Projet Web PWA

Morpion – Tic Tac Toe

PSB M2DB12

Axel Maison

Maxime Germain

Arthur Apolant

Charle

Première étape :

Afin de réaliser notre projet de Progressive Web App, nous avons décidé de recréer le jeu du morpion (ou tic tac toe en anglais) car nous pensions qu'il s'agissait d'un jeu à notre portée.

Le morpion est un jeu de réflexion se pratiquant à deux joueurs au tour par tour et dont le but est de créer le premier un alignement sur une grille. Le jeu se joue généralement avec papier et crayon.

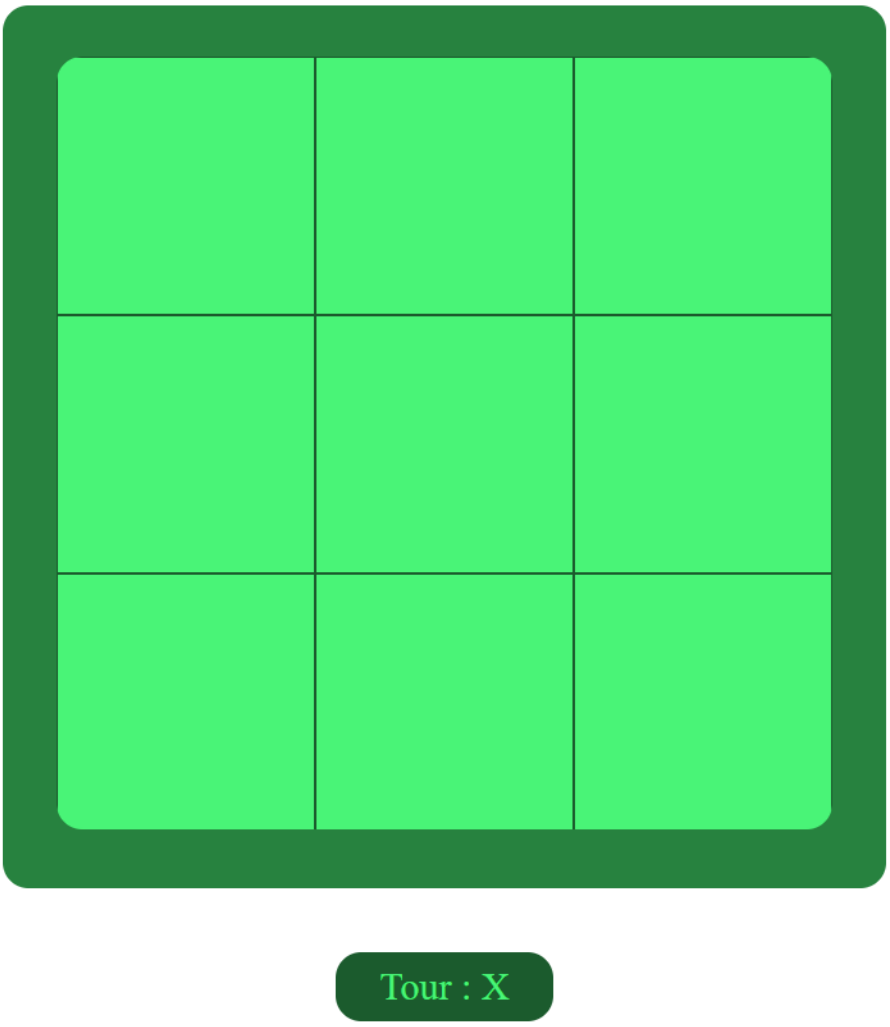
Deuxième étape :

Nous avons décidé de partir sur une grille de 3x3 pour être sûrs de pouvoir y arriver et nous avons choisi les langages Javascript Html et Css.

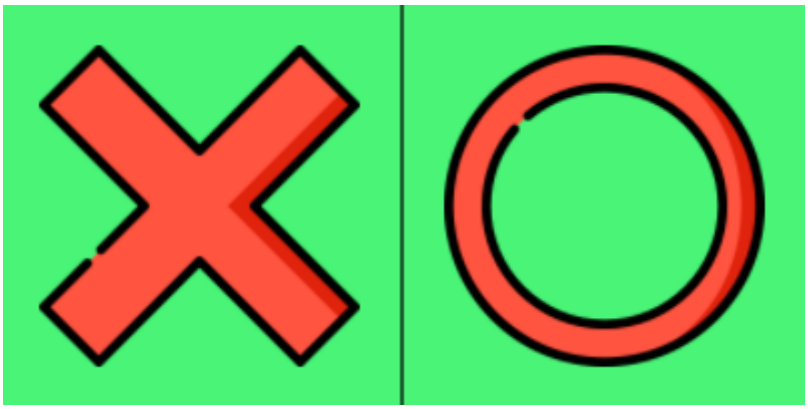
Nous commencerons par créer la page Html avec le style Css et nous passerons ensuite à l'implémentation du code Javascript.

Troisième étape :

Comme dit précédemment la première étape consiste en la création d'une page Html contenant la grille du jeu et affichant le tour du joueur en cours ou le nom du vainqueur. Une fois que les différents éléments ont été disposés, nous ajoutons le code Css et cela donne ça :



Nous avons ensuite choisi deux images à afficher pour représenter les coups des deux joueurs :



A présent, nous créons le fichier morpion.js afin d'y implémenter toutes les fonctions de notre jeu.

