Leçon 6-1 – HTML5 : Dessin

s1 ----------

Dans cette partie, nous allons étudier la balise canvas qui permet de dessiner dynamiquement (c'est-à-dire à l'aide de JavaScript) des formes et des images. L'enjeu est ici de bien comprendre la notion de contexte associé à un canvas, notion qui n'est pas forcément simple. Mais plusieurs exercices vont vous y aider.

s2 ----------

Canvas est ce qu'on appelle un conteneur pour des objets créés dynamiquement. Quels sont ces objets ? Des courbes, des formes, des images, des animations. Dynamiquement, signifie ici que ces sont créés au moment de l'affichage de la page web par des instructions JavaScript.

L'utilisation de canvas offre plusieurs possibilités :

- Tracer des courbes, des formes
- Transformer des images
- Développer des animations interactives

et constitue une vraie alternative au langage Flash.

s3 ----------

Commençons par un exemple simple qui va nous permettre de comprendre le principe de base de la balise canvas. Je vous propose de reproduire dans un fichier HTML le code affiché ici. En voici les points-clés.

- D'abord la balise canvas avec l'identifiant "dessin" qui nous permettra de la cibler par du code JavaScript ainsi que des dimensions ; ces trois attributs sont obligatoires.
- Ensuite le code JavaScript qui va créer des éléments à l'intérieur de canvas ; dans ce code
    - on crée une variable JavaScript monCanvas que l'on associe au canvas
    - à cette variable, par la méthode getContex("2d"), on associe un objet HTML5 qui possède de nombreuses méthodes et propriétés de dessin ; on appelle cet objet le contexte de dessin (nous préciserons plus loin cette notion)
    - enfin, sur ce contexte, on utilise des méthodes de dessin, ici les méthodes fillStyle (choix d'une couleur de remplissage) et fillRect (dessin d'un rectangle)
- dernier point : il est conseillé d'effectuer un test permettant de savoir si l'élément canvas est supporté par le navigateur : ainsi si la méthode getContext renvoie false, on affiche une alerte

Une explication s'impose quant à la notion de contexte. Le contexte définit la façon dont les objets vont être dessinés ; il est sans dimension, mais, pour les besoins de l'explication, ici et dans les planches qui suivent, le contexte peut être vu comme un calque représenté ici en bleu clair dont le coin supérieur gauche se positionne par défaut en x=0 et y=0 c'est-à-dire dans le coin supérieur gauche du canvas représenté ici en gris. x est mesuré en pixels depuis la gauche vers la droite, y est mesuré en pixels depuis le haut vers le bas.

Tout se passe comme si méthodes précédentes fillStyle et fillRect venaient dessiner dans ce calque.

Maintenant que nous avons compris le fonctionnement de base, nous allons introduire d'autres méthodes de dessin.

s4 ----------

Dans l'exemple suivant, nous allons tracer un triangle, segment par segment.

Quelques explications sur ce code : une fois que le canvas est créé,

- on définit grâce à la méthode strokeStyle la couleur qui va plus tard être utilisée pour un remplissage ;
- on définit grâce à la méthode lineWidth l'épaisseur des traits à venir ;
- on précise au canvas que l'on souhaite tracer un contour grâce à la méthode beginPath ;
- ensuite grâce à la méthode moveTo, on déplace l'origine d'un tracé dans le canvas (un peu comme si on déplaçait la pointe du crayon au-dessus de la feuille) ;
- une série de lineTo permet de tracer un trait depuis la dernière position de tracé jusqu'à un nouveau point ;
- enfin, on ferme le tracé ; on applique le remplissage et la taille des traits.

Je vous propose de reproduire cet exemple, puis dans l'ordre de changer la couleur de remplissage, la taille du trait, les coordonnées des points.

Enfin, si ça vous intéresse vous trouverez sur Internet comment tracer des arcs (et donc des cercles) ou même des courbes de Bézier.

s5 ----------

Dans l'exemple suivant, nous allons créer un dégradé linéaire dans un rectangle, et ce de façon très similaire au premier exemple présenté dans cette leçon, sauf qu'ici l'argument de la méthode fillStyle sera un dégradé et non une couleur unique.

Quelques explications sur ce code : une fois que le canvas est créé,

- on associe au contexte un gradient linéaire et on l'associe à la variable lingrad ; ce se fait grâce à la méthode createLinearGradient ; les arguments de cette méthode sont les coordonnées des points de début et de fin du dégradé : ici le point (0,0) et le point (0,150) ;
- on ajoute trois couleurs grâce à la méthode addColorStop où le premier paramètre prend une valeur entre 0 et 1 définissant sa position dans le dégradé complet ;
- ensuite, on utilise la méthode fillStyle que l'on avait utilisée tout à l'heure, mais cette fois avec pour argument la variable lingrad ;
- enfin, on crée le rectangle grâce à la méthode fillRect.

