JAVASCRIPT : TP2

Durée prévue : 3h.
Problématique : réalisation de pages web utilisant la balise <canvas>

Plan de l'étude	Remarque
Introduction	
I. Quelques exemples simples	
II. Première application : logo + texte	
III. Deuxième application : smiley LaMache	
IV. Pour aller plus loin	

Introduction

Canvas est une balise HTML qui va créer un cadre dans lequel nous allons pouvoir dessiner. Et c'est avec javascrit que nous allons pouvoir dessiner.

Pour la petite histoire, cette balise a été créée par Apple pour être utilisée sur son navigateur, Safari. Ensuite cette balise a été incorporée dans Firefox (Mozilla) et son utilisation c'est généralisé (aujourd'hui elle est intégrée dans HTML5).

I Quelques exemples simples

1. La structure

- Tout d'abord il faut créer la structure "canvas" (en gros un rectangle d'une certaine taille) :
  <canvas id="mon_canvas" width="400" height="250">
  là on a créé un rectangle "canvas" de hauteur (height) 250 pixels et de largeur 400 pixels et de nom "mon_cavas" (il doit toujours y avoir un nom!)

- ensuite il faut ajouter le script de javascript. Ce script doit récupérer l'élément Canvas (avec les différentes méthodes disponibles), créer ce que l'on appelle son "context" (aujourd'hui seul le 'context 2d' fonctionne parfaitement). Il ne reste plus qu'à utiliser les méthodes (fonctions) associées à ce context :
  var canvas1 = document.getElementById('mon_canvas'); // récupération du canvas
  var context1 = canv1.getContext('2d'); // récupération du context
- on remarquera que l'on peut créer plusieurs canvas (on choisira des nom différents pour pouvoir y accéder). Ces "canvas" pourront être positionnés dans la page à l'aide du CSS et du HTML.

2. Tracer des rectangles

a) ***Essayez le code html suivant (à copier dans un fichier et à enregistrer sous nom_du_fichier.html) et décrivez ce qu'il fait.***

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>la balise canvas</title>
  <style>body { background: black; } canvas { background: white; }</style>
 </head>

 <body>

  <canvas id="mon_canvas" width="400" height="250">
   <p>Désolé, votre navigateur ne supporte pas Canvas. Mettez-vous à jour</p>
  </canvas>
```

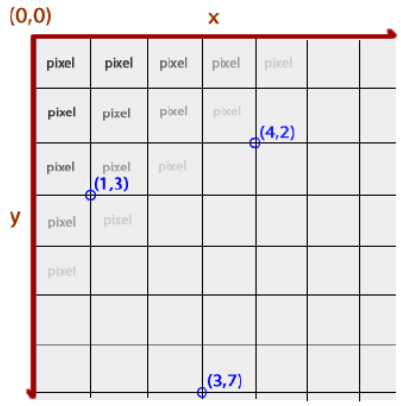
```
  <script>

      var canv1 = document.getElementById('mon_canvas');
      var context1 = canv1.getContext('2d');
      context1.fillStyle = "gold";
      context1.fillRect(0, 0, 50, 80);

