F`	`rgb(0,0,255)`

Styling

COLORS

- Only the objects drawn **after a color change** are affected
- Current browsers support CSS3 colors and allow the use of color spaces other than RGB: RGBA, HSL, and HSLA





Styling

TRANSPARENCY/OPACITY

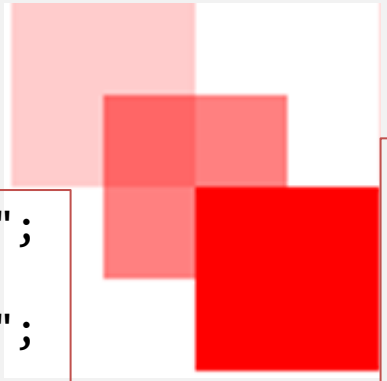
- A (*alpha*): float value to define opacity

  0: totally transparent

  1: totally opaque

- Opacity can also be defined using property **`globalAlpha`**

```
ctx.fillStyle = "rgba(255,0,0,0.2)";
ctx.fillRect(0,0,50,50);
ctx.fillStyle = "rgba(255,0,0,0.5)";
ctx.fillRect(25,25,50,50);
ctx.fillStyle = "rgba(255,0,0,1)";
ctx.fillRect(50,50,50,50);
```



```
ctx.fillStyle = "red";
ctx.globalAlpha = 0.2;
ctx.fillRect(100,0,50,50);
ctx.globalAlpha = 0.5;
ctx.fillRect(125,25,50,50);
ctx.globalAlpha = 1;
ctx.fillRect(150,50,50,50);
```

Styling

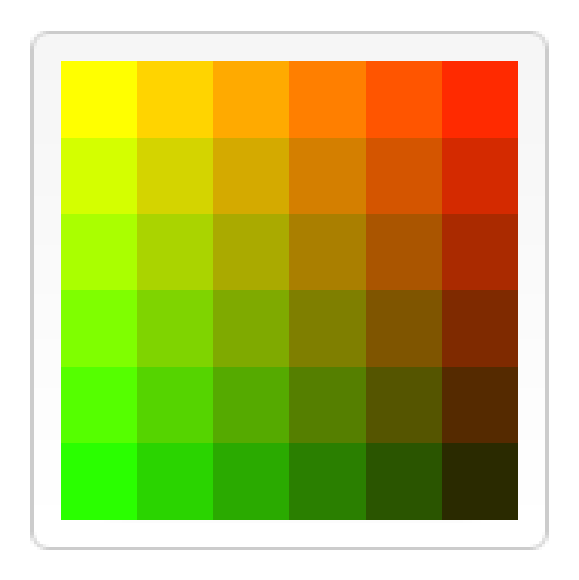
COLORS & TRANSPARENCY/OPACITY

```
"#f00"              // Hexadecimal RGB value: red
"#00ff00"           // RRGGBB value: green

"rgb(60, 60, 255)"      // RGB as integers: blue
"rgb(100%, 25%, 100%)" // RGB as percentages: purple
"rgba(100%,25%,100%,0.5)"   // Plus alpha 0-1: translucent
"rgba(0,0,0,0)"         // Transparent black

"transparent"           // Synonym for the above

"hsl(60, 100%, 50%)"    // Fully saturated yellow
"hsl(60, 75%, 50%)"     // Less saturated yellow
"hsl(60, 100%, 75%)"    // Fully saturated, lighter
"hsl(60, 100%, 25%)"    // Fully saturated, darker
"hsla(60,100%, 50%, 0.5)"   // 50% opaque
```

Styling

COLORS & TRANSPARENCY/OPACITY

```
const ctx = document.getElementById("canvas").getContext("2d");
  for (let i = 0; i < 6; i++) {
    for (let j = 0; j < 6; j++) {
      ctx.fillStyle = `rgb(${Math.floor(255-42.5*i)}, ${Math.floor(255-42.5*j,)}, 0)`;
      ctx.fillRect(j * 25, i * 25, 25, 25);
    }
  }
```



Try yourself…

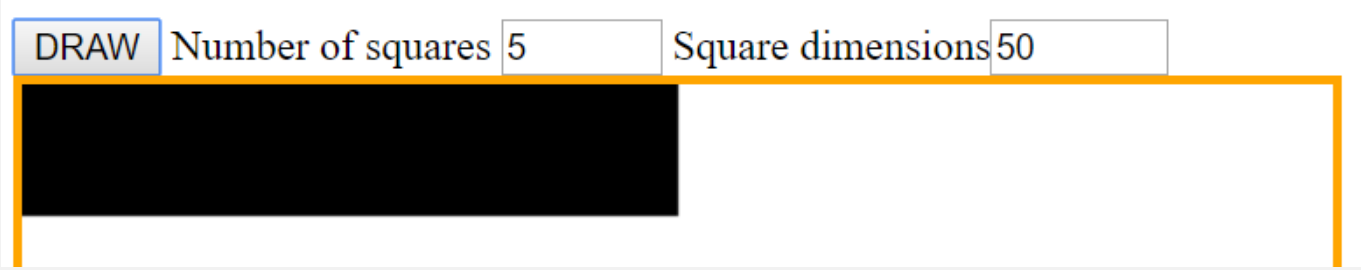
1. Can you try to make flags of countries like:
   Germany
   France
   Sweden
   Switzerland