Je vous propose de reproduire cet exemple, puis dans l'ordre de changer le nombre de couleurs du dégradé et leur valeur.

s6 ----------

Des transformations peuvent être appliquées au contexte de dessin. Parmi celles-ci, un changement d'échelle, une rotation, une translation.

Revenons au premier exercice. Appliquons une rotation au contexte juste après qu'il ait été créé et avant même d'y dessiner les deux rectangles (comme sur l'imagette présentée ici). Testons-le ! Sans doute allez-vous obtenir la figure suivante.

Une explication s'impose. Nous avons vu que le contexte (représenté en bleu dans la figure) définit à chaque instant la façon dont les nouveaux objets vont être dessinés dans le canvas (représenté en gris).

Première remarque : par défaut le contexte se positionne en 0,0 c'est-à-dire dans le coin supérieur gauche du canvas, mais on peut déplacer ce contexte, le faire tourner et lui appliquer un changement d'échelle. Chaque changement apporté au contexte affectera les nouveaux objets qui seront dessinés. Pas les objets déjà dessinés ! À chaque changement du contexte, c'est comme si on ajoutait une feuille de calque sur un dessin existant.

Deuxième remarque : les objets dessinés par le contexte peuvent se retrouver hors du cadre défini par le canvas. Dans notre exemple, le contexte est positionné dans le coin supérieur gauche du canvas. Une rotation du contexte autour de ce coin supérieur gauche produit le cadre pointillé. Le fait que le rectangle rouge soit tronqué par cette rotation ne nous surprend donc pas.

Troisième remarque : si l'on veut à présent que le rectangle rouge apparaisse en totalité, il faut avant la rotation - et toujours avant de dessiner quoique ce soit - appliquer au contexte une translation, par exemple selon l'axe x. Ceci produit le second cadre en pointillés. Bien entendu, l'ordre des opérations est important (vous pouvez essayer d'appliquer une rotation puis une translation).

Quatrième remarque qui découle des remarques précédentes : si on veut qu'un objet, par exemple un rectangle, ne soit dessiné avec une rotation non plus autour de son coin supérieur gauche comme c'est le cas dans l'exemple présenté ici, mais autour de son centre, il faut le dessiner dans le contexte de telle façon que ce centre coïncide avec le coin supérieur gauche du contexte, comme sur la figure (rect supplémentaire bleu sur le contexte translation et rotation).

Ne pas oublier donc ce point essentiel : avant de dessiner un objet, penser en fonction du rendu souhaité à bien identifier les rotations, translations et changement d'échelle à appliquer au contexte.

s7 ----------

Pour bien comprendre cette notion de changement de contexte, je vous propose de mettre en oeuvre ce concept de changement de contexte en dessinant deux rectangles : le premier sans rotation, le second avec rotation comme sur la figure.

s8 ----------

Dans cet exercice je vous propose de créer des éléments textuels à l'intérieur d'un contexte.

Quelques explications sur ce code : une fois que le canvas est créé,

- on associe au contexte un style de remplissage a priori grâce à la méthode fillStyle ;
- on associe au contexte les fontes à utiliser grâce à la méthode font ;
- on précise comment le texte est aligné verticalement à l'intérieur de l'espace prévu ;
- enfin, on crée deux textes : le premier grâce à méthode fillText dont les arguments sont la

chaîne de caractères à afficher et la position du coin supérieur gauche du cadre contenant ce texte, le second grâce à la méthode strokeText qui produit juste un contour des lettres.

Je vous propose de reproduire cet exemple, puis dans l'ordre de changer l'argument de baseLine, la localisation des textes et les chaînes à afficher.

s9 ----------

Dans ce dernier exercice, je vous propose d'importer une image à l'intérieur d'un contexte.

Quelques explications sur ce code : une fois que le canvas est créé,

- on définit un nouvel objet image, objet auquel sont associés des méthodes et des attributs que l'on va découvrir ;
- ensuite, comme on doit charger une image, on veut être sûr qu'elle est bien présente au moment de l'afficher ; pour ce faire, on se sert de l'attribut onload : ainsi lorsque l'image sera chargée, on l'associera au contexte grâce à la méthode drawImage dont les arguments sont l'objet image précédent et la position du coin supérieur gauche de l'image dans le contexte. drawImage possède d'autres arguments optionnels comme la largeur et la hauteur de l'image.

Je vous propose de reproduire cet exemple, puis avant affichage de modifier le contexte en lui appliquant une rotation par exemple.

s10 ----------

L'étude des méthodes qui permettent de manipuler les pixels d'une image, les ombrages et la transparence des objets sort du cadre de ce module. Je suis sûr que, si ce domaine vous intéresse, vous trouverez par vous-même ces éléments sur Internet.

Bon travail !

s11 ----------

Dans cette partie nous avons donc vu comment dessiner à l'intérieur d'un canzas. Dans la partie suivante, nous allons découvrir comment créer l'illusion du mouvement en faisant évoluer ce dessin dans le temps.

À tout de suite.