  </script>
 </body>
</html>
```

Explications de la partie Javascript:

- la méthode xxxx.fillStyle permet de choisir la couleur. Ici la couleur est donnée à l'aide d'un nom prédéfini (il existe white, black, red, …) mais cela est un peu limité. On peut aussi utiliser les autres méthodes :
  - "rgba(rouge, vert, bleu, alpha)". Chaque valeur a un nombre entre 0 et 255. Alpha représente la transparence ou l'opacité (de 0 à 1 avec 0=transparent, 1=opaque, 0,5:semi-transparent, ….).
  - #xxxxxx : c'est le code hexadécimal de la couleur.
  - Pour récupérer le code de la couleur il y a des ressources internet disponibles ou plus simplement on peut utiliser le logiciel GIMP (il peut récupérer le code couleur dans une image, ...). GIMP fournit alors soit le code hexadécimal soit les codes RGB (Red, Green, Blue).
- la méthode xxxx.fillRect(, , ,) permet de tracer un rectangle : fillrect(x, y, largeur, hauteur) où x et y sont les coordonnées du point en haut à gauche

Positionnement :

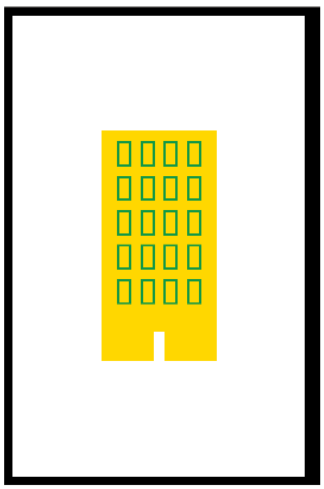
Le positionnement se fait en pixels, l'origine (0,0) se situant en haut à gauche du canvas :



D'autres méthodes :

- nom_du_context.strokeRect(x , y , l , h) : permet de tracer un rectangle non plein (juste le contour)
- nom_du_context.lineWidth = "x"; permet de choisir l'épaisseur du contour
- nom_du_context.clearRect(x, y, l, h) : permet d'effacer un rectangle d'un élément dessiné

Petit exercice : dessin d'un immeuble
On veut obtenir le résultat ci-contre :



données et contraintes:

- taille du canvas (partie blanche) : largeur 250px et hauteur 400px
- immeuble : largeur 100px et hauteur 200px
- l'immeuble est centré dans le canvas
- Essayer d'optimiser le code (nombre de lignes).

b) Quand tout fonctionne faites valider votre résultat par le professeur

3. Tracer des lignes

a) Essayez le code html suivant et décrivez ce qu'il fait.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>la balise canvas</title>
  <style>body { background: black; } canvas { background: white; }</style>
 </head>

 <body>

  <canvas id="mon_canvas" width="250" height="400">
   <p>Désolé, votre navigateur ne supporte pas Canvas. Mettez-vous à jour</p>
  </canvas>

  <script>
        var canv1 = document.getElementById('mon_canvas');
        var context1 = canv1.getContext('2d');
        context1.strokeStyle = "rgb(23, 145, 167)";
        context1.beginPath();
        context1.moveTo(20, 20);  // 1er point
        context1.lineTo(130, 20); // 2e point
        context1.lineTo(130, 50); // 3e
        context1.lineTo(75, 130); // 4e
        context1.lineTo(20, 50);  // 5e
        context1.closePath();     // On relie le 5e au 1er
        context1.stroke();
  </script>
 </body>
</html>
```

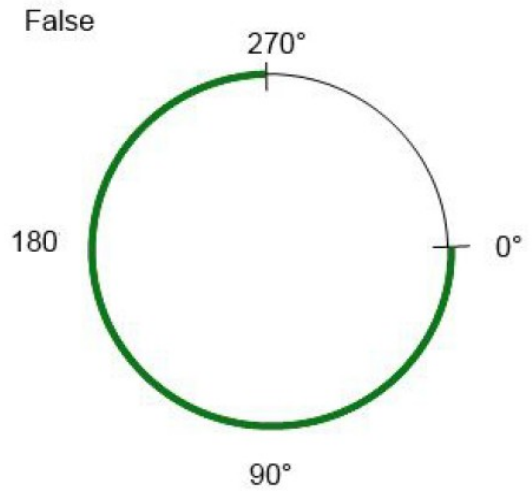
b) A quoi sert d'après vous la méthode "xxxx.moveTo(x,y)" ?
c) Expliquez comment fonctionne le code javascript précédent (utilisé pour le tracé)
d) Dans le code HTML précédent, remplacez xxxx.strokeStyle par xxxx.fillStyle et xxxx.stroke() par xxxx.fill(). Décrivez ce qui se passe

4. Tracer des cercles et des arcs de cercles

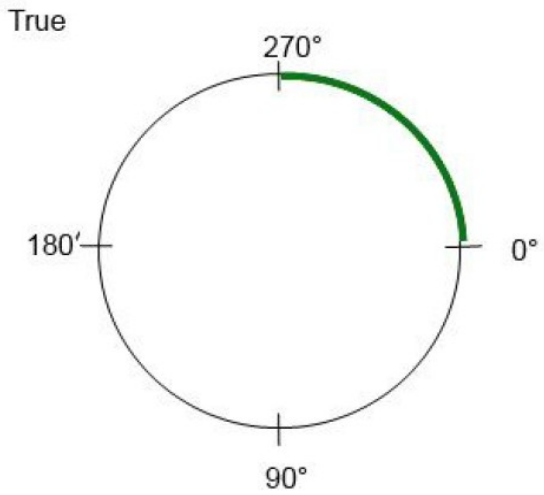
La méthode pour tracer des cercles (ou des arcs de cercles) est la suivante :
nom_du_context.arc(x, y, rayon, angleDepart, angleFin, sensInverse)

Par défaut sensInverse=false (on est donc dans le sens horaire par défaut)

Exemple de tracé d'un arc de cercle (en vert):
xxxx.arc (x , y , rayon, 0, 270) :



xxxx.arc (x , y , rayon, 0, 270, true)



Attention : nous vous avons donné l'exemple précédent en degrés pour faciliter la compréhension mais il faut donner les angles en radians.

Une remarque : la méthode "Math.PI" fournit la valeur de Pi (3,14….)

a) *Essayez le code html suivant et décrivez ce qu'il fait.*

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>la balise canvas</title>
  <style>body { background: black; } canvas { background: white; }</style>
 </head>

 <body>
```

```
<canvas id="mon_canvas" width="250" height="400">
  <p>Désolé, votre navigateur ne supporte pas Canvas. Mettez-vous à jour</p>
</canvas>

<script>
        var canv1 = document.getElementById('mon_canvas');
        var context1 = canv1.getContext('2d');
        context1.strokeStyle = "red";
        context1.fillStyle = "red";
        context1.beginPath(); // La bouche, un arc de cercle
        context1.arc(75, 75, 40, 0, Math.PI);
        context1.fill();

        context1.beginPath(); // Le cercle extérieur
        context1.arc(75, 75, 50, 0, Math.PI * 2);

        context1.moveTo(41, 58); // L'œil gauche
        context1.arc(55, 70, 20, (Math.PI / 180) * 220, (Math.PI / 180) * 320);

        context1.moveTo(81, 58); // L'œil droit
        context1.arc(95, 70, 20, (Math.PI / 180) * 220, (Math.PI / 180) * 320);
        context1.stroke();
  </script>
 </body>
</html>
```

b) Pourquoi y-a-t-il 2 méthodes (2 lignes de code) utilisées pour choisir la couleur rouge ?
c) Expliquez de manière détaillée, comment le code javascript s'y prend pour faire le dessin
d) Retrouvez, en degrés, les 2 angles (départ et arrivé) des arcs utilisés pour dessiner les yeux.
e) Que se passe-t-il si l'on oublie de faire les deux "context1.moveTo" du code (vous pouvez faire des essais!)?

5. Insérer une image

Il est possible d'insérer des images au sein d'un canvas. Pour ce faire, on utilisera la méthode drawImage(image, x, y), mais attention : pour qu'une image puisse être utilisée, elle doit au préalable être accessible via un objet Image ou un élément <img />. Il est également possible d'insérer un canvas dans un canvas ! En effet, le canvas que l'on va insérer est considéré comme une image.

La méthode pour insérer des images est la suivante :

non_du_context.drawImage(image, sx, sy, sLargeur, sHauteur, dx, dy, dLargeur, dHauteur)

avec : sx et sy coordonnées de l'image (du point en haut à gauche)
sLargeur et sHauteur : la taille de l'image
dx, dy, dLargeur et dHauteur permettent de recadrer l'image (option)

Remarque : il faut faire attention à l'endroit où se trouve l'image (répertoire, ...).

a) ***Récupérez l'image "rostand.jpg", mettez là dans le même répertoire que vos fichiers HTML et essayez le code html suivant. Décrivez ce qu'il fait.***

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>la balise canvas</title>
<style>body { background: black; } canvas { background: white; }</style>
</head>
<body>
<canvas id="mon_canvas" width="250" height="140">
<p>Désolé, votre navigateur ne supporte pas Canvas. Mettez-vous à jour</p>
</canvas>
<script>
var canv1 = document.getElementById('mon_canvas');
var context1 = canv1.getContext('2d');
var monimage = new Image();
monimage.src = 'rostand.jpg';
monimage.onload = function() {
context1.drawImage(monimage, 135, 35, 99, 86);
}
context1.strokeStyle = "red";
context1.fillStyle = "red";
context1.beginPath(); // La bouche, un arc de cercle
context1.arc(75, 75, 40, 0, Math.PI );
context1.fill();
context1.beginPath(); // Le cercle extérieur
context1.arc(75, 75, 50, 0, Math.PI * 2);
context1.moveTo(41, 58); // L'oeil gauche
context1.arc(55, 70, 20, (Math.PI / 180) * 220, (Math.PI / 180) * 320);
context1.moveTo(81, 58); // L'oeil droit
context1.arc(95, 70, 20, (Math.PI / 180) * 220, (Math.PI / 180) * 320);
context1.stroke();
</script>
</body>
</html>
```

