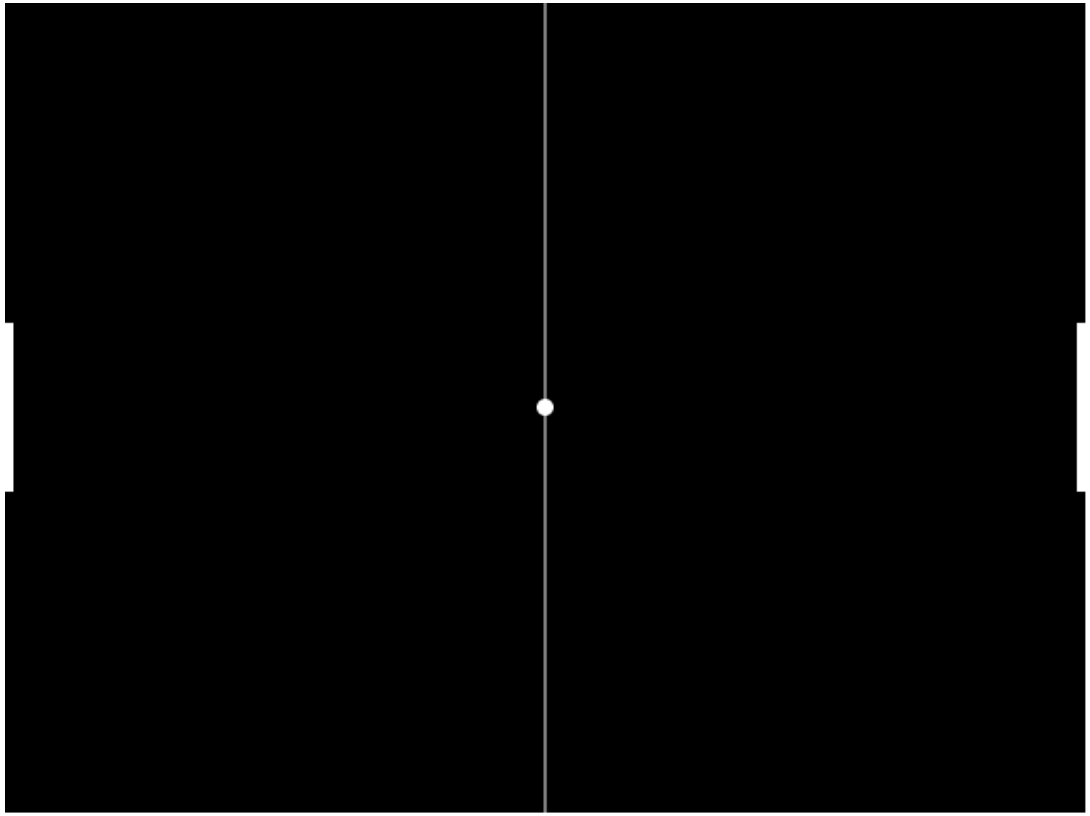
CRÉER UN PONG EN JAVASCRIPT



Bonjour, aujourd'hui nous allons voir un grand classique de programmation : la création d'un jeu Pong 🙂 . Le jeu se lancera dans un navigateur et sera développé en JavaScript (avec un soupçon de HTML/CSS pour l'interface).

Vous pouvez tester le jeu à cette adresse : https://devoreve.github.io/pong/ (je sais, le design est

COMMENTAIRES RÉCENTS

GAPA DANS FORMULAIRE DE CONTACT AVEC AJAX

CATÉGORIES

TUTO

ARCHIVES

JUILLET 2018

JUIN 2018

Fièrement propulsé par WordPress
Thème Writr par WordPress.com.

terrible mais je vous autorise à le reprendre si vous le trouvez si irrésistible 😉).

De manière à le rendre accessible au plus de monde possible, j'ai limité le nombre de bibliothèques externes au maximum. Pour les mêmes raisons, je ne parlerai pas de programmation orientée objet ni de compatibilité navigateurs et je n'utiliserai pas de notions mathématiques complexes (pour les calculs des trajectoires par exemple) mais je n'exclus pas l'idée de refaire un article avec une version plus optimisée par la suite ;).

J'ai essayé de rendre le code le plus simple possible mais il faudra quand même avoir les bases en JavaScript et en algorithmique pour bien comprendre 😉 . Sur ce, codons !

Étape 1 : L'interface graphique

La structure de la page

Elle sera minimaliste pour l'instant pour qu'on se concentre plutôt sur le code JavaScript mais ne vous inquiétez pas, nous rajouterons un peu plus loin un panel pour l'affichage du score et des boutons pour la gestion de la partie.

```
01.  <!DOCTYPE HTML>
02.  <html>
```

```
<head>
    <meta charset="utf-8">
    <title>Rétro game - Pong</title>
</head>
<body>
<h1>Pong</h1>
<main>
    <canvas id="canvas" width="640" height="480">
</canvas>
</main>
<script src="js/main.js"></script>
</body>
</html>
```

La ligne importante ici c'est : `<canvas id="canvas" width="640" height="480"></canvas>`

En effet pour notre jeu nous allons utiliser l'élément canvas apparu avec html 5 qui va nous permettre de dessiner le plateau et autres éléments du jeu.

Si vous ne connaissez pas cet élément je vous invite à lire ce lien : https://www.alsacreations.com/tuto/lire/1484-introduction.html

Pour l'instant il n'y a pas grand-chose sur notre page, passons donc à la zone de jeu à proprement parler.

La zone de jeu

```
'use strict';
```

```
02.
03.  var canvas;
04.
05.  function draw() {
06.      var context = canvas.getContext('2d');
07.
08.      // Draw field
09.      context.fillStyle = 'black';
10.      context.fillRect(0, 0, canvas.width, canvas.height);
11.
12.      // Draw middle line
13.      context.strokeStyle = 'white';
14.      context.beginPath();
15.      context.moveTo(canvas.width / 2, 0);
16.      context.lineTo(canvas.width / 2, canvas.height);
17.      context.stroke();
18.  }
19.
20.  document.addEventListener('DOMContentLoaded', function
     () {
21.      canvas = document.getElementById('canvas');
22.      draw();
23.  });
```

Rien d'extraordinaire jusque là : on récupère l'élément canvas de notre fichier html et on dessine un rectangle noir et une ligne blanche en plein milieu, tout cela dans une fonction. Notez que l'on n'utilise pas des valeurs au hasard pour définir la zone de jeu mais la largeur et la hauteur du canvas, coomme ça notre zone s'adapte en fonction de la taille de notre canvas.

Les joueurs et la balle

Commençons par rajouter quelques données : une variable représentant le jeu (qui contiendra les joueurs et la balle) et des constantes avec les tailles des joueurs.

```
01.  'use strict';
02.
03.  var canvas;
04.  var game;
05.
06.  const PLAYER_HEIGHT = 100;
07.  const PLAYER_WIDTH = 5;
```

Maintenant initialisons notre nouvelle variable game :

```
01.  document.addEventListener('DOMContentLoaded', function
     () {
02.      canvas = document.getElementById('canvas');
03.      game = {
04.          player: {
05.              y: canvas.height / 2 - PLAYER_HEIGHT / 2
06.          },
07.          computer: {
08.              y: canvas.height / 2 - PLAYER_HEIGHT / 2
09.          },
10.          ball: {
11.              x: canvas.width / 2,
12.              y: canvas.height / 2,
13.              r: 5
14.          }
15.      }
```

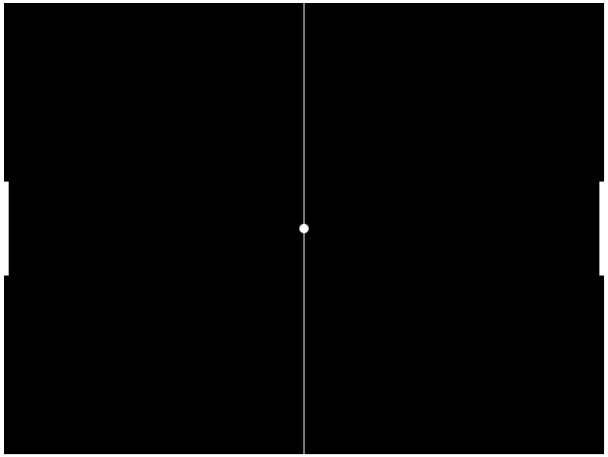
```
16.
17.         draw();
18.     });
```

Comme pour la zone de jeu, j'utilise les propriétés width et height du canvas pour définir la position des joueurs et de la balle. Cela va nous permettre de bien centrer les éléments. Notez également que l'on retranche à cette position la moitié de la hauteur. Il ne reste plus qu'à dessiner.

