Tutoriel de création d'un jeu Pong avec HTML, CSS et JavaScript

Ce tutoriel vous guidera pas à pas dans la création d'un jeu Pong basique en utilisant:

- CSS Grid pour la structure de la page
- Flexbox pour l'alignement des éléments
- JavaScript pour les interactions et la logique du jeu

Table des matières

1. Structure HTML de base
2. Mise en page avec CSS Grid
3. Centrage avec Flexbox
4. Création du terrain de jeu
5. Ajout des éléments du jeu
6. Contrôles et gestion des événements
7. Logique de jeu
8. Collisions et score
9. Améliorations possibles

1. Structure HTML de base

Commençons par créer la structure HTML de base pour notre jeu:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Jeu Pong</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <header>
            <h1>Jeu Pong</h1>
        </header>

        <main>
            <!-- Le contenu du jeu sera ajouté ici -->
        </main>

        <footer>
            <p>Tutoriel Grid, Flex et Événements JavaScript</p>
        </footer>
    </div>
```

```
    <script src="script.js"></script>
</body>
</html>
```

Cette structure simple divise notre page en trois sections principales:

- Un en-tête (`header`) avec le titre du jeu
- Une section principale (`main`) qui contiendra notre jeu
- Un pied de page (`footer`) avec des informations complémentaires

2. Mise en page avec CSS Grid

Maintenant, utilisons CSS Grid pour organiser ces trois sections verticalement sur la page:

```
/* style.css */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

/* Layout principal utilisant Grid */
.container {
    display: grid;
    grid-template-columns: 1fr;
    grid-template-rows: auto 1fr auto;
    grid-template-areas:
        "header"
        "main"
        "footer";
    height: 100vh;
}

header {
    grid-area: header;
    background-color: #333;
    color: white;
    padding: 1rem;
}

main {
    grid-area: main;
    padding: 1rem;
}

footer {
    grid-area: footer;
    background-color: #333;
    color: white;
    padding: 1rem;
```

```
    text-align: center;
}
```

Explication:

- `display: grid` active la mise en page Grid.
- `grid-template-columns: 1fr` définit une seule colonne qui prend tout l'espace disponible.
- `grid-template-rows: auto 1fr auto` définit trois rangées: l'en-tête et le pied de page s'adaptent à leur contenu (`auto`), tandis que la section principale prend tout l'espace restant (`1fr`).
- `grid-template-areas` assigne des noms à ces zones pour les référencer facilement.
- `height: 100vh` fait en sorte que le conteneur occupe toute la hauteur de la fenêtre.

3. Centrage avec Flexbox

Maintenant, ajoutons Flexbox pour centrer le contenu de notre section principale:

```
main {
    grid-area: main;
    padding: 1rem;
    /* Centrage du contenu avec Flexbox */
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}
```

Explication:

- `display: flex` active la mise en page Flexbox.
- `flex-direction: column` empile les éléments verticalement.
- `align-items: center` centre les éléments horizontalement.
- `justify-content: center` centre les éléments verticalement.

4. Création du terrain de jeu

Ajoutons le HTML pour notre terrain de jeu et le tableau des scores:

```
<main>
    <div id="score-board">
        <div id="score-left">0</div>
        <div id="score-right">0</div>
    </div>

    <div id="game-board">
        <!-- Les éléments du jeu seront ajoutés ici -->
    </div>
```

```
    <div class="controls">
        <button id="start-btn">Démarrer</button>
        <button id="reset-btn">Réinitialiser</button>
    </div>
</main>
```

Et maintenant, le CSS pour styliser ces éléments:

```
/* Style pour le terrain de jeu */
#game-board {
    width: 600px;
    height: 400px;
    background-color: #000;
    position: relative;
    border: 2px solid #333;
    margin: 20px 0;
}

/* Style pour le tableau des scores */
#score-board {
    display: flex;
    justify-content: space-between;
    width: 600px;
    padding: 10px 0;
    color: #333;
    font-size: 24px;
    font-weight: bold;
}

/* Style pour les contrôles */
.controls {
    display: flex;
    justify-content: space-around;
    width: 100%;
    max-width: 600px;
    margin-top: 20px;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
}

button:hover {
    background-color: #45a049;
}
```

Explication:

- Le terrain de jeu (`#game-board`) est un conteneur noir de 600×400 pixels.
- Le tableau des scores (`#score-board`) utilise Flexbox pour espacer les scores des deux joueurs.
- Les contrôles (`.controls`) utilisent également Flexbox pour espacer les boutons.

5. Ajout des éléments du jeu

Ajoutons maintenant les raquettes et la balle dans notre terrain de jeu:

```
<div id="game-board">
    <div id="paddle-left" class="paddle"></div>
    <div id="ball"></div>
    <div id="paddle-right" class="paddle"></div>
</div>
```

Et le CSS pour ces éléments:

```
/* Styles pour les éléments du jeu */
.paddle {
    width: 10px;
    height: 80px;
    background-color: white;
    position: absolute;
}

#paddle-left {
    left: 10px;
    top: 160px;
}

#paddle-right {
    right: 10px;
    top: 160px;
}

#ball {
    width: 15px;
    height: 15px;
    background-color: white;
    border-radius: 50%;
    position: absolute;
    top: 192px;
    left: 292px;
}
```

Explication:

- Les raquettes (`.paddle`) sont des rectangles blancs de 10×80 pixels.
- La balle (`#ball`) est un carré blanc de 15×15 pixels avec `border-radius: 50%` pour le rendre rond.
- Tous ces éléments utilisent `position: absolute` pour être positionnés précisément dans le terrain de jeu et simplifier le traitement.

6. Contrôles et gestion des événements

Maintenant, passons au JavaScript pour gérer les contrôles du jeu :

```
// script.js

// Éléments du DOM
const gameBoard = document.getElementById('game-board');
const ball = document.getElementById('ball');
const paddleLeft = document.getElementById('paddle-left');
const paddleRight = document.getElementById('paddle-right');
const scoreLeft = document.getElementById('score-left');
const scoreRight = document.getElementById('score-right');
const startBtn = document.getElementById('start-btn');
const resetBtn = document.getElementById('reset-btn');

// Variables du jeu
let gameRunning = false;
let ballX = 292;
let ballY = 192;
let ballSpeedX = 5;
let ballSpeedY = 3;
let paddleLeftY = 160;
let paddleRightY = 160;
let leftScore = 0;
let rightScore = 0;
const paddleSpeed = 10;

// État des touches
const keys = {
    w: false,
    s: false,
    arrowUp: false,
    arrowDown: false
};

// Gestionnaire d'événement pour les touches
function handleKeyDown(e) {
    switch(e.key.toLowerCase()) {
        case 'w':
            keys.w = true;
            break;
        case 's':
            keys.s = true;
            break;
        case 'arrowup':
            keys.arrowUp = true;
```

```
            break;
        case 'arrowdown':
            keys.arrowDown = true;
            break;
    }
}

function handleKeyUp(e) {
    switch(e.key.toLowerCase()) {
        case 'w':
            keys.w = false;
            break;
        case 's':
            keys.s = false;
            break;
        case 'arrowup':
            keys.arrowUp = false;
            break;
        case 'arrowdown':
            keys.arrowDown = false;
            break;
    }
}

// Événements pour le clavier
document.addEventListener('keydown', handleKeyDown);
document.addEventListener('keyup', handleKeyUp);

// Événements pour les contrôles
startBtn.addEventListener('click', () => {
    if (!gameRunning) {
        gameRunning = true;
        gameLoop();
    }
});

resetBtn.addEventListener('click', resetGame);
```

Explication:

- Nous sélectionnons tous les éléments du DOM dont nous aurons besoin.
- Nous initialisons les variables pour la position des éléments et l'état du jeu.
- Nous définissons un objet `keys` pour suivre l'état des touches du clavier.
- Nous ajoutons des écouteurs d'événements pour les touches du clavier (`keydown` et `keyup`).
- Nous ajoutons des écouteurs d'événements pour les clics sur les boutons.

7. Logique de jeu

Implémentons maintenant les fonctions principales pour la logique du jeu:

```
// Fonction pour réinitialiser la balle
function resetBall() {
    ballX = 292;
    ballY = 192;
    // Direction aléatoire au démarrage
    ballSpeedX = Math.random() > 0.5 ? 5 : -5;
    ballSpeedY = Math.random() > 0.5 ? 3 : -3;
}

// Fonction pour réinitialiser le jeu
function resetGame() {
    gameRunning = false;
    resetBall();
    paddleLeftY = 160;
    paddleRightY = 160;
    leftScore = 0;
    rightScore = 0;
    scoreLeft.textContent = leftScore;
    scoreRight.textContent = rightScore;
    updatePositions();
}

// Fonction pour mettre à jour les positions des éléments
function updatePositions() {
    ball.style.left = ballX + 'px';
    ball.style.top = ballY + 'px';
    paddleLeft.style.top = paddleLeftY + 'px';
    paddleRight.style.top = paddleRightY + 'px';
}

// Fonction pour déplacer les raquettes
function movePaddles() {
    // Raquette gauche
    if (keys.w && paddleLeftY > 0) {
        paddleLeftY -= paddleSpeed;
    }
    if (keys.s && paddleLeftY < gameBoard.clientHeight - paddleLeft.clientHeight)
{
        paddleLeftY += paddleSpeed;
    }

    // Raquette droite
    if (keys.arrowUp && paddleRightY > 0) {
        paddleRightY -= paddleSpeed;
    }
    if (keys.arrowDown && paddleRightY < gameBoard.clientHeight -
paddleRight.clientHeight) {
        paddleRightY += paddleSpeed;
    }
}

// Boucle de jeu principale
function gameLoop() {
```

```
    if (gameRunning) {
        movePaddles();
        updateBall();
        updatePositions();
        requestAnimationFrame(gameLoop);
    }
}

// Initialisation des positions
resetGame();
```

Explication:

- `resetBall()` replace la balle au centre avec une direction aléatoire.
- `resetGame()` réinitialise toutes les variables du jeu.
- `updatePositions()` met à jour la position des éléments dans le DOM.
- `movePaddles()` déplace les raquettes en fonction des touches actuellement enfoncées.
- `gameLoop()` est la boucle principale qui fait avancer le jeu, en utilisant `requestAnimationFrame` pour obtenir une animation fluide.

8. Collisions et score

Finalement, implémentons la fonction pour gérer les collisions et le score :

```
// Fonction pour mettre à jour la position de la balle
function updateBall() {
    ballX += ballSpeedX;
    ballY += ballSpeedY;

    // Collision avec les bords haut et bas
    if (ballY <= 0 || ballY >= gameBoard.clientHeight - ball.clientHeight) {
        ballSpeedY = -ballSpeedY;
    }

    // Détection pour la raquette gauche
    if (
        ballX <= paddleLeft.clientWidth + 10 &&
        ballY + ball.clientHeight >= paddleLeftY &&
        ballY <= paddleLeftY + paddleLeft.clientHeight
    ) {
        ballSpeedX = -ballSpeedX;
    }

    // Détection pour la raquette droite
    if (
        ballX >= gameBoard.clientWidth - paddleRight.clientWidth -
ball.clientWidth - 10 &&
        ballY + ball.clientHeight >= paddleRightY &&
        ballY <= paddleRightY + paddleRight.clientHeight
    ) {
```

```
            ballSpeedX = -ballSpeedX;
        }

        // Balle qui sort du terrain
        if (ballX < 0) {
            // Point pour le joueur droit
            rightScore++;
            scoreRight.textContent = rightScore;
            resetBall();
        } else if (ballX > gameBoard.clientWidth - ball.clientWidth) {
            // Point pour le joueur gauche
            leftScore++;
            scoreLeft.textContent = leftScore;
            resetBall();
        }
    }
```

Explication:

- Nous mettons à jour la position de la balle en ajoutant sa vitesse à sa position.
- Nous détectons les collisions avec les bords haut et bas et inversons la direction verticale.
- Nous détectons les collisions avec les raquettes et inversons la direction horizontale.
- Nous détectons si la balle sort du terrain et attribuons un point au joueur correspondant.

9. Améliorations possibles

Pour améliorer ce jeu de base, vous pourriez :

1. **Améliorer la détection des collisions**
    - Faire varier l'angle de rebond en fonction de l'endroit où la balle touche la raquette
    - Ajouter des effets comme l'accélération de la balle au fil du temps
2. **Ajouter des effets visuels**
    - Effets de particules lors des collisions
    - Animations lors des points marqués
    - Effets de traînée derrière la balle
3. **Implémenter des niveaux de difficulté**
    - Modifier la vitesse des raquettes et de la balle
    - Ajouter des obstacles sur le terrain
4. **Ajouter des effets sonores**
    - Sons pour les rebonds, les points, etc.
    - Musique de fond
5. **Créer un menu de pause et d'options**
    - Permettre de mettre le jeu en pause

- Offrir des options de personnalisation (couleurs, vitesse, etc.)

Code Complet

HTML (index.html)

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Jeu Pong</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <header>
            <h1>Jeu Pong</h1>
        </header>

        <main>
            <div id="score-board">
                <div id="score-left">0</div>
                <div id="score-right">0</div>
            </div>

            <div id="game-board">
                <div id="paddle-left" class="paddle"></div>
                <div id="ball"></div>
                <div id="paddle-right" class="paddle"></div>
            </div>

            <div class="controls">
                <button id="start-btn">Démarrer</button>
                <button id="reset-btn">Réinitialiser</button>
            </div>

            <div class="rules">
                <div>
                    <h3>Contrôles Joueur 1</h3>
                    <p>Utilisez les touches <strong>W</strong> (haut) et
<strong>S</strong> (bas) pour déplacer la raquette gauche.</p>
                </div>
                <div>
                    <h3>Contrôles Joueur 2</h3>
                    <p>Utilisez les touches <strong>Flèche Haut</strong> et
<strong>Flèche Bas</strong> pour déplacer la raquette droite.</p>
                </div>
            </div>
        </main>

        <footer>
```

```
            <p>Tutoriel Grid, Flex et Événements JavaScript</p>
        </footer>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

CSS (style.css)

```
/* Styles de base */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

/* Layout principal utilisant Grid */
.container {
    display: grid;
    grid-template-columns: 1fr;
    grid-template-rows: auto 1fr auto;
    grid-template-areas:
        "header"
        "main"
        "footer";
    height: 100vh;
}

header {
    grid-area: header;
    background-color: #333;
    color: white;
    padding: 1rem;
}

main {
    grid-area: main;
    padding: 1rem;
    /* Centrage du contenu avec Flexbox */
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}

footer {
    grid-area: footer;
    background-color: #333;
    color: white;
```

```
    padding: 1rem;
    text-align: center;
}

/* Style pour le terrain de jeu */
#game-board {
    width: 600px;
    height: 400px;
    background-color: #000;
    position: relative;
    border: 2px solid #333;
}

/* Styles pour les éléments du jeu */
.paddle {
    width: 10px;
    height: 80px;
    background-color: white;
    position: absolute;
}

#paddle-left {
    left: 10px;
    top: 160px;
}

#paddle-right {
    right: 10px;
    top: 160px;
}

#ball {
    width: 15px;
    height: 15px;
    background-color: white;
    border-radius: 50%;
    position: absolute;
    top: 192px;
    left: 292px;
}

#score-board {
    display: flex;
    justify-content: space-between;
    width: 600px;
    padding: 10px 0;
    color: #333;
    font-size: 24px;
    font-weight: bold;
}

/* Flex pour les contrôles */
.controls {
    display: flex;
```

```
    justify-content: space-around;
    width: 100%;
    max-width: 600px;
    margin-top: 20px;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
}

button:hover {
    background-color: #45a049;
}

/* Grille pour les règles du jeu */
.rules {
    display: grid;
    grid-template-columns: 1fr 1fr;
    grid-gap: 20px;
    width: 100%;
    max-width: 600px;
    margin-top: 20px;
}

.rules div {
    background-color: #fff;
    padding: 10px;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

/* Media queries pour la responsivité */
@media (max-width: 650px) {
    #game-board, #score-board {
        width: 100%;
    }

    .rules {
        grid-template-columns: 1fr;
    }
}
```

JavaScript (script.js)

```
// Éléments du DOM
const gameBoard = document.getElementById('game-board');
const ball = document.getElementById('ball');
const paddleLeft = document.getElementById('paddle-left');
const paddleRight = document.getElementById('paddle-right');
const scoreLeft = document.getElementById('score-left');
const scoreRight = document.getElementById('score-right');
const startBtn = document.getElementById('start-btn');
const resetBtn = document.getElementById('reset-btn');

// Variables du jeu
let gameRunning = false;
let ballX = 292;
let ballY = 192;
let ballSpeedX = 5;
let ballSpeedY = 3;
let paddleLeftY = 160;
let paddleRightY = 160;
let leftScore = 0;
let rightScore = 0;
const paddleSpeed = 10;

// État des touches
const keys = {
    w: false,
    s: false,
    arrowUp: false,
    arrowDown: false
};

// Fonction pour réinitialiser la balle
function resetBall() {
    ballX = 292;
    ballY = 192;
    // Direction aléatoire au démarrage
    ballSpeedX = Math.random() > 0.5 ? 5 : -5;
    ballSpeedY = Math.random() > 0.5 ? 3 : -3;
}

// Fonction pour réinitialiser le jeu
function resetGame() {
    gameRunning = false;
    resetBall();
    paddleLeftY = 160;
    paddleRightY = 160;
    leftScore = 0;
    rightScore = 0;
    scoreLeft.textContent = leftScore;
    scoreRight.textContent = rightScore;
    updatePositions();
}

// Fonction pour mettre à jour les positions des éléments
```

```
function updatePositions() {
    ball.style.left = ballX + 'px';
    ball.style.top = ballY + 'px';
    paddleLeft.style.top = paddleLeftY + 'px';
    paddleRight.style.top = paddleRightY + 'px';
}

// Gestionnaire d'événement pour les touches
function handleKeyDown(e) {
    switch(e.key.toLowerCase()) {
        case 'w':
            keys.w = true;
            break;
        case 's':
            keys.s = true;
            break;
        case 'arrowup':
            keys.arrowUp = true;
            break;
        case 'arrowdown':
            keys.arrowDown = true;
            break;
    }
}

function handleKeyUp(e) {
    switch(e.key.toLowerCase()) {
        case 'w':
            keys.w = false;
            break;
        case 's':
            keys.s = false;
            break;
        case 'arrowup':
            keys.arrowUp = false;
            break;
        case 'arrowdown':
            keys.arrowDown = false;
            break;
    }
}

// Fonction pour déplacer les raquettes
function movePaddles() {
    // Raquette gauche
    if (keys.w && paddleLeftY > 0) {
        paddleLeftY -= paddleSpeed;
    }
    if (keys.s && paddleLeftY < gameBoard.clientHeight - paddleLeft.clientHeight)
{
        paddleLeftY += paddleSpeed;
    }

    // Raquette droite
```

```
    if (keys.arrowUp && paddleRightY > 0) {
        paddleRightY -= paddleSpeed;
    }
    if (keys.arrowDown && paddleRightY < gameBoard.clientHeight -
paddleRight.clientHeight) {
        paddleRightY += paddleSpeed;
    }
}

// Fonction pour mettre à jour la position de la balle
function updateBall() {
    ballX += ballSpeedX;
    ballY += ballSpeedY;

    // Collision avec les bords haut et bas
    if (ballY <= 0 || ballY >= gameBoard.clientHeight - ball.clientHeight) {
        ballSpeedY = -ballSpeedY;
    }

    // Détection pour la raquette gauche
    if (
        ballX <= paddleLeft.clientWidth + 10 &&
        ballY + ball.clientHeight >= paddleLeftY &&
        ballY <= paddleLeftY + paddleLeft.clientHeight
    ) {
        ballSpeedX = -ballSpeedX;
    }

    // Détection pour la raquette droite
    if (
        ballX >= gameBoard.clientWidth - paddleRight.clientWidth -
ball.clientWidth - 10 &&
        ballY + ball.clientHeight >= paddleRightY &&
        ballY <= paddleRightY + paddleRight.clientHeight
    ) {
        ballSpeedX = -ballSpeedX;
    }

    // Balle qui sort du terrain
    if (ballX < 0) {
        // Point pour le joueur droit
        rightScore++;
        scoreRight.textContent = rightScore;
        resetBall();
    } else if (ballX > gameBoard.clientWidth - ball.clientWidth) {
        // Point pour le joueur gauche
        leftScore++;
        scoreLeft.textContent = leftScore;
        resetBall();
    }
}

// Boucle de jeu principale
function gameLoop() {
```

```
    if (gameRunning) {
        movePaddles();
        updateBall();
        updatePositions();
        requestAnimationFrame(gameLoop);
    }
}

// Événements pour les contrôles
startBtn.addEventListener('click', () => {
    if (!gameRunning) {
        gameRunning = true;
        gameLoop();
    }
});

resetBtn.addEventListener('click', resetGame);

// Événements pour le clavier
document.addEventListener('keydown', handleKeyDown);
document.addEventListener('keyup', handleKeyUp);

// Initialisation des positions
resetGame();
```