La première étape est de créer un tableau contenant 3 tableaux afin d'enregistrer les emplacements des différents coups des joueurs :

```
let plateau = [
    [0, 0, 0],
    [0, 0, 0],
    [0, 0, 0]
];

const img = new Map();
img.set("X", "./images/blue.png");
img.set("O", "./images/red.png");

let turn;
```

Ensuite, la fonction newGame permettant au début de la partie d'établir le tour de jeu au joueur « X » et de l'afficher :

```
function newGame() {
    turn = "X";
    document.getElementById("turn").innerHTML = "Tour : " + turn;
}
```

La fonction play est appelée à chaque fois qu'un joueur clique sur une case, elle prend en paramètre les coordonnées de cette dernière.

Si la case est vide, on place le pion du joueur à cet emplacement et on affiche l'image correspondante.

Ensuite, on vérifie si un joueur a gagné, sinon on change le tour de jeu et on l'affiche. Si oui, on empêche les joueurs de pouvoir cliquer à nouveau :

```
function play(x, y) {
    if (plateau[x][y] === 0) {
        plateau[x][y] = turn;
        document.getElementById(x.toString() + y.toString()).style.backgroundImage = "url(" + img.get(turn) + ")";
        if (!check()) {
            if (turn === "X") {
                turn = "O";
            } else {
                turn = "X";
            }
            document.getElementById("turn").innerHTML = "Tour : " + turn;
        } else {
            document.body.style.pointerEvents = "none";
        }
    }
}
```

Finalement, la fonction check nous permet de vérifier si un joueur a remporté la partie.

On commence par vérifier si trois fois le même pion est aligné sur une ligne à l'aide d'une boucle for parcourant la grille et de la méthode every qui retourne true si tous les éléments d'un tableau respectent la condition passée en paramètre. Si c'est le cas on affiche quel joueur a gagné :

```
function check() {
    for (let i = 0; i < 3; i++) {
        let every = plateau[i].every(p => p === turn);
        if (every) {
            document.getElementById("turn").innerHTML = "Victoire : " + turn;
            return true;
        }
    }
```

Ensuite, nous créons un nouveau tableau contenant les colonnes du tableau originel et nous le parcourons de la même manière que pour les lignes avec une boucle for et la méthode every :

```
let newlist = [···
];

for (let i = 0; i < 3; i++) {
    let every = newlist[i].every(p => p === turn);
    if (every) {
        document.getElementById("turn").innerHTML = "Victoire : " + turn;
        return true;
    }
}
```

Afin de vérifier également les diagonales nous utilisons deux if :

```
if (plateau[0][0] === turn
    && plateau[1][1] === turn
    && plateau[2][2] === turn)
{
    document.getElementById("turn").innerHTML = "Victoire : " + turn;
    return true;
}

if (plateau[0][2] === turn
    && plateau[1][1] === turn
    && plateau[2][0] === turn)
{
    document.getElementById("turn").innerHTML = "Victoire : " + turn;
    return true;
}
```

Finalement, nous vérifions s'il y a égalité en créant une variable x qui lors du parcours du tableau avec deux boucles for va augmenter à chaque fois qu'une case n'est pas vide. Si x est égal à 9, nous affichons qu'il y a égalité. Si aucune des conditions citées n'est respectée, nous retournons false :

```
let x = 0;
for (let i = 0; i < 3; i++) {
    for (let y = 0; y < 3; y++) {
        if (plateau[i][y] !== 0) {
            x++;
        }
    }
}

if (x === 9) {
    document.getElementById("turn").innerHTML = "Égalité";
    return true;
}

return false;
```

Quatrième étape :

La dernière étape consiste en l'ajout des fonctionnalités PWA.

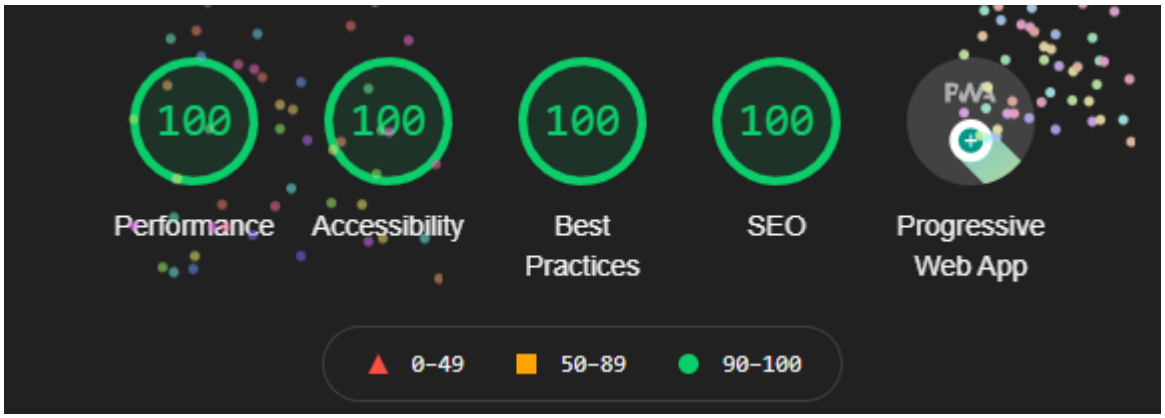
Nous avons commencé par créer le service worker :

```
self.addEventListener('install', e => {
    e.waitUntil(
        caches.open('pwa').then(cache => {
            return cache.addAll([
                '/',
                '/index.html',
                '/index.js',
                '/manifest.json',
                '/morpion.js',
                '/style.css',
                '/sw.js'
            ])
            .then(() => self.skipWaiting());
        })
    );
});
```

Et dans un second temps le fichier manifest.json :

```
{
    "name": "Morpion",
    "short_name": "Mp",
    "icons": [...
    ],
    "start_url": "/",
    "display": "standalone",
    "background_color": "red",
    "theme_color": "red"
}
```

Nous avons réussi à obtenir un score parfait sur LightHouse :


100
Performance
100
Accessibility
100
Best Practices
100
SEO
PWA
Progressive Web App
0-49
50-89
90-100