Nous allons rajouter ces lignes dans notre fonction draw :

```
01.     // Draw players
02.     context.fillStyle = 'white';
03.     context.fillRect(0, game.player.y, PLAYER_WIDTH,
        PLAYER_HEIGHT);
04.     context.fillRect(canvas.width - PLAYER_WIDTH,
        game.computer.y, PLAYER_WIDTH, PLAYER_HEIGHT);
05.
06.     // Draw ball
07.     context.beginPath();
08.     context.fillStyle = 'white';
09.     context.arc(game.ball.x, game.ball.y, game.ball.r, 0,
        Math.PI * 2, false);
10.     context.fill();
```

Ce dernier code conclut notre première étape. Voici le résultat :



Étape 2 : L'animation

Maintenant que nous avons notre interface, nous allons lancer l'animation du jeu : déplacer la balle de gauche à droite et les joueurs du haut vers le bas. Commençons par le déplacement de la balle.

Le mouvement de la balle

Pour que la balle se déplace, il faut modifier sa position sur le canvas. Par exemple :

```
01.   game.ball.x += 2;
02.   game.ball.y += 2;
```

Mais si vous testez ce code, vous allez constater qu'il ne se passe rien. En effet nous avons bien modifié la position de la balle mais il faut également redessiner le canvas pour que la balle et les joueurs soient réactualisés :

```
01.  game.ball.x += 2;
02.  game.ball.y += 2;
03.  draw();
```

Si vous testez, vous allez constater que notre balle s'est bien déplacée. Il ne reste plus qu'à lancer l'animation. Pour cela nous allons mettre ce code dans une fonction et l'appeler 60 fois par seconde avec requestAnimationFrame. En outre pour travailler facilement sur la vitesse de la balle, nous allons la stocker comme une propriété de la balle. Bien codons tout ça.

Déjà rajoutons notre propriété dans notre objet balle :

```
01.  game = {
02.          player: {
03.              y: canvas.height / 2 - PLAYER_HEIGHT / 2
04.          },
05.          computer: {
06.              y: canvas.height / 2 - PLAYER_HEIGHT / 2
07.          },
08.          ball: {
09.              x: canvas.width / 2,
10.              y: canvas.height / 2,
11.              r: 5,
12.              speed: {
13.                  x: 2,
14.                  y: 2
15.              }
16.          }
17.      }
```

Et maintenant notre fonction play :

```
01.  function play() {
02.      game.ball.x += game.ball.speed.x;
03.      game.ball.y += game.ball.speed.y;
04.      draw();
05.
06.      requestAnimationFrame(play);
07.  }
```

Bien sûr il faut que nous l'appelions une première fois pour que l'animation se lance. Nous allons faire ça au démarrage de la page, juste après avoir dessiné la zone de jeu.

```
01.  draw();
02.  play();
```

Voilà après ça notre balle se déplace bien dans la zone de jeu !

Nous n'avons pas encore géré le cas où la balle rencontre le bord du terrain ou un joueur donc la balle continue son chemin une fois qu'elle arrive à la limite du canvas et... disparaît. Il nous faudra gérer les collisions pour indiquer à la balle de repartir dans l'autre sens mais, avant cela, faisons se déplacer les joueurs.

Le déplacement des joueurs

Commençons par le joueur que l'on contrôle. Nous allons mettre en place un événement au niveau du canvas lorsque la souris bouge.

```
01.  // Mouse move event
02.  canvas.addEventListener('mousemove', playerMove);
```

La fonction playerMove sera appelée dès que l'on déplacera la souris dans le canvas. Faisons en sorte que notre joueur suive le mouvement de la souris.

```
01.  function playerMove(event) {
02.      // Get the mouse location in the canvas
03.      var canvasLocation = canvas.getBoundingClientRect();
04.      var mouseLocation = event.clientY -
     canvasLocation.y;
05.
06.      game.player.y = mouseLocation - PLAYER_HEIGHT / 2;
07.  }
```

On récupère la position de la souris relative au canvas et on modifie la position sur l'axe y (c'est-à-dire la position verticale) du joueur. Vous constaterez que l'on retranche la moitié de la hauteur du joueur, ceci afin que l'on contrôle le joueur à partir du milieu (encore une fois n'hésitez pas à tester avec et sans pour voir la différence).

Bon maintenant nous arrivons bien à bouger le joueur mais si nous allons jusqu'au bout de la zone de jeu… le joueur en sort. Or, tels des gladiateurs, aucun joueur n'a le

droit de sortir de l'arène tant que le comb... euh match n'est pas terminé. Nous gérerons ça dans la partie collisions. Passons maintenant aux mouvements de l'ordinateur.

```
01.  function computerMove() {
02.      game.computer.y += game.ball.speed.y * 0.85;
03.  }
```

Rien de très poussé, on fait en sorte que le joueur suive la balle et on fait déplacer l'ordinateur à la vitesse de la balle (un peu moins vite pour laisser une chance au joueur 😉).

N'oublions pas d'appeler cette fonction dans notre fonction play :

```
01.  draw();
02.
03.  computerMove();
```

Bien occupons-nous maintenant des collisions.

Étape 3 : Les collisions

Les joueurs

Empêchons d'ores et déjà les joueurs de sortir du terrain (ouais on est comme ça). Rajoutons ces conditions dans notre fonction playerMove.

```
if (mouseLocation < PLAYER_HEIGHT / 2) {
    game.player.y = 0;
} else if (mouseLocation > canvas.height - PLAYER_HEIGHT / 2) {
    game.player.y = canvas.height - PLAYER_HEIGHT;
} else {
    game.player.y = mouseLocation - PLAYER_HEIGHT / 2;
}
```

La balle

Commençons par faire une fonction qui va gérer le déplacement de la balle (à la base dans la fonction play).

```
function ballMove() {
    game.ball.x += game.ball.speed.x;
    game.ball.y += game.ball.speed.y;
}
```

Modifions notre fonction play pour incorporer le code ci-dessus.

```
function play() {
    draw();

    computerMove();
    ballMove();

    requestAnimationFrame(play);
}
```

À présent faisons en sorte que la balle rebondisse lorsqu'elle touche le haut et le bas du terrain.

```
01.  function ballMove() {
02.      // Rebounds on top and bottom
03.      if (game.ball.y > canvas.height || game.ball.y < 0)
     {
04.          game.ball.speed.y *= -1;
05.      }
06.
07.      game.ball.x += game.ball.speed.x;
08.      game.ball.y += game.ball.speed.y;
09.  }
```

On inverse la vitesse sur l'axe y de la balle lorsqu'elle touche les limites haute et basse du terrain. Et maintenant le rebond lorsque les joueurs touchent la balle.

```
01.  if (game.ball.x > canvas.width - PLAYER_WIDTH) {
02.      collide(game.computer);
03.  } else if (game.ball.x < PLAYER_WIDTH) {
04.      collide(game.player);
05.  }
```

On commence par regarder quel joueur doit taper dans la balle puis on vérifie si le joueur a réussi à frapper la balle.

```
01.  function collide(player) {
02.      // The player does not hit the ball
```

```
03.     if (game.ball.y < player.y || game.ball.y > player.y
    + PLAYER_HEIGHT) {
04.         // Set ball and players to the center
05.         game.ball.x = canvas.width / 2;
06.         game.ball.y = canvas.height / 2;
07.         game.player.y = canvas.height / 2 -
    PLAYER_HEIGHT / 2;
08.         game.computer.y = canvas.height / 2 -
    PLAYER_HEIGHT / 2;
09.
10.         // Reset speed
11.         game.ball.speed.x = 2;
12.     } else {
13.         // Increase speed and change direction
14.         game.ball.speed.x *= -1.2;
15.     }
16. }
```

Si le joueur tape la balle, on inverse la vitesse sur l'axe x en multipliant par -1 (j'ai multiplié par -1.2 pour augmenter légèrement la vitesse de la balle). Si jamais il la rate, on réinitialise tous les éléments.

Trajectoire

Notre jeu est maintenant fonctionnel mais rendons la trajectoire de la balle un peu plus aléatoire.

```
01. function changeDirection(playerPosition) {
02.     var impact = game.ball.y - playerPosition -
    PLAYER_HEIGHT / 2;
03.     var ratio = 100 / (PLAYER_HEIGHT / 2);
```

```
04.
05.         // Get a value between 0 and 10
06.         game.ball.speed.y = Math.round(impact * ratio / 10);
07.     }
```

En appliquant une petite formule maison, on peut obtenir une valeur entre 0 et 10 en fonction du point d'impact de la balle (0 vers le centre, 10 sur les côtés). La trajectoire n'est pas forcément réaliste mais, tout comme pour l'IA de l'ordinateur, je souhaitais faire quelque chose de simple.

Appelons notre fonction lorsque le joueur tape dans la balle :

```
01.     // Increase speed and change direction
02.     game.ball.speed.x *= -1.2;
03.     changeDirection(player.y);
```

Et voilà notre jeu fonctionne ! On va juste rajouter un peu de style, d'interactivité et gérer les scores pour finir.

