Prog web client riche - JavaScript
IUT Montpellier-Sète — Département Informatique
TD2
Thème : objets, classes

INTRODUCTION

Pour ce TD2, vous disposez d'une architecture web comprenant un fichier html, un fichier css et quelques autres fichiers. Clonez cette architecture dans un dossier `JS/TD2` de votre `public_html`.

Dans le TD1 nous avons eu une approche fonctionnelle de JavaScript : tout notre code a été construit autour de quelques fonctions, et c'était suffisant.

Dans ce TD2, vous allez devoir construire des objets JavaScript, sans effet immédiat sur le navigateur. La console du navigateur sera par contre d'une grande utilité (touche F12)

Quelques éléments de compréhension extra-JavaScript sont utiles pour réaliser ce TD :
Dans un championnat de foot à 8 équipes, chaque équipe rencontre chacune des sept autres équipes deux fois (match aller et match retour) : une fois sur son terrain (à domicile) et une fois sur le terrain de l'autre équipe (à l'extérieur).

Un championnat à 8 équipes comporte donc 14 journées, chaque journée étant composée de 4 matchs. Le planning de ces 14 journées est donné dans le tableau suivant :

	Match n°1	Match n°2	Match n°3	Match n°4
Journée n°01	E1 — E2	E3 — E4	E5 — E6	E7 — E8
Journée n°02	E1 — E3	E2 — E4	E5 — E7	E6 — E8
Journée n°03	E4 — E1	E3 — E2	E8 — E5	E7 — E6
Journée n°04	E1 — E5	E2 — E6	E3 — E7	E4 — E8
Journée n°05	E8 — E1	E2 — E7	E6 — E3	E4 — E5
Journée n°06	E6 — E1	E5 — E2	E8 — E3	E7 — E4
Journée n°07	E1 — E7	E2 — E8	E3 — E5	E4 — E6
Journée n°08	E3 — E1	E4 — E2	E7 — E5	E8 — E6
Journée n°09	E1 — E4	E2 — E3	E5 — E8	E6 — E7
Journée n°10	E5 — E1	E6 — E2	E7 — E3	E8 — E4
Journée n°11	E2 — E1	E4 — E3	E6 — E5	E8 — E7
Journée n°12	E1 — E8	E7 — E2	E3 — E6	E5 — E4
Journée n°13	E1 — E6	E2 — E5	E3 — E8	E4 — E7
Journée n°14	E7 — E1	E8 — E2	E5 — E3	E6 — E4

Ce planning est arrangé pour qu'une équipe ne joue pas plus de 2 matchs consécutifs à domicile (idem à l'extérieur).

Chaque match verra son issue déterminée par un tirage au sort :
- l'équipe jouant à domicile marque un certain nombre de buts ;
- idem pour l'équipe jouant à l'extérieur ;
- en cas de victoire d'une équipe, elle marque 3 points, et l'autre 0; en cas de match nul, chaque équipe marque 1 point ;

On comptabilise en permanence le nombre total de buts marqués par une équipe, ainsi que le nombre total de buts encaissés. Cela peut servir, au niveau du classement, à départager les équipes ayant le même nombre de points.

Les critères pour le classement sont, dans l'ordre :
- le nombre total de points ;
- la différence entre buts marqués et buts encaissés ;
- la meilleure attaque (nombre de buts marqués) ;
- un nombre aléatoire entre 0 et 1 qui départage les ex-aequo éventuels.

EXERCICE 1 - l'objet Equipe

Une Equipe aura comme attributs :

- son nom ;
- son classement ;
- son nombre de points ;
- son nombre de matchs gagnés (pour enrichir l'affichage) ;
- son nombre de matchs nuls (idem) ;
- son nombre de match perdus (idem) ;
- son nombre de buts pour (les buts marqués) ;
- son nombre de buts contre (les buts encaissés) ;
- son « évaluation » qui permettra de définir son classement.

Exemple de calcul d'évaluation : une équipe qui aurait cet état :

21 points, 23 buts pour, 11 buts contre

aura pour évaluation **de base** le nombre obtenu par le calcul

`21 * 10000 + (23 - 11) * 100 + 23`, ce qui donne `211223` et ce nombre donne donc la priorité aux points, puis à la différence de buts (buts marqués - buts encaissés), puis à l'attaque (buts marqués). Pour parer à l'éventualité de deux ex-aequo, on complète toujours cette évaluation de base en y ajoutant un nombre aleatoire entre 0 et 1. Ainsi, il n'y aura jamais d'égalité parfaite et les équipes auront toutes un rang différent.

Par exemple, il est possible que l'équipe précédente ait une évaluation complète égale à 211223.41526784568.

1. Complétez le constructeur donné dans le fichier `equipe.js`. Vous initialiserez l'attribut `evaluation` à une valeur donnée par `Math.random()` (nombre aléatoire entre 0 et 1).

2. Incorporez le fichier `equipe.js` avant `</body>` par

```
<script type="text/javascript" src="js/equipe.js"></script>
```

3. actualisez la page `championnat.html` et testez le constructeur dans la console, par exemple en créant une nouvelle équipe par une instruction comme

```
let eq1 = new Equipe("PSG");
```

puis affichez `eq1` dans la console (`eq1` et `Entrée`). Vous pouvez « déplier » l'objet créé et examiner ce qui s'affiche.

4. Codez la fonction `evaluer()` comme c'est suggéré en page précédente. Testez cette fonction en changeant « à la main » les valeurs de certains attributs de `eq1` (dans la console) et lancez l'exécution de `eq1.evaluer()` dans la console également. Réaffichez `eq1` pour voir le résultat.

5. Codez la fonction `affichage()`, équivalent d'un `toString()` et qui donnera le résumé de l'état actuel de l'équipe. Ci-dessous une succession de commandes lancées dans la console, dont l'affichage final :

```
> var eq1 = new Equipe("PSG");
<- undefined
> eq1.G = 6
<- 6
> eq1.N = 3
<- 3
> eq1.P = 2
<- 2
> eq1.butsPour = 17
<- 17
> eq1.butsContre = 8
<- 8
> eq1.nbPoints = eq1.G * 3 + eq1.N * 1
<- 21
> eq1.affichage()
<- "1   PSG   21   6   3   2   17   8   9"
```

Le numéro 1 devant le nom de l'équipe est son classement.

Pour le moment, ce n'est pas important, mais vous pourrez, plus tard, styliser un peu mieux cet affichage pour que les affichages des huit équipes donnent un rendu de tableau, comme ci-dessous :

rang	équipe	pts	G	N	P	bp	bc	diff
1	EAG	24	8	0	3	23	15	8
2	PSG	19	6	1	4	19	14	5
3	ASSE	17	5	2	4	19	19	0
4	MHSC	17	5	2	4	16	18	-2
5	ASM	15	4	3	4	20	19	1
6	FCN	13	4	1	6	13	14	-1
7	OL	11	3	2	6	15	19	-4
8	OM	10	3	1	7	9	16	-7

6. Codez enfin la fonction `mise_a_jour(bp,bc)` qui met à jour le nombre de points, le nombre de buts pour et contre, le nombre de victoires, nuls et défaites, et qui actualise l'évaluation en lançant la fonction `evaluer()`.

Remarque : les paramètres `bp` et `bc` correspondent bien sûr au résultat d'un match joué par l'équipe, où elle marque `bp` buts et en encaisse `bc`.

7. Testez cette nouvelle fonction à partir de `eq1` qui a certaines valeurs d'attributs suite à vos différentes manœuvres. Par exemple, pour continuer le précédent écran :

```
> var eq1 = new Equipe("PSG");
<- undefined
> eq1.G = 6
<- 6
> eq1.N = 3
<- 3
> eq1.P = 2
<- 2
> eq1.butsPour = 17
<- 17
> eq1.butsContre = 8
<- 8
> eq1.nbPoints = eq1.G * 3 + eq1.N * 1
<- 21
> eq1.affichage()
<- "1   PSG   21   6   3   2   17   8   9"
> eq1.mise_a_jour(3,1)
<- undefined
> eq1.affichage()
<- "1   PSG   24   7   3   2   20   9   11"
```

Remarque : il peut être utile, puisque nous rafraîchissons régulièrement la page, de sauvegarder les commandes à insérer dans la console dans un script, inséré après `equipe.js`, en bas de `championnat.html` et dont le code pourrait être :

```
let eq1 = new Equipe("PSG");
eq1.G = 6;
eq1.N = 3;
eq1.P = 2;
...
```

EXERCICE 2 - l'objet Match

Un Match aura comme attributs :

- son équipe 1 notée `equ1` ;
- son équipe 2 notée `equ2` ;
- le nombre de buts marqués par equ1, qui sera noté `res1`;
- le nombre de buts marqués par equ2, qui sera noté `res2`;
- un booléen `played` qui dit si le match a été joué ou non.

1. Complétez le constructeur donné dans le fichier `match.js`. Vous initialiserez l'attribut `played` à `false`, puisque le match créé n'est pas encore joué. Vous initialiserez les autres attributs de manière sensée.

2. Incorporez le fichier `match.js` à la suite de `equipe.js`.

3. Testez votre constructeur en créant deux équipes, puis un match entre ces deux équipes. Exemple de test :

```
> var eq1 = new Equipe("PSG");
< undefined
> var eq2 = new Equipe("FCN");
< undefined
> var match1 = new Match(eq1,eq2);
< undefined
> match1
< ▼Match {equ1: Equipe, equ2: Equipe, res1: 0, res2: 0, played: false}
    ▶equ1: Equipe {nom: "PSG", classement: 1, nbPoints: 0, G: 0, N: 0, …}
    ▶equ2: Equipe {nom: "FCN", classement: 1, nbPoints: 0, G: 0, N: 0, …}
     played: false
     res1: 0
     res2: 0
```

4. Codez la fonction `jouer()` qui permet de donner des valeurs à `this.res1` et à `this.res2`. En général, même s'il y a bien des exceptions, une équipe qui joue à domicile est légèrement favorisée. A vous de le mettre en œuvre.

Remarques :

- `Math.floor(…)` renvoie la partie entière
- `Math.floor(Math.random()*5)` donne un entier entre 0 et 4.

5. Codez la fonction `maj_equipes()` qui met à jour les attributs des deux équipes du match, à partir des valeurs de `this.res1` et de `this.res2`.

Conseil : réutilisez la méthode `mise_a_jour` de `Equipe`.

6. Codez enfin la fonction `affichage()` qui sera le `toString()` du match et qui produira quelque chose comme ça :

```
> var eq1 = new Equipe("PSG");
< undefined
> var eq2 = new Equipe("FCN");
< undefined
> var match1 = new Match(eq1,eq2);
< undefined
> match1
< ▶Match {equ1: Equipe, equ2: Equipe, res1: 0, res2: 0, played: false}
> match1.jouer()
< undefined
> match1
< ▶Match {equ1: Equipe, equ2: Equipe, res1: 1, res2: 2, played: true}
> match1.affichage()
< "PSG   1 - 2   FCN"
```

EXERCICE 3 - l'objet Journee

Une Journee aura comme attributs :

- son match n°1 noté `match1`;
- son match n°2 noté `match2`;
- son match n°3 noté `match3`;
- son match n°4 noté `match4`;
- un booléen `played` qui dit si la journée a été jouée ou non.

1. Complétez le constructeur donné dans le fichier journee.js. Vous initialiserez l'attribut played à false.

2. Incorporez le fichier journee.js à la suite de match.js.

3. Testez votre constructeur comme précédemment, en créant dans la console 8 équipes, puis 4 matchs, puis 1 journée.

4. Codez la fonction afficher(). Elle est un peu différente des précédents affichages. En effet, l'idée est ici de remplir le innerHTML des 4 du document html qui correspondent aux 4 matchs (repérez-les par leurs identifiants). Chacun de ces div devra refléter l'affichage du match qui lui correspond.

5. Testez votre fonction afficher() dans la console. Elle doit produire un affichage visible (enfin !) dans le navigateur. Vous améliorerez l'esthétique de votre affichage plus tard (avec des balises <table>). L'essentiel n'est pas là.

6. Codez la fonction jouer() qui, si la journée n'est pas encore jouée :
- joue les 4 matchs de la journée ;
- affiche la journée par la méthode précédente ;
- passe le booléen played à true ;
- met à jour chaque équipe grâce à la fonction maj_equipes().

7. Testez cette fonction en la lançant dans la console après avoir testé votre question 5. Vous devez constater l'affichage des nouveaux scores.

EXERCICE 4 - l'objet Championnat

1. Le constructeur de Championnat vous est fourni. Un objet Championnat a trois attributs :
- un tableau d'équipes nommé tabEquipes ;
- en entier numJournee qui dit quelle est la journée active ;
- un tableau journees contenant les 14 journées du planning.

Analysez ce code et comