

Langages Dynamiques
Master 1 DS Apprentissage
Nom :
Prénom:

TP Noté

durée 2h00

Modalités

L'interrogation dure 2h00 et est sur 20 points. L'accès aux machines est autorisé (y compris les supports de cours, la console JavaScript de Chrome et plus généralement l'accès à Internet). Les réponses sont à écrire sur l'énoncé (n'oubliez pas vos nom et prénom en haut à gauche). Le code de l'exercice 3 doit être rendu grâce au formulaire prévu à cet effet :

https://www.lri.fr/~kn/rendu_ld.php

1 QCM (6 points)

Pour chaque affirmation de cet exercice, vous pouvez

- entourer √ si vous pensez qu'elle est vraie;
- entourer × si vous pensez qu'elle est fausse;
- laisser la ligne vierge si vous ne savez pas.

Une bonne réponse rapporte 0.5 point, une mauvaise réponse -0.25 point et une absence de réponse 0 point.

Question 1

Le langage JavaScript ...

est un langage statiquement typé	√ ×
est exécutable par la plupart des navigateurs Web	√ ×
permet de comparer des valeurs de types différents	√ ×
permet de modifier la structure d'une page HTML	√ ×

Réponse:

- exécutable dans la plupart des navigateurs Web
- permet de comparer des valeurs de types différents
- permet de modifier la structure d'une page HTML

Question 2

On considère le code ci-dessous :

```
class Point {  
  constructor (x, y) {  
    this.x = x;  
    this.y = y;  
  }  
  
  move = function (i, j) {  
    this.x += i;  
    this.y += j;  
  }  
}
```

```

};
let p1 = new Point(0,0);
let p2 = new Point(0,0);
let p3 = new Point(0,0);

p2.move = p3.move;
p3.move = function () { };
p1.move = function (i, j) { p2.x += i; p2.y += j; };

p1.move(1,1);
p2.move(1,1);
p3.move(1,1);

```

console.log(p1.x + ", " + p1.y); affiche « 1, 1 » dans la console.	V ×
console.log(p2.x + ", " + p2.y); affiche « 1, 1 » dans la console.	V ×
console.log(p2.x + ", " + p2.y); affiche « 2, 2 » dans la console.	V ×
console.log(p3.x + ", " + p3.y); affiche « 1, 1 » dans la console.	V ×

Réponse:

— console.log(p2.x + ", " + p2.y); affiche « 2, 2 » dans la console.

Question 3

On considère l'expression régulière r valant $/([a-z]|[A-Z])^*([0-9]+)$/$. On considère le résultat v de $r.test(s)$.

si s commence par un chiffre, alors v vaut toujours true	V ×
si s contient un chiffre, alors v vaut toujours true	V ×
s peut contenir autre chose qu'une lettre ou un chiffre et v peut quand même valoir true	V ×
si s est vide, v vaut toujours false	V ×

Réponse:

- s peut contenir autre chose qu'une lettre ou un chiffre et v peut quand même valoir true
- si s est vide, v vaut toujours false

La première affirmation est **incorrecte**, car on peut trouver une chaîne commençant par un chiffre telle que $r.test(s)$ vaut false (par exemple $s = "9x"$). La seconde affirmation est **incorrecte**, car on peut trouver une chaîne contenant telle que $r.test(s)$ vaut false (par exemple $s = "9x"$).

2 Promesse (4 points)

On souhaite utiliser des promesses pour encapsuler des événements DOM. Lire le code et répondre aux questions ci-dessous :

```

function eventHandler(obj, eventName) {

    return new Promise((next, err) => {
        obj.addEventListener(eventName, next, { once : true });
    });
}

```

Le paramètre `{once : true }` indique qu'une fois l'évènement déclenché, le gestionnaire est retiré (la fonction `next`) ne s'exécute qu'une fois, même si l'évènement est déclenché plusieurs fois.

On suppose que dans la suite il existe trois boutons HTML dont les ids sont respectivement `b1`, `b2`, `b3` et que leur DOM est récupéré au moyen de

```
const b1 = document.getElementById("b1");
const b2 = document.getElementById("b2");
const b3 = document.getElementById("b3");

// Code à compléter ici...
```

Dans la suite vous pouvez utiliser les méthodes explicites (`.then` par exemple) ou la notation `await` ou un mélange des deux.

Questions

1. (1 point) Écrire une portion de code permettant de capter, dans cet ordre, un clic sur `b1`, un clic sur `b2`, et un clic sur `b3`.

Réponse

Réponse:

```
await eventHandler(b1, "click");
await eventHandler(b2, "click");
await eventHandler(b3, "click");
```

Ici, il est inutile (mais pas incorrect) d'affecter le résultat du `await` dans une variable. On attends le premier clic. Lorsqu'il se produit, alors la promesse est résolue et le `await` rend la main. On peut alors attendre le second clic, puis le troisième.

2. (1.5 point) Écrire une portion de code permettant de capter un clic sur n'importe lequel des trois boutons.

Réponse

Réponse:

```
const t = [ eventHandler(b1, "click"), eventHandler(b2, "click"),
            eventHandler(b3, "click") ];
await Promise.race(t);
```

Ici on crée un tableau de trois promesses (en cours d'exécution), puis on attend que l'une d'elle soit résolue au moyens de `Promise.race`.

3. (1.5 point) Écrire une portion de code permettant de capter un clic sur chacun des trois boutons, dans un ordre quelconque.

Réponse

Réponse:

```
const t = [ eventHandler(b1, "click"), eventHandler(b2, "click"),
  eventHandler(b3, "click") ];
await Promise.all(t);
```

Ici on crée un tableau de trois promesses (en cours d'exécution), puis on attend qu'elles soient résolues au moyens de `Promise.all`.

3 Memory (10 points)

Le but de cet exercice est de compléter le code se trouvant dans le fichier `memory.js`. Ce dernier implémente le célèbre jeu de cartes :

- 10 paires de cartes sont posées face retournée sur une table.
- Un joueur peut choisir deux cartes et les retourner
- Si les deux cartes sont identiques, elles restent face découverte
- Si les deux cartes sont différentes, elles sont cachées de nouveau au bout de 5 secondes.
- La partie se termine lorsque toutes les cartes ont été retournées.

Le jeu est implémenté en HTML et Javascript. La partie HTML est complètement écrite, vous ne devez modifier et rendre **que** le fichier `memory.js`. La structure du code est la suivante :

- tout le code est situé dans le gestionnaire de l'évènement `load` de la fenêtre
- Un objet singleton `GAME` qui possède trois propriétés décrivant l'état courant du jeu :

CARDS initialisé par la fonction `reset()` contient un tableau 4 cases, chacune contenant un tableau de 5 cases, chacune contenant un objet `Card` (donc 20 cartes au total)

REMAINING , qui contient le nombre de carte restant à découvrir, initialisé à 20

SELECTED , un tableau d'au plus deux cases, contenant les deux cartes que l'on souhaite retourner.

- Une classe `Card` représente les objets cartes. Une carte possède les propriété suivantes :

visible : un booléen indiquant si la carte est retournée ou pas

selected : un booléen indiquant si la carte est sélectionnée ou pas

number : un entier entre 0 et 9 qui représente la valeur de la carte

display : un élément HTML `td` correspondant à l'une des 20 cases de la table HTML

toggle() : une méthode qui permet de sélectionner ou désélectionner la carte

flip() : une méthode qui permet de retourner la carte

- Une fonction auxiliaire `shuffleArray(tab)` qui permet de mélanger les cases d'un tableau en place
- Une fonction `initArray()` qui renvoie un tableau de tableaux d'objets `Card`, aléatoirement mélangés
- Un ensemble de déclarations permettant de récupérer des références vers les éléments HTML du jeu
- Une fonction `victory()` qui rend visible le message de victoire
- Une fonction `reset()` qui réinitialise une partie
- Des gestionnaires d'évènements `"click"` pour les deux boutons et les cases du plateau

La page du cours contient des captures d'écran présentant le déroulement d'une partie. Il est suggéré de lire attentivement les méthodes `flip()` et `toggle()` de la classe `Card`, avant de faire les question 7 à 10.

Questions

1. (1.5 point) Compléter le code de la fonction `shuffleArray` pour cette dernière mélange le tableau `tab` donné en entrée de manière aléatoire. Vous pouvez utiliser l'algorithme de Fischer-Yates dont le pseudo code est le suivant :

Entrée: un tableau `T` de taille `N` (indiqué entre `0` et `N-1`)

Sortie: aucune

Effet: les cases du tableau passé en entrée sont mélangés aléatoirement

```
pour i allant de 0 à N-2
    soit j un nombre aléatoire tel que  $i \leq j < N$ 
    échanger T[i] et T[j]
fin pour
```

Les fonctions suivantes sont à votre disposition :

— `Math.random()` : renvoie un nombre aléatoire flottant entre 0 (inclus) et 1 (exclus)

— `Math.trunc(n)` : renvoie la partie entière du nombre flottant `n`.

2. (2 points) Compléter le code de la fonction `initArray()`, pour qu'elle renvoie un tableau de tableaux d'objets `Card`. On pourra procéder comme suit :
 - Créer un tableau `tab` de vingt entiers 0, 0, 1, 1, 2, 2, ..., 9, 9
 - Mélanger ce tableau au moyen de `shuffleArray`
 - Utiliser le tableau mélangé pour créer un tableau de tableaux `result`, de taille 4×5 , de telle sorte que `result[i][j]` contiennent un objet `Card` construit à partir de l'entier `tab[5 × i + j]` et l'élément `td` d'id "`id_i_j`".
3. (0.5 point) Initialiser les quatre variables `victoryDiv`, `flipButton`, `table` et `resetButton` pour qu'elles contiennent respectivement les éléments HTML correspondant au div d'id "`id_victory`", à la table HTML, et aux deux boutons (cf. le code HTML).
4. (0.5 point) Compléter la fonction `victory()` pour qu'elle change la propriété CSS `display` de l'élément `victoryDiv` en la mettant à "`block`".
5. (1.5 point) Compléter la fonction `reset()` pour qu'elle qu'elle change la propriété CSS `display` de l'élément `victoryDiv` en la mettant à "", et qu'elle l'objet `GAME` ainsi que le document HTML :
 - `REMAINING` vaut 20
 - `SELECTED` est un tableau vide
 - `CARDS` est un tableau obtenu par `initArray()`
 - pour chaque élément `td` se trouvant dans la table, son contenu est supprimé (en utilisant l'API `Dom` ou l'attribut `innerHTML`).
6. (0.5 point) Faire en sorte que lorsqu'on clique sur le bouton `Reset`, la fonction `reset()` soit appelée.
7. (1.5 point) Faire en sorte que lorsqu'on clique sur un élément `td`, la carte correspondante soit sélectionnée ou désélectionnée (méthode `toggle()` de la classe `Card`). On rappelle que chaque élément `td` possède un id de la forme "`id_i_j`" et que la carte correspondante se trouve dans `GAME.CARDS[i][j]`.
Attention vous ne devez utiliser qu'un seul gestionnaire d'évènement sur l'élément HTML `table` et vous servir de la délégation d'évènements pour répondre à cette question. Il est **interdit** d'accrocher un gestionnaire pour chaque élément `td`.
8. (2 points) Faire en sorte que lorsqu'on clique sur le bouton `Flip` ! il se produise les choses suivantes :
 - S'il n'y a pas exactement deux cartes sélectionnées (cf. le tableau `GAME.SELECTED`) ne rien faire
 - Sinon
 - désactiver le bouton `Flip` !. Un objet `DOM o` représentant un élément HTML interactif (bouton, champs de texte, ...) peut être désactivé en faisant `o.disabled = true` et réactivé en faisant `o.disabled = false`.
 - retourne les deux cartes sélectionnées (méthode `flip()` de la classe `Card`)
 - Si les deux cartes sont de même valeur, décrémente de 2 `GAME.REMAINING`, appelle `toggle()` sur les deux cartes sélectionnées (pour les désélectionner) et réactive le bouton `Flip` !. S'il ne reste plus de cartes à retourner, appelle la fonction `victory()`.
 - Sinon, installe un *timer* qui effectue au bout de 5 secondes les actions (dans cet ordre) : retourner les deux cartes sélectionnées, désélectionner les deux cartes sélectionnées, et réactiver le bouton `Flip` !.

Vous devez rendre le code au moyen du formulaire disponible sur la page du cours.