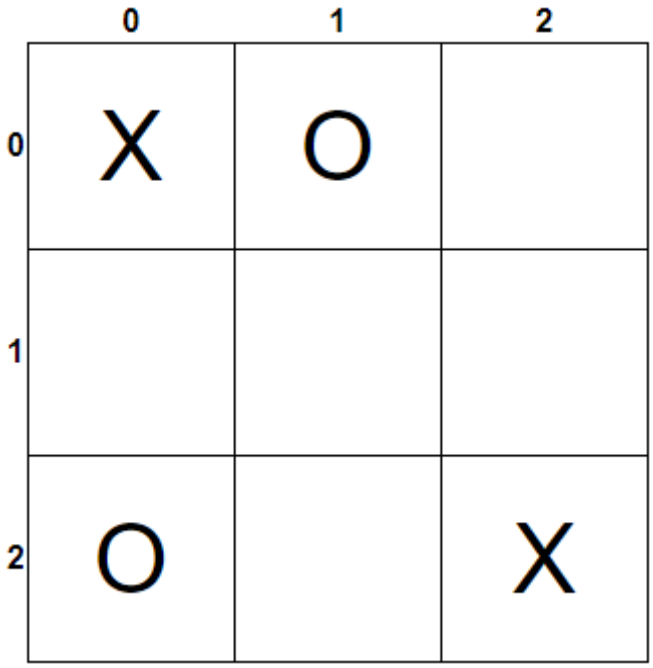
JavaScript

TP : Tic Tac Toe

Nicolas LETHUILLIER
nlethuillier@gmail.com

Introduction

Tic Tac Toe (aussi connu sous son nom de jeu à gratter « Morpion ») est un jeu dans lequel deux joueurs s'affrontent sur une grille de 9 cases.



Chaque joueur à son tour occupe une case en notant son propre symbole à l'intérieur (X ou O). La partie s'arrête lorsque l'un des joueurs est parvenu à aligner trois symboles identiques dans le sens horizontal, vertical ou diagonal (il a alors remporté la partie) ou lorsque la grille est complète (match nul).

L'objectif de ce TP est d'implémenter un jeu de Tic Tac Toe.

Pour ce faire, un fichier HTML et un fichier CSS vous sont fournis.

Au cours des explications, certaines alternatives vous seront proposées : une façon de faire « facile » (l'algorithme naïf) et une façon de faire « avancée » (afin d'obtenir un code plus propre/performant/efficace). Libre à vous d'opter pour l'énoncé qui vous sied selon le niveau de challenge que vous souhaitez.

La dernière partie « Pour aller plus loin » est facultative.

Prérequis

- ✓ Un navigateur web (un vrai, pas Internet Explorer)
- ✓ Un IDE

Initialisation

- Ouvrir le répertoire du projet avec un IDE.
- Créer le fichier `script.js` à la racine du projet.
- Modifier `tictactoe.html` pour appeler le fichier JavaScript (dans le `head`).

La base du projet est construite. On vous fournit déjà :

- Le fichier HTML qui contient la grille du jeu, ainsi que diverses div qui seront utilisées au fil du développement ;
- Le fichier CSS qui fait que la grille ressemble déjà à peu près à quelque chose.

Gestion du joueur actuel

Nous allons d'abord prévoir une variable qui indiquera si le joueur actuel est X ou O.

- Éditer le fichier `script.js` et déclarer la variable `symboleJoueurActuel`.

Oui, cette fois, on met les termes en français. Ça vous montre que d'un projet à l'autre, ça peut changer.

Initialisation

Au démarrage du jeu, il va falloir déterminer qui commence à jouer.

- Créer une fonction `init()`. Celle-ci contiendra l'ensemble des instructions de début de partie.

Version facile	Version avancée
○ Dans le corps de la fonction `init`, donner à `symboleJoueurActuel` la valeur « `X` ».	○ Dans le corps de la fonction `init`, faites en sorte que `symboleJoueurActuel` prenne aléatoirement « `X` » ou « `O` »

- Appeler `init` lorsque la page a fini de charger (car on a bien sûr intégré notre script dans la balise `head`, voir cours : « *Chargement de la page* »).

Clic sur une case

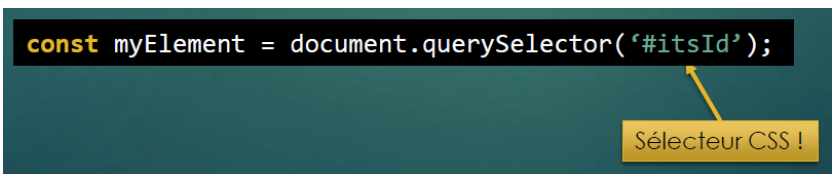
Chaque case a un identifiant de la forme « `cell-{X}-{Y}` ». Nous allons donc créer un événement au clic sur chacune des cases de façon à la remplir d'un X ou d'un O.

- Créer la fonction `jouer(x, y)` qui prend en entrée les coordonnées d'une case.
- Dans la fonction `jouer`, récupérer dans une constante `case` la case correspondante grâce à son identifiant.
- Mettre dans cette case (*innerText*) le symbole du joueur actuel.
- Tester en allant sur la page, en ouvrant la console (*F12 > Console*) et en appelant, par exemple, `jouer(1, 2)`.

Ça marche ! La case est remplie.

Nous allons donc maintenant attacher un événement au clic sur les cases de façon à ce que, lorsqu'on clique sur une case, on appelle automatiquement la fonction `jouer`.

- Compléter la fonction `init` pour récupérer chaque case et poser un événement au clic dessus. Le clic doit déclencher un appel à la fonction `jouer` avec les bons arguments. Pour rappel :




```
myElement.onclick = function() {
    // Do something
};
```

Version facile	Version avancée
○ Copier-coller la même (presque) opération pour les 9 cases	○ Utiliser une (double) boucle pour attacher les événements

- Tester en cliquant sur les cases.

Ça marche : on pose des croix partout. Mais il va falloir alterner, quand même, parce que tout seul, ce n'est pas passionnant, comme jeu.

Alternance des joueurs

- Compléter la fonction `jouer` de telle sorte qu'une fois qu'on a posé le symbole dans la case : a) Si le symbole du joueur actuel était un « X », il devient un « O » b) sinon, il devient un « X ».
- Tester en cliquant sur plusieurs cases : on doit normalement alterner entre les X et les O.

Nous allons ensuite afficher à l'utilisateur le joueur qui doit maintenant jouer.

- Compléter la fonction `init` pour créer une `div` avec l'identifiant « `tour-de-jeu` » (*voir cours : « Manipulation du DOM »*).
- Ajouter cette `div` à la fin de la balise `body`.

Cette `div` contiendra tantôt « *C'est au tour de X* », tantôt « *C'est au tour de O* ».

- Créer une fonction `miseAJourMessageJoueur()` qui modifier le contenu de la `div` précédemment créée en fonction du symbole du joueur actuel.
- Appeler cette fonction à l'issue de `init()`, ainsi qu'à chaque tour (dans `jouer`, après avoir changé le joueur actuel).

Erreur

Il y a un souci : on peut cliquer sur un symbole pour le remplacer par un autre… un peu abusé !

- Modifier la fonction `jouer` de façon à ce que l'ensemble des instructions (remplir la case, changer de joueur) ne soit exécuté que si la cellule est vide (*indices : innerText, chaîne de caractères vide*).
- Si la cellule n'est pas vide (*indice : sinon…*), afficher à l'utilisateur une pop-up (*alert*) l'informant qu'il ne peut pas jouer sur cette case.
- Tester en cliquant sur une case non vide.

Match nul

Intéressons-nous aux conditions de fin de partie. Une partie de Tic Tac Toe se termine en match nul si la grille est remplie (i.e. s'il n'y a plus aucune case vide).

- Créer la fonction `matchNul()` qui retourne `true` si la grille est complète, `false` sinon.

Version facile	Version avancée
○ Dans la fonction `matchNul`, créer une variable `toutesCasesPleines` initialisée à `true`. ○ Pour chaque case (on peut copier-coller le même code 9 fois si besoin), si son `innerText` est vide, mettre `toutesCasesPleines` à `false`. ○ Retourner `toutesCasesPleines`.	○ Utiliser `querySelectorAll`, le pseudo-sélecteur CSS `:empty` et le nombre d'éléments trouvés.

- Compléter `jouer` pour que, après avoir déposé le symbole dans la case, mais avant de changer de joueur, on appelle `matchNul`. Si la fonction retourne `true`, on affiche une pop-up indiquant « *Match nul, la partie est terminée.* » et on fait disparaître la `div#tour-de-jeu` (*en jouant sur son display, voir cours*). Si la fonction retourne `false`, on continue la partie (alternance du joueur actuel, mise à jour du message).

Condition de victoire

Le vainqueur d'une partie de Tic Tac Toe est le joueur qui parvient le premier à aligner 3 de ses symboles sur la grille, dans le sens horizontal, vertical ou diagonal.

- Déclarer, au début du script, la variable `partieFinie`.
- Compléter `init` pour initialiser `partieFinie` à `false`.
- Modifier `jouer` pour que l'ensemble de son corps ne soit exécuté que si `partieFinie` vaut `false`.

- Créer la fonction `victoire(x, y)` qui prend en entrée la position du jeton que le joueur actuel vient de poser. Cette fonction doit retourner `true` si ce dernier coup (*en x,y*) est un coup gagnant, `false` sinon.

Version facile	Version avancée
○ Récupérer la fonction à la dernière page de ce document.	○ Utiliser x et y pour regarder dans toutes les directions et déterminer s'il y a, grâce à ce jeton, trois mêmes symboles alignés.

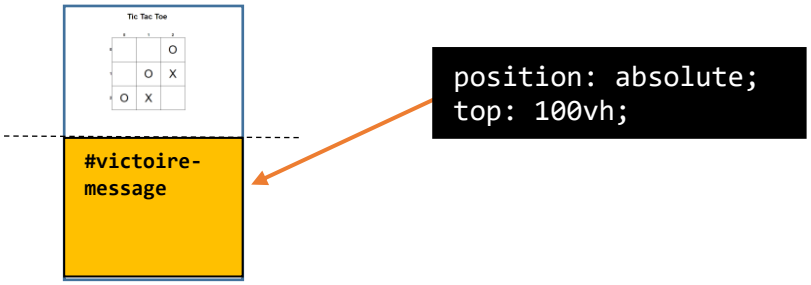
- Compléter `jouer` pour que, une fois qu'on a posé le jeton :
  - Si `victoire` retourne `true`, on affiche une pop-up indiquant « *{Joueur courant} a gagné !* », on place `true` dans `partieFinie`, et on masque la `div#tour-de-jeu`.
  - Sinon, si `matchNul()` … — *déjà fait*
  - Sinon … — *déjà fait*

Un peu de manipulation du DOM

Tous ces X et ces O, c'est bien joli, mais on aimerait égayer un peu tout ça.

- Modifier `jouer` pour que, juste après avoir joué, on ajoute à la case cliquée la classe « `cellule-x` » ou la classe « `cellule-o` » (*indice : voir cours, il y a un exemple d'ajout de classe*).
- Modifier la feuille de style pour donner aux cellules `.cellule-x` et `.cellule-o` une couleur de fond qui les distingue.

On souhaite ensuite ajouter, en cas de victoire, un message à l'écran « Victoire de {joueur} ». Pour ce faire, il existe une div avec l'identifiant message-victoire dans le HTML. Elle n'apparaît pas car elle est positionnée en absolute « sous » l'écran.



- Dans `jouer`, en cas de victoire, mettre dans la `div#victoire-message` le texte qu'on avait auparavant mis dans la pop-up (supprimer la pop-up).
- Dans la foulée, modifier la valeur top du style de cette `div` pour la faire remonter (`top` passe à `0` au lieu de `100vh`).

Pour aller plus loin

On peut imaginer ajouter à ce jeu :

- Un peu de design (ça mange pas de pain) ;
- Un bouton qui permet, en cas de victoire ou de match nul, de relancer une partie (rafraîchir la page) ;
- Un chronomètre (avec `setInterval`)
- Etc.

Spoiler alert

N’aller à la page suivante que pour récupérer le corrigé de la fonction « victoire ».

Annexe

```
function victoire(x, y) {
    let alignes = 0;
    // Nord - sud
    for (let dir = -2; dir <= 2; dir++) {
        if (document.querySelector('#cell-' + x + '-' + (y + dir))
            && document.querySelector('#cell-' + x + '-
' + (y + dir)).innerText === symboleDuJoueurActuel) {
            alignes++;
        }
    }
    if (alignes >= 3) {
        return true;
    }

    // Est - ouest
    alignes = 0;
    for (let dir = -2; dir <= 2; dir++) {
        if (document.querySelector('#cell-' + (x + dir) + '-' + y)
            && document.querySelector('#cell-' + (x + dir) + '-
' + y).innerText === symboleDuJoueurActuel) {
            alignes++;
        }
    }
    if (alignes >= 3) {
        return true;
    }

    // Sud-est - Nord-ouest
    alignes = 0;
    for (let dir = -2; dir <= 2; dir++) {
        if (document.querySelector('#cell-' + (x + dir) + '-' + (y + dir))
            && document.querySelector('#cell-' + (x + dir) + '-
' + (y + dir)).innerText === symboleDuJoueurActuel) {
            alignes++;
        }
    }
    if (alignes >= 3) {
        return true;
    }

    // Sud-ouest - Nord-est
    alignes = 0;
    for (let dir = -2; dir <= 2; dir++) {
        if (document.querySelector('#cell-' + (x + dir) + '-' + (y - dir))
            && document.querySelector('#cell-' + (x + dir) + '-
' + (y - dir)).innerText === symboleDuJoueurActuel) {
            alignes++;
        }
    }
    if (alignes >= 3) {
        return true;
    }

    return false;
}
```