Étape 4 : Gestion du score et amélioration de l'interface

Un peu de style

Commençons par modifier le html pour rajouter l'affichage du score et les boutons pour gérer le lancement/arrêt de la partie :

```
<h1>PONG</h1>
<main role="main">
    <p>Joueur 1 : <em id="player-score">0</em> - Joueur 2 : <em id="computer-score">0</em></p>
    <ul>
        <li>
            <button id="start-game">Démarrer</button>
        </li>
        <li>
            <button id="stop-game">Arrêter</button>
        </li>
    </ul>
    <canvas id="canvas" width="640" height="480">
</canvas>
</main>
```

Et rajoutons un peu (mais alors un tout petit peu) de style :

```
<style>
    ul {
        list-style: none;
        padding: 0;
    }
    li {
        display: inline-block;
    }
</style>
```

Bon c'est pas fou mais ça fera l'affaire.

Gestion du score

Créons une nouvelle propriété dans nos objets player et computer.

```
player: {
    y: canvas.height / 2 - PLAYER_HEIGHT / 2,
    score: 0
},
computer: {
    y: canvas.height / 2 - PLAYER_HEIGHT / 2,
    score: 0
}
```

Dans notre fonction collide, lorsque le joueur ne tape pas la balle et après que celle-ci ait été remise au centre, on met à jour le score.

```
// Update score
if (player == game.player) {
    game.computer.score++;
    document.querySelector('#computer-score').textContent = game.computer.score;
} else {
    game.player.score++;
    document.querySelector('#player-score').textContent = game.player.score;
}
```

Interactions avec les boutons

Plutôt que de lancer la partie au chargement, nous allons attendre que l'utilisateur appuie sur le bouton pour

démarrer la partie. Dans notre code principal on retire l'appel de la fonction play et on la remplace par cette ligne :

```
01.  document.querySelector('#start-
     game').addEventListener('click', play);
```

On récupère l'élément bouton sur lequel on met en place un événement click qui va lancer la partie. Nous allons faire pareil pour le bouton arrêter la partie.

Dans la déclaration des variables, on rajoute cette ligne :

```
01.  var anim;
```

Puis dans la fonction play on va stocker la référence de l'animation qui est lancée.

```
01.  anim = requestAnimationFrame(play);
```

Enfin on rajoute l'événement (en-dessous de l'événement sur le bouton démarrer la partie) et la fonction qui sera appelée.

```
01.  document.querySelector('#stop-
     game').addEventListener('click', stop);
```

```
01.  function stop() {
02.      cancelAnimationFrame(anim);
03.
```

```
    // Set ball and players to the center
    game.ball.x = canvas.width / 2;
    game.ball.y = canvas.height / 2;
    game.player.y = canvas.height / 2 - PLAYER_HEIGHT / 2;
    game.computer.y = canvas.height / 2 - PLAYER_HEIGHT / 2;

    // Reset speed
    game.ball.speed.x = 2;
    game.ball.speed.y = 2;

    // Init score
    game.computer.score = 0;
    game.player.score = 0;

    document.querySelector('#computer-score').textContent = game.computer.score;
    document.querySelector('#player-score').textContent = game.player.score;

    draw();
}
```

On réinitialise les joueurs et la balle dans la zone de jeu et on remet le score à 0.

Voilà cette fonction conclut notre tuto « Créer un pong en JavaScript ». J'espère que cela vous a intéressé. N'hésitez pas à me faire part de vos remarques sur le tuto et/ou idées de nouveaux tutos. En attendant vous retrouverez le code complet (avec quelques

optimisations : refactorisation de code dupliqué, rajout de style) ici : https://github.com/devoreve/pong .

Bien sûr ce programme peut être amélioré mais cela vous fait déjà une bonne base 🙂 . Et puis n'oubliez pas le plus important : le code c'est la vie !

Ressources utiles

- https://developer.mozilla.org/fr/docs/Web/API/Window/requestAnimationFrame
- https://www.alsacreations.com/tuto/lire/1484-introduction.html
- https://www.alsacreations.com/article/lire/1445-dom-queryselector-queryselectorall-selectors-api.html

6 JUIN 2018

LAISSER UN COMMENTAIRE

FORMULAIRE DE CONTACT AVEC AJAX

LAISSER UN COMMENTAIRE

Votre adresse de messagerie ne sera pas publiée. Les champs obligatoires sont indiqués avec *

Commentaire

Nom *

Adresse de messagerie *

Site web

☐Enregistrer mon nom, mon e-mail et mon site web dans le navigateur pour mon prochain commentaire.

☐ J'ai lu et j'accepte la Politique de confidentialité *

LAISSER UN COMMENTAIRE