582-31F-MA Programmation d’interface Web 2

TP 1

Description :

Produire une application dynamique à l'aide de *JavaScript* utilisant la programmation orientée objet modulaire.

Pondération

25%

Modalité particulière

Travail individuel

Sujet

Le 21

Date de remise

Cours 12 - lundi 14 mars 2022

Modalités de remise

- Le dossier zippé, à votre nom (nomdefamille-prenom) doit m'être remis sur Léa avant le 14 mars 23h59.
- Le site doit être en ligne avant le 14 mars 23h59.
- Dans ce dossier, vous devez inclure un fichier `.txt` avec l'*URL* de votre site.

Retard

Selon les règles du collège, 5% par jour de retard seront enlevé, jusqu'à 5 jours de retard maximum.

Règles du jeu

Le jeu qui vous est demandé de développer s'inspire du 21, en plus simple et sans argent.

Au début de chaque partie, on demande à l'utilisateur combien de joueurs seront de la partie (champ *number* obligatoire). Une fois le chiffre saisi, on instancie ce nombre de joueurs. À tour de rôle, chaque joueur peut demander de recevoir une carte ou arrêter de jouer. Le ou les joueur(s) qui atteindra le plus près la valeur de 21 - sans dépasser - par l'addition de ses cartes gagnera la partie.

La valeur des cartes se décline ainsi :
- as = 11
- 2 à 10 = leur valeur nominative
- figure = 10

Un joueur qui dépasse 21 ne pourra plus jouer et un joueur qui stoppe son attribution de carte ne se verra plus offrir la possibilité de recevoir une nouvelle carte.

Le jeu est terminé et affiche le ou les gagnant(s) lorsque tous les joueurs ont fini de jouer, soit parce qu'ils ont dépassé 21 ou qu'ils ont confiance en leur main et stoppé leur partie. Évidemment, si tous les joueurs dépassent 21, il n'y a pas de gagnant(s).

À la fin de chaque partie, le nombre de partie(s) jouée(s) et un bouton 'Rejouer' sont affichés. Le bouton 'Rejouer' permet de recommencer la partie sans rafraichir la page, le joueur peut alors saisir un nouveau nombre de joueurs dans le formulaire initial et jouer une nouvelle partie.

Fonctionnalités principales

- Le formulaire initial doit avoir un nombre positif dépassant 0.
- Nombre de joueurs dynamique selon le nombre saisi.
- Un seul joueur peut jouer à la fois.
- Les joueurs jouent à tour de rôle.
- Le joueur peut demander une carte, celle-ci sera affichée dynamiquement et associée au joueur (vous n'avez pas à enlever la carte tirée du jeu, ainsi plusieurs joueurs pourraient avoir la même carte).
- Total dynamique de la valeur additionnée des cartes de chaque joueur.
- Le joueur peut arrêter de jouer.
- Le joueur a perdu si son total dépasse 21.
- Un joueur qui a fini/perdu ne peut plus jouer.
- Styles cohérents selon l'état actuel de chaque joueur.
- Lorsque tous les joueurs ne peuvent plus jouer, le jeu met en valeur le gagnant.
- S'il y a égalité, le jeu met en valeur les gagnants.
- Le jeu n'affiche aucun gagnant si tous les joueurs dépassent 21.
- À la fin de la partie, le jeu affiche le nombre de parties jouées depuis l'ouverture de l'onglet navigateur (via un `sessionStorage`).
- À la fin de la partie, le joueur peut retourner au formulaire initial sans rechargement de page et saisir un nouveau nombre de joueurs afin de commencer une nouvelle partie.

Stratégies de développement

Ce jeu va vous demander beaucoup de réflexion. L'idée est de bien comprendre la portée et le comportement attendu de chaque élément du jeu, comme si vous y étiez en personne. Pour vous aider, voici la structure de scripts que j'ai utilisé :

main.js

- Traite le formulaire initial.
- Lance le jeu (classe `Board`).

Board.js

- Créer le *DOM* de chaque joueur selon le nombre reçu.
- Chaque joueur instancie la classe `Player`.

Player.js

- Mécanique de chaque joueur individuellement.
- Hérite de la classe `Game`.
- Gestion du cas 'Jouer', plus précisément :
  - tirage d'une carte (classe `Card`) ;
  - affichage de la carte ;
  - gestion du pointage et l'état du joueur.
- Gestion du cas 'Stop'.

Game.js

- Mécaniques de jeu communes à tous les joueurs :
  - joueur suivant ;
  - comportements en fin de la partie.

Card.js

- Détails de la carte tirée.

Exigences techniques et critères d'évaluation

- Réussite des différentes fonctionnalités.
- Utilisation de classe *ES6* de type *Module*.
- Absence de fonctions fléchées (*arrow functions*).
- Ergonomie générale : ce n'est pas un travail de *CSS*, mais les styles doivent offrir un comportement de jeu intuitif.
- Structure du code.
- Code sémantique.
- Commentaires fréquents et pertinents.
- Qualité des algorithmes et qualité du code source.
- Le jeu est en ligne.

Plagiat

Nous travaillons dans le Web, alors les réponses s'y trouvent et il va de soi que vous allez devoir chercher sur *Google*. Dans l'esprit *open-source* du Web, vous avez tout à fait le droit de copier-coller un bout de code que vous avez trouvé. Cela-dit, vous devez citer les extraits de code qui dépasse une ligne ou deux et suivre un tutoriel (en tout ou en partie) sera considéré comme du plagiat.

Il s'agit d'un travail individuel, vous pouvez poser des questions à vos collègues, mais ce doit être votre logique et vos scripts. L'objectif est de développer votre autonomie. En cas de plagiat, je devrai appliquer les sanctions de la *PIEA*.

Si vous êtes vraiment bloqué, contactez-moi.