6. Écrire du texte

Pour écrire du texte au sein d'un canvas, il y a les méthodes fillText() et strokeText(), ainsi que la propriété
font (qui permet de définir le style du texte)

a) Essayez le code html suivant et décrivez ce qu'il fait.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>la balise canvas</title>
<style>body { background: black; } canvas { background: white; }</style>
</head>
<body>
<canvas id="mon_canvas" width="150" height="150">
<p>Désolé, votre navigateur ne supporte pas Canvas. Mettez-vous à jour</p>
</canvas>
<script>
var canv1 = document.getElementById('mon_canvas');
var context1 = canv1.getContext('2d');
context1.fillStyle = "rgba(23, 145, 167, 1)";
context1.fillRect(50, 50, 50, 50);
context1.font = "bold 22pt Calibri,Geneva,Arial";
context1.fillStyle = "#ffffff";
context1.fillText("js", 78, 92);
</script>
</body>
</html>
```

b) Modifiez le code afin d'afficher le texte en rouge, au centre du rectangle et en plus gros.

II. Première application : logo + texte

Cahier des charges :

- On dispose de l'image correspondant au logo du lycée (rostand.jpg).
- On va y rajouter un texte affichant la section (à l'aide d'une variable "section" qui vaudra soit "CIM" soit "SNIR".
- La taille du canvas sera de 400 × 220 pixels.

On veut obtenir le résultat suivant

si section='SNIR' :

si section='CIM'

Faites le code HTML avec sa balise "canvas" et son script Javascript permettant de répondre au cahier des charges. Quand tout fonctionne faites valider votre résultat par le professeur.

III. Deuxième application : smiley LaMache

Nous disposons de l'image suivante :

et nous voulons créer le smiley laMache suivant :

Cahier des charges :

- le canvas aura pour taille 150 × 150 pixels
- le fichier ***smiley.html*** sera le suivant (vous n'avez pas à modifier ce code):

```
<!DOCTYPE html>
<html>
  <head>
    <!-- En-tête de la page -->
    <meta charset="utf-8" />
    <script type="text/javascript" src="smiley.js"></script>
    <title>Smiley</title>
  </head>
```

```
<body>
  <!-- Corps de la page -->
  <canvas id="mon_canevas1" width="150" height="150">
                [No canvas support]
          </canvas>
</body>
</html>
```

- le dessin sera réalisé dans fichier ***smiley.js*** :

```
window.onload = function ()
{

    Votre code

}
```

Complétez le code du fichier "smiley.js" et quand tout fonctionne faites valider votre résultat par le professeur.

IV. Pour aller plus loin

Bien évidemment il existe d'autres méthodes associées à la balise canvas et à javascrit.

On peut ainsi modifier les images, faire des dégradés, créer des ombres, faire des animations ….

Un exemple d'animation que vous pouvez essayer:

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>la balise canvas</title>
  <style>body { background: black; } canvas { background: white; }</style>
 </head>

 <body>

  <canvas id="mon_canvas" width="150" height="150">
   <p>Désolé, votre navigateur ne supporte pas Canvas. Mettez-vous à jour</p>
  </canvas>
```

```
<script>
    var canv1 = document.getElementById('mon_canvas');
    var context1 = canv1.getContext('2d');

    var dernierX = canv1.width * Math.random();
    var dernierY = canv1.height * Math.random();
    var hue = 0;   // couleur
    function animate() {
      context1.save();
      context1.lineCap = "round";
      context1.beginPath();
      context1.lineWidth = 5 + Math.random() * 10;
      context1.moveTo(dernierX, dernierY);
      dernierX = canv1.width * Math.random(); // prochain point
      dernierY = canv1.height * Math.random();
      context1.bezierCurveTo(canv1.width * Math.random(),  canv1.height * Math.random(),
      canv1.width * Math.random(), canv1.height * Math.random(),  dernierX, dernierY);
      hue = hue + 10 * Math.random();  // modification de la couleur
      context1.strokeStyle = "hsl("+ hue +",50%,50%)";
      context1.shadowColor = "white";
      context1.shadowBlur = 10;
      context1.stroke();
      context1.restore();
    }
    function assombrir() {
      context1.fillStyle = "rgba(0,0,0,0.1)";
      context1.fillRect(0, 0, canv1.width, canv1.height);
    }
    setInterval(animate, 100);
    setInterval(assombrir, 40);


  </script>
 </body>
</html>
```

Pour compléter ce TP :

- sur le site "openclass Room" il y a une formation javascript avec une rubrique sur <canvas> (https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript)
- parmi les multiples ressources disponibles sur internet nous vous recommandons : http://www.installations-electriques.net/Logiq/canvas/Canvas.htm