

# Javascript

## Dessiner avec Canvas



# Canvas

Il s'agit d'un espace de pixels initialement transparents, armés de JavaScript pour réaliser un bon nombre de fonctions graphiques.

Ca permet de créer des dessins en 2D et en 3D directement dans le navigateur, du simple tracé aux animations.

# Initialisation HTML

```
<canvas id="mon_canvas" width="350"  
height="350">
```

Votre navigateur ne suppose pas Canvas :(

```
</canvas>
```

On initialise Canvas dans le HTML mais tout le reste se passera en Javascript !



# Accéder à l'élément **Canvas**

Pour ce faire, on va réutiliser ce qu'on a appris dans le cours sur le DOM...

**getElementById();**



# Définir le context **Canvas**

Cela sert à expliciter le contexte du dessin que nous allons utiliser. On précise à Javascript quelles fonctions il pourra utiliser.

**2D**

**3D**

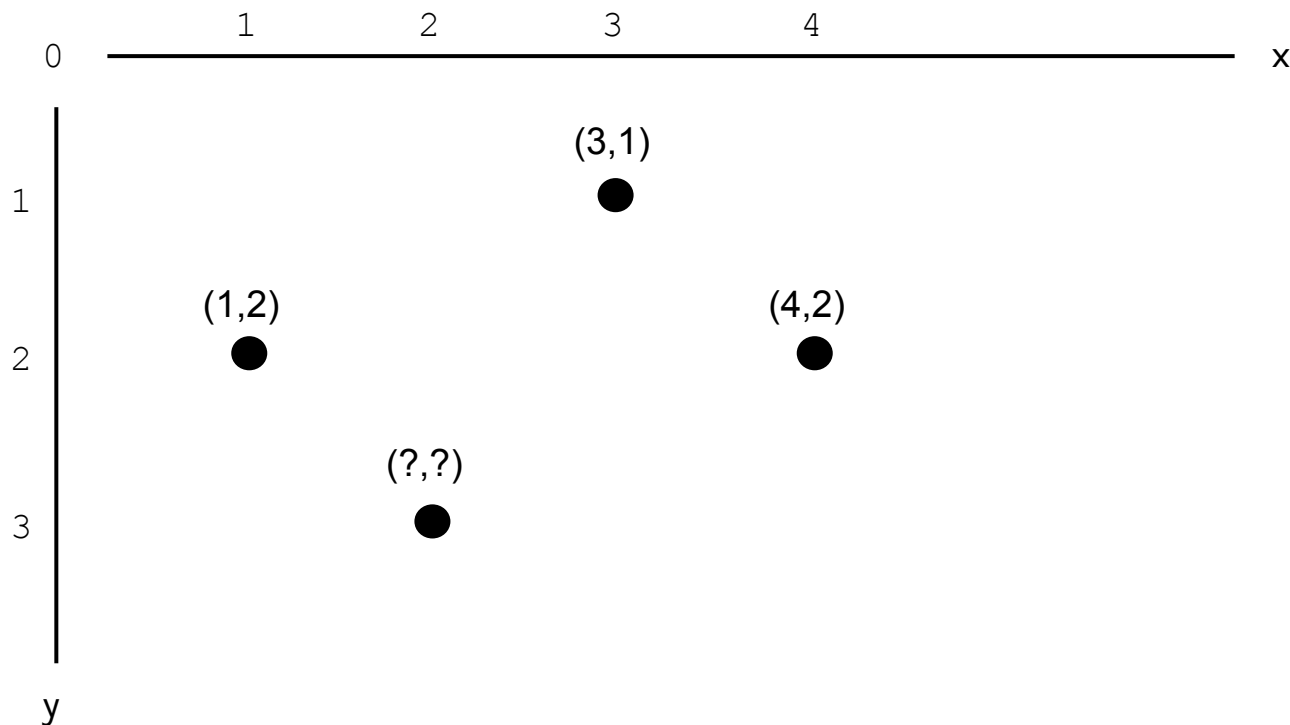


# Exemple

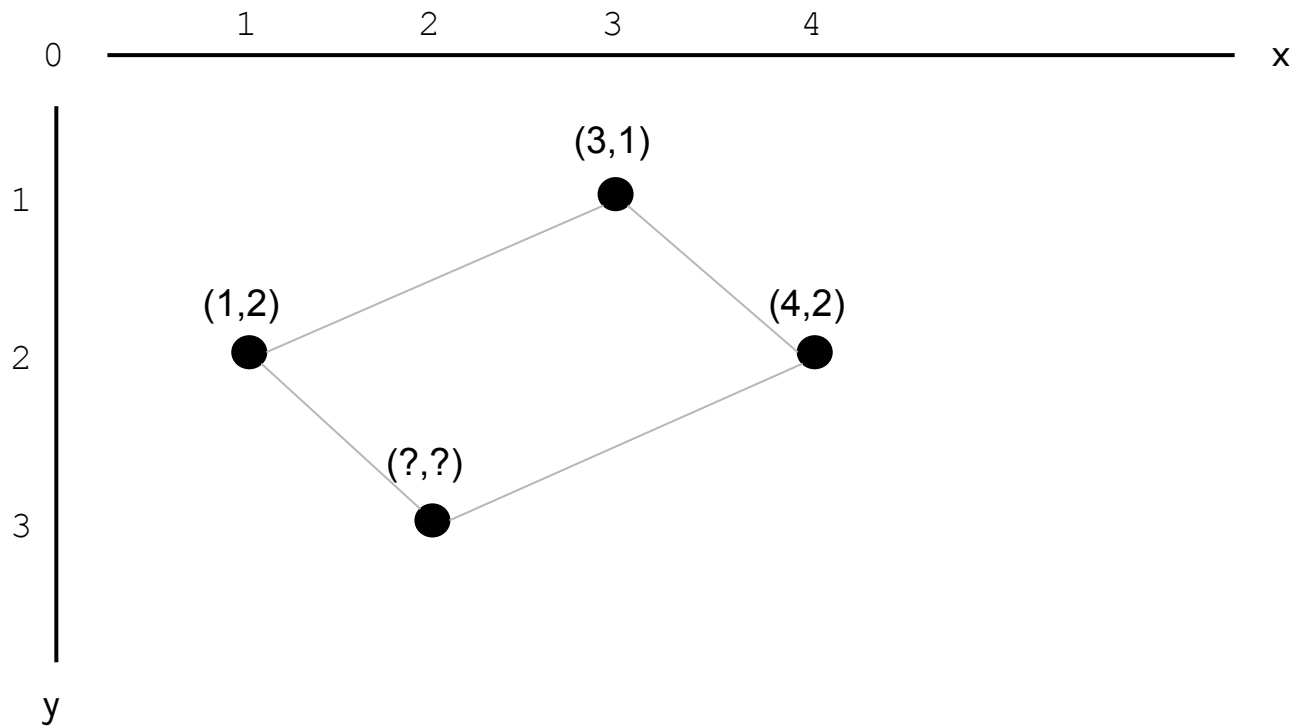
```
<script type="text/javascript">  
  var c =  
document.getElementById("mon_canvas");  
  var ctx = c.getContext("2d");  
  // ici, le reste du script  
</script>
```



# Le système de coordonnées



# Tracé en 2D





# Étapes d'un tracé

Initialisation - **beginPath()**;

Point de départ - **moveTo(x,y)**;

Point d'arrivée - **lineTo(x,y)**;

Clôture - **closePath()**;

Contour ou remplissage - **stroke()**; | **fill()**;

La forme n'apparaît qu'une fois qu'on a appelé  
**stroke()** (pour créer un contour)  
ou **fill()** (pour remplir)



# Exemple de tracé

```
var ctx = c.getContext("2d");  
ctx.beginPath();  
ctx.moveTo(50,50);  
ctx.lineTo(200,200);  
ctx.moveTo(200,50);  
ctx.lineTo(50,200);  
ctx.closePath();
```



# Les propriétés de style du contexte

## Couleurs

Tous les codes couleurs utilisés en CSS sont reconnus.  
Attention : on n'utilise pas de parenthèses, on applique un style !

```
ctx.fillStyle = "red";  
ctx.strokeStyle = "#ecf0f1";
```



# Les propriétés de style du contexte

## Style de lignes

Vous pouvez modifier la largeur des lignes et éditer les fins de lignes.

```
ctx.lineJoin = "bevel";  
ctx.lineCap = "round";
```



## lineJoin

round



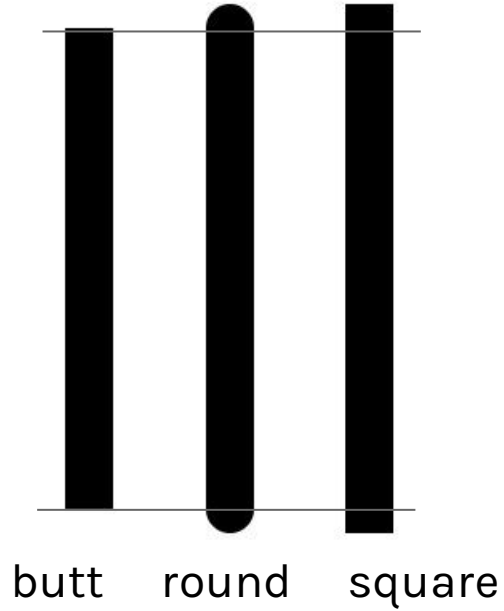
bevel



miter



## lineCap



# Les formes

## Rectangles et carrés

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



# Les formes

## Cercles et arcs de cercle

```
arc(x, y, rayon, startAngle, endAngle, sensAntihoraire)
```

coordonnées  
du centre

souvent = 0

en radiant,  
avec un  
Math.PI

true, false



# Les formes

## Cercles - exemples

```
ctx.arc(150,150,60,0,Math.PI,false);  
ctx.arc(180,130,15,0,Math.PI*2,false);
```



# Les formes

## Courbes de Bézier

```
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, destx, desty)
```

Coordonnées  
de départ


Point de  
passage

Coordonnées  
d'arrivée

# Les formes

## Courbes quadratiques

```
quadraticCurveTo(cp1x, cp1y, destx, desty)
```



Coordonnées  
de départ



Coordonnées  
d'arrivée

# CHALLENGE

Dessiner :

- Un drapeau
- Un bateau
- Une maison
- Un bonhomme

Bonus : refaire le logo d'android



## Quelques **ressources**...

- [CrunchZilla](#) !!!
- [Documentation canvas](#) chez Mozilla
- [Documentation canvas](#) chez w3schools
- [Tuto](#) sur Alsacreations
- Faire des [animations basiques](#)
- Faire des [animations avancées](#)