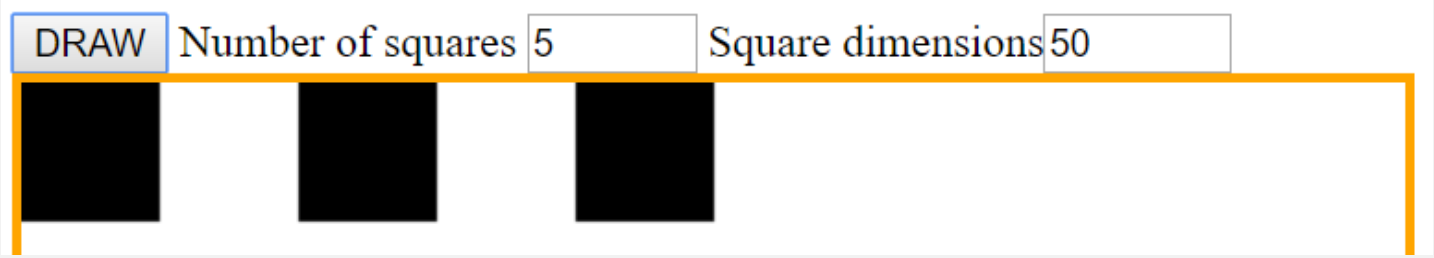

Try yourself…

2. Now make use of your algorithmic skills to draw a configurable black-and-white checkerboard

a) Draw 1 row of N black squares of size D

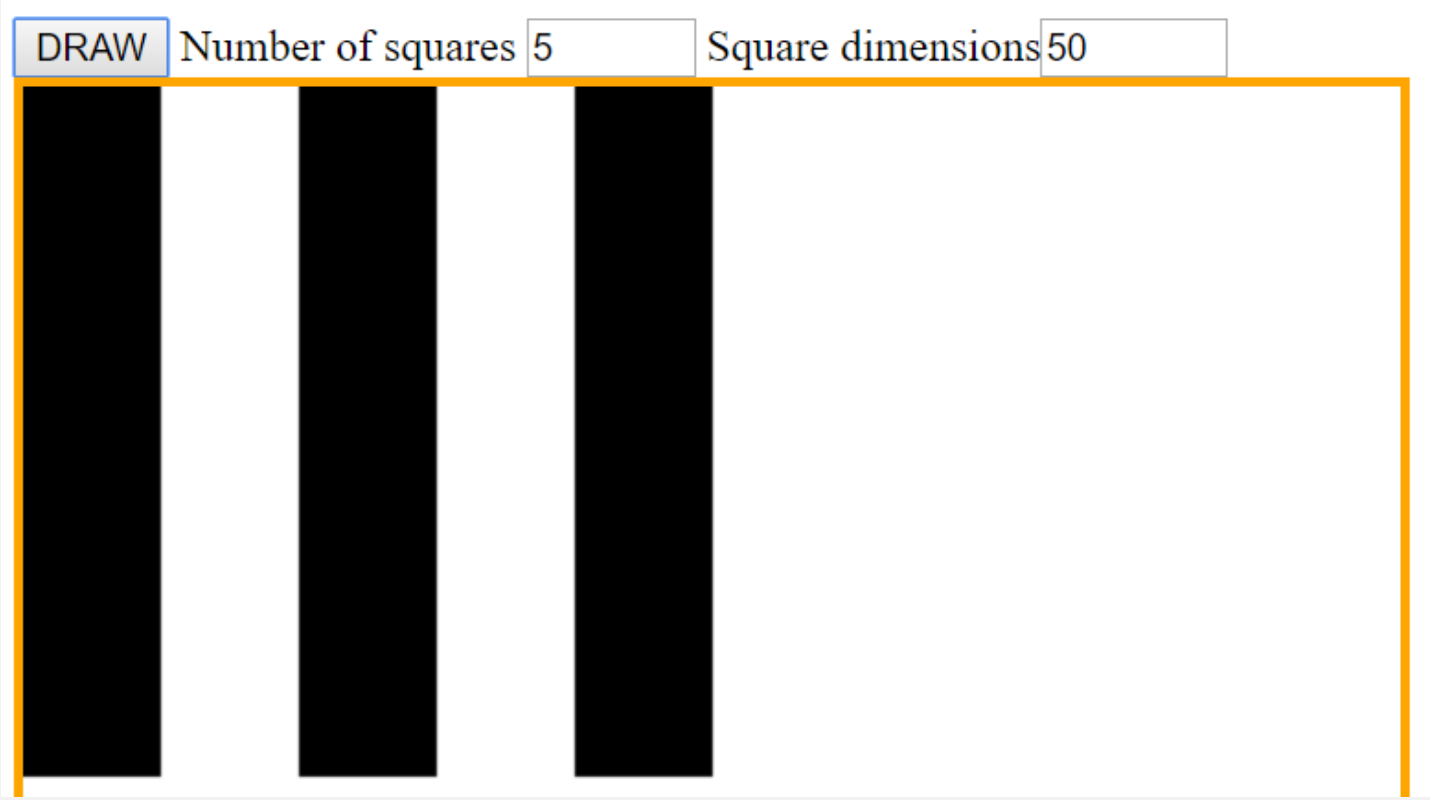


b) From that row, skip drawing the even squares



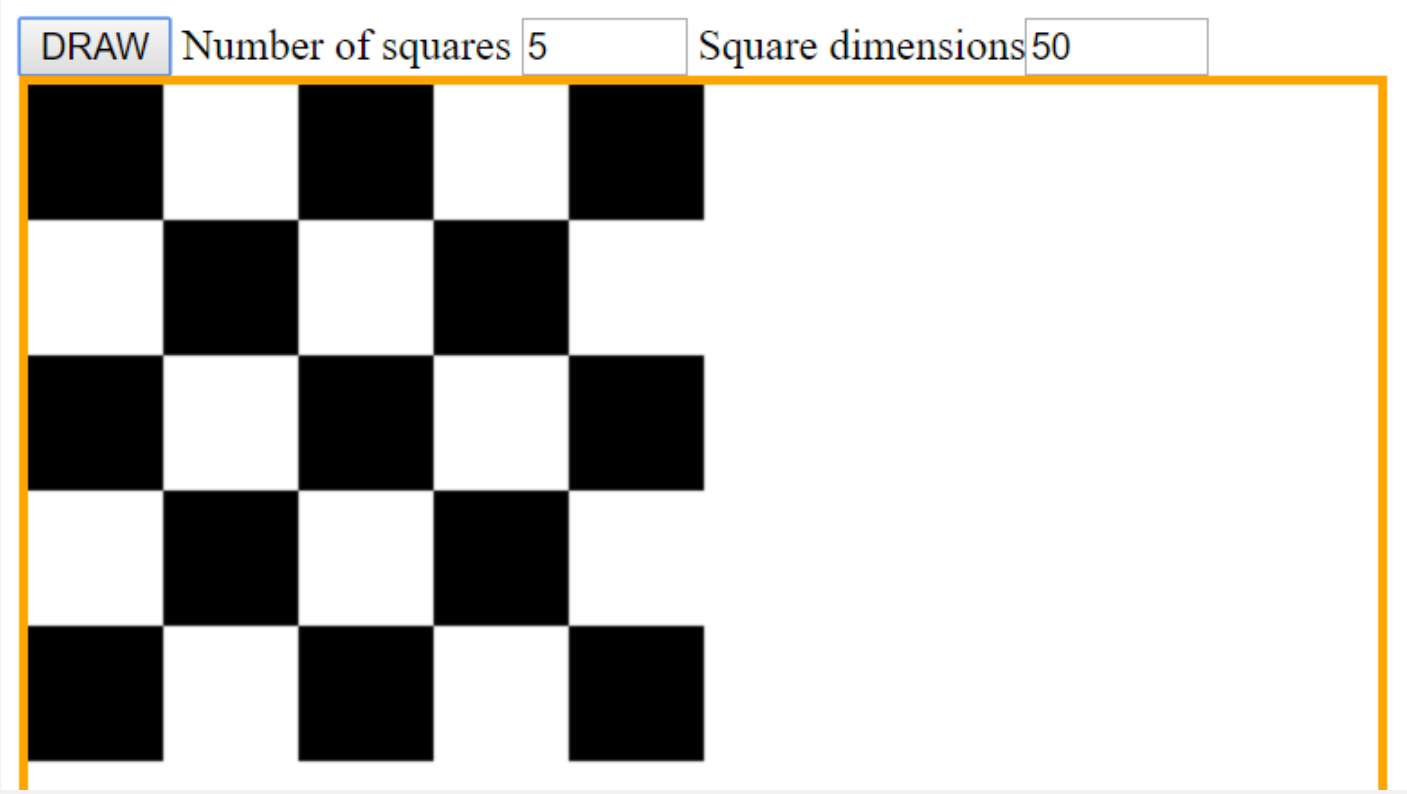
Try yourself…

c) Draw N rows of black squares in odd positions



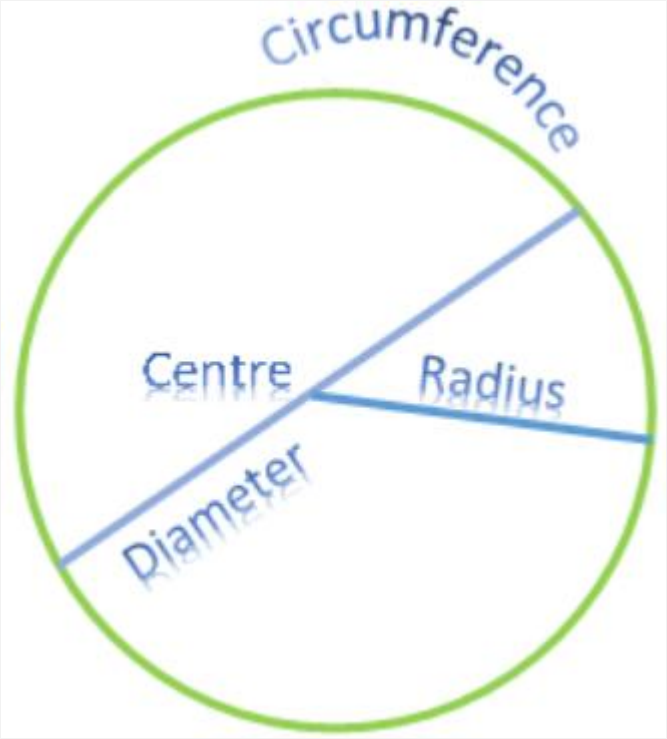
Try yourself…

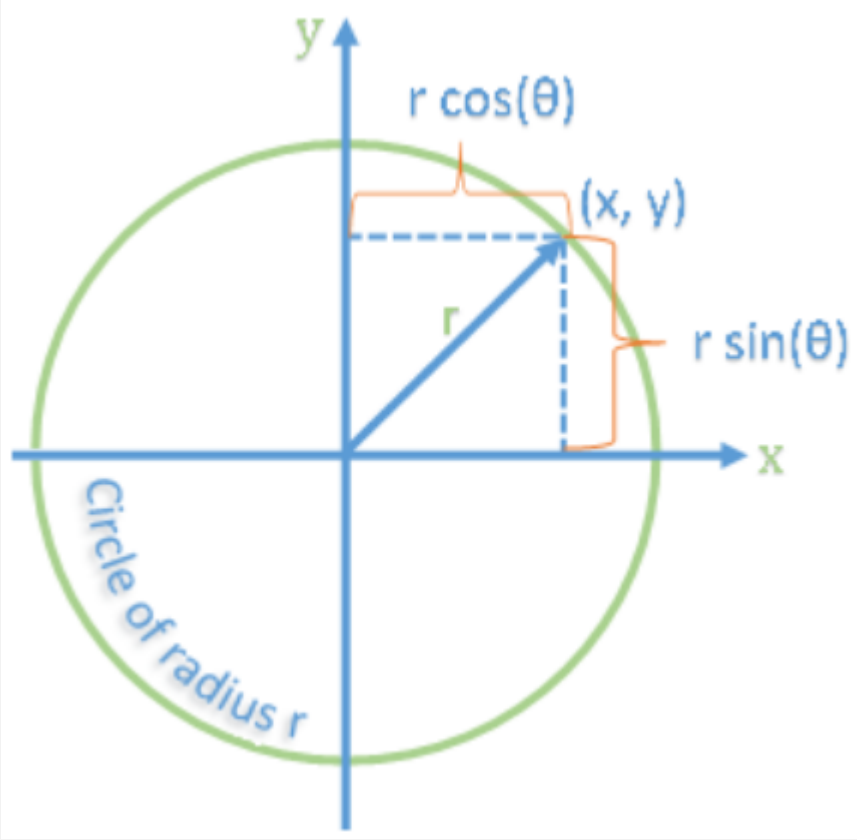
d) Now, fix the skipping part of the algorithm to draw a proper checkerboard



Try yourself…

3. Finish by refreshing some 2D geometry





Try yourself…

Knowing the:

- **Parametric** circumference equations:

$$\begin{cases} x = x_0 + r\cos t \\ y = y_0 + r\sin t \end{cases}$$

- $360° = 2\pi \text{ radians}$

Draw N circles equally spaced around a circle, centered in the Canvas element, with a radius of 100 pixels.

Try yourself…