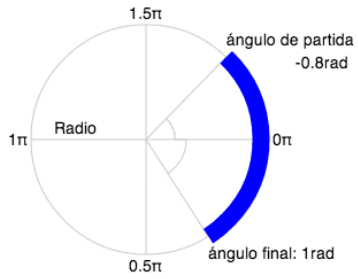
Arcos y círculos

Repaso de geometría



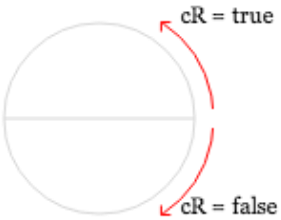
En el canvas para trabajar con ángulos, utilizamos radianes.
Un circulo completo tiene 2π radianes, con lo cual podemos escribir: `360° = 2π rad;`
o si hablamos JavaScript: `2*Math.PI;`
Para convertir grados sexagesimales a radianes utilizamos la siguiente fórmula:

```
radianes = (π / 180) * grados;
```

lo que traducido a JavaScript es:

```
var radianes = (Math.PI / 180) * grados;
```

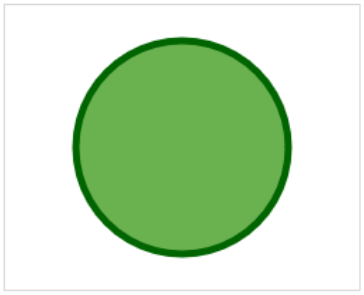
Dibujar un arco



Para dibujar un arco utilizamos el método **`arc(x, y, r, ap, af, aC)`**, donde
- **`x`** e **`y`** son las coordenadas del centro,
- **`ap`** es el ángulo de partida (*en radianes*),
- **`af`** es el ángulo final (*en radianes*), y
- **`cR`** (*contra reloj*) es la dirección.

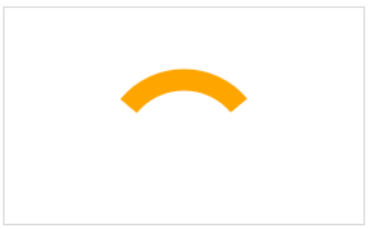
El parametro **`cR`** puede tomar dos valores:
- **`true`** (*verdadero, o sea: en sentido contrario al de las agujas del reloj*) y
- **`false`** (*falso, o sea: en el sentido de las agujas del reloj*).

Ejercicio 1:

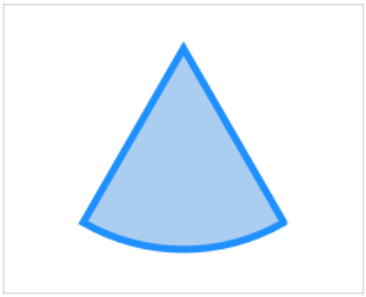


La propiedad **`strokeStyle`** permite definir el color del borde. El método **`strokeRect()`** permite dibujar el borde de un rectángulo y el método **`stroke()`** permite dibujar un borde (para un círculo, en el ejemplo siguiente).

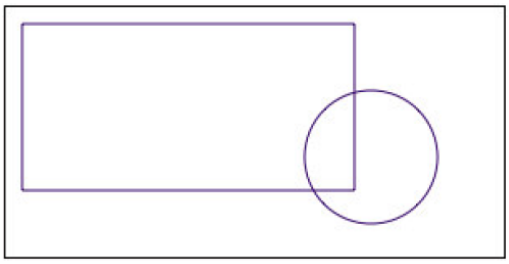
Ejercicio 2:



Ejercicio 3:



Ejercicio 4:



Dibujar un óvalo

Para dibujar un óvalo, dibujaremos un **círculo**, que aplastaremos luego, utilizando el método **scale().**

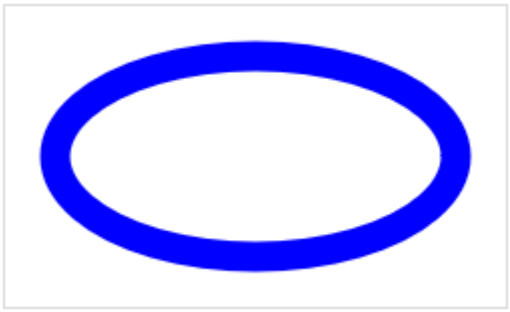
El método **scale(x,y)** reduce o amplía a escala el dibujo actual, y acepta dos parámetros: la deformación en **x**, y la deformación en **y**. Por ejemplo: si y = 0.5 la altura se verá reducida a un 50%. Si por el contrario x = 2, la anchura aumentara a 200% de su valor inicial. En nuestro caso queremos que la anchura aumente a 200% de su valor inicial, y la altura quede sin cambiar.

```
ctx.scale(2, 1);
```

Importante: para centrar el óvalo resultante hemos dibujado el círculo original hacia la izquierda.

```
ctx.arc( centroX/2, centroY, radio, 0, 2 * Math.PI );
```

Ejercicio 4:



Dibujar una elipse

En este momento podemos dibujar una elipse en canvas, utilizando el método **`ellipse()`**. La mala noticia es que este método funciona solo en Chrome y versiones modernas de Opera.

	JavaScript	Descripción	Defecto
ellipse()	context.ellipse(X, Y, rX, rY, ar, ap, af, cR);	Dibuja una elipse **`X`** y **`Y`** son las coordenadas del centro, **`rX`** y **`rY`** representan el radio en x y el radio en y, **`ar`** representa el ángulo de rotación del eje horizontal (en radianes), **`ap`** es el ángulo de partida (en radianes), **`af`** es el ángulo final (en radianes), y **`cR`** en el sentido del reloj (**`false`**) o en sentido contrario (**`true`**)	

Dibujar una elipse con ellipse()

El método **`ellipse()`** tiene la siguiente sintaxis:

```
ctx.ellipse(centroX, centroY, radioX, radioY, rotacion, ap, af, cR);
```

Donde:
- **`centroX`** y **`centroY`** son las coordenadas del centro,
- **`radioX`** y **`radioY`** representan el radio en x y el radio en y,
- la **`rotacion`** representa el ángulo de rotación del eje horizontal (en radianes),
- **`ap`** es el ángulo de partida (en radianes),
- **`af`** es el ángulo final (en radianes), y
- **`cR`** (contra reloj) es la dirección.

Exactamente como en el caso de **`arc(),`** el parametro **`cR`** puede tomar dos valores:
- **`true`** (verdadero, o sea: en sentido contrario al de las agujas del reloj) y
- **`false`** (falso, o sea: en el sentido de las agujas del reloj).

Ejercicio 5: