

# Activité Canvas N° 1

## Présentation

La balise `<canvas>` est **élément HTML** qui peut être utilisé pour **dessiner et animer** des éléments graphiques à l'aide de scripts JavaScript notamment.

Cette activité présente les **bases** d'utilisation de l'**élément <canvas>** pour dessiner des **graphiques en 2D**.

L'élément `<canvas>` n'est **pas reconnu par les anciens navigateurs**. La taille par défaut de `<canvas>` est 300 px x 150 px (largeur x hauteur) mais des tailles personnalisées peuvent être définies à l'aide des propriétés HTML **height et width**.

Pour dessiner des graphiques sur un éléments `<canvas>`, il faut utiliser un objet de contexte JavaScript, qui crée des graphiques à la volée.

## Créer un canvas dans un page HTML

Voici un élément `<canvas>` avec un contenu de repli affichant le texte **Canvas non supporté par votre navigateur !**.

```
<canvas id="affichagefleche" width="150" height="150">
    Canvas non supporté par votre navigateur !
</canvas>
```

### Explications :

- L'attribut **id** permettra d'identifier de manière unique cet élément avec javascript,
- Les attributs **width** et **height** définissent la **largeur et la hauteur** du canvas,
- la balise fermante `</canvas>` est **obligatoire**.

## Le contexte de rendu

L'élément `<canvas>` crée une **surface vide** pour dessiner à grandeur fixe selon un ou plusieurs **contextes de rendu**. Ici nous utiliserons le **contexte de rendu 2D**.

Pour afficher quelque chose, le script javascript doit :

- **sélectionner** l'élément `<canvas>` grâce à son **id**
- et accéder au contexte de rendu 2d pour pouvoir dessiner dessus grâce à la méthode `getContext("2d")`.

```
<canvas id="mon_canvas" width="350" height="350">
    Canvas non supporté par votre navigateur !
</canvas>
<script>
var canvas = document.getElementById('mon_canvas');
var ctx = canvas.getContext('2d');
</script>
```

## Le système de coordonnées

Un canvas est un espace de pixels initialement transparents sur lequel vous pouvez dessiner des formes, mettre des images et les animer.

- Le point d'**origine (0,0)** est situé en haut à gauche,
- Pour les coordonnées (x,y) :
  - L'axe **horizontal** correspond à **x**,

- L'axe **vertical** à **y**.

Ces valeurs correspondent à la grille entourant les pixels, et non pas aux pixels eux-mêmes

## Lignes et tracés

Pour réaliser un tracé, il faut utiliser les méthodes suivantes :

- `beginPath()` pour l'initialiser
- `moveTo(x,y)` pour définir le début du tracé,
- `lineTo(x,y)` pour tracer la ligne elle-même ce qui va ajouter un segment au chemin commencé par `beginPath()`,
- ajouter autant de trajet que l'on souhaite.
- `closePath()` pour éventuellement **fermer** la forme et revenir automatiquement au point de départ.

```
ctx.beginPath();           // Début du chemin
ctx.moveTo(50,50);        // Le tracé part du point 50,50
ctx.lineTo(200,200);      // Un segment est ajouté vers 200,200
ctx.moveTo(200,50);       // Puis on saute jusqu'à 200,50
ctx.lineTo(50,200);       // Puis on trace jusqu'à 50,200
ctx.closePath();
```

Pour afficher le résultat il faut maintenant préciser les couleurs, le style de trait ou de remplissage :

- `stroke()` permet l'affichage du contour,
- `fill()` permet de remplir une forme

```
ctx.stroke();
```

Cela affiche le tracé avec les couleurs par défaut. <html>

</html> Pour modifier les styles par défaut :

- `fillStyle` en précisant la couleur de remplissage par son nom (red, black), par code hexadécimal (#f00, #000000), par code rgb, etc.
- `strokeStyle` en précisant la couleur du trait.
- `lineWidth` en précisant l'épaisseur du trait (1 pixel par défaut).

```
// Triangle
ctx.beginPath();           // Début du chemin
ctx.moveTo(150,80);        // Le tracé part du point 150,80
ctx.lineTo(300,230);      // Un segment est ajouté vers 300,230
ctx.lineTo(150,230);      // Un segment est ajouté vers 150,230
ctx.closePath();           // Fermeture du chemin
ctx.fillStyle = "lightblue"; // Définition de la couleur de remplissage
ctx.strokeStyle = "sienna"; // Définition de la couleur de contour
ctx.lineWidth = 5;
ctx.fill();                // Remplissage du dernier chemin tracé
ctx.stroke();               // Application du contour
```

<html>

</html>

## Dessiner des formes

Pour dessiner un Carré ou un rectangle on précise :

- sa hauteur, sa largeur, son point de départ pour les coordonnées `startx`, `starty`,
- en utilisant la fonction `fillRect()`

```
fillRect(startx, starty, hauteur, largeur)
```

## Effacer une portion de surface ou tout le canvas

`clearRect(startx, starty, hauteur, largeur)`

## Arc de cercle

Deux méthodes sont disponibles :

- `arcTo()`
- `arc()` qui définit les coordonnées centrales de l'arc, son rayon (toujours en pixels), l'angle de début et de fin, et enfin dans quel sens le pinceau va tourner grâce à un booléen.

```
arc( x, y, radius, startAngle, endAngle, sensAntiHoraire )
```

Les angles sont définis en radians avec `Math.PI` (un tour complet de cercle = `2*Math.PI`) et le sens de rotation est contraire aux aiguilles d'une montre lorsqu'il vaut `true`.

```
ctx.arc(150,150,100,0,Math.PI*2,true);
```

## les Images

Il faut utiliser :

- le constructeur `new Image()` pour initialiser une nouvelle image grâce au constructeur `new Image()` ;
- la propriété `src` (adresse de l'image) qui est indiquée permet au navigateur de charger l'image en arrière-plan
- l'événement `load` grâce à la propriété `onload` pour écrire une fonction de callback qui utilisera l'image lorsque celle-ci aura été totalement reçue. Cette phase est absolument nécessaire : les chargements sont asynchrones et il est impossible de savoir exactement quand une ressource sera prête, sous peine de ne rien voir s'afficher du tout.
- La méthode `drawImage(img,x,y)` copie l'image `img` chargée sur la surface de dessin, aux coordonnées `(x,y)`. Il est possible de se resserrer de la même ressource image une infinité de fois.

```
var image = new Image();
image.src = 'hat.jpg';
image.onload = function() {
  // Cette fonction est appelée lorsque l'image a été chargée
  ctx.drawImage(this,50,50); // this fait référence à l'objet courant (=image)
};
```

```
<html>
</html>
```

## Ressources Les premières balises utile pour le HTML

Voici deux liens parmi d'autres pour découvrir l'utilisation des Canvas en HTML 5 :

- <https://www.alsacreations.com/tuto/lire/1484-introduction.html>
- [https://developer.mozilla.org/fr/docs/Tutoriel\\_canvas](https://developer.mozilla.org/fr/docs/Tutoriel_canvas)

## Activité Les langages pour créer des sites Web ...

- [Les langages pour créer des sites Web](#)

From:  
[/- Les cours du BTS SIO](#)

Permanent link:  
[/doku.php/isn/canvas1](#)

Last update: **2019/01/31 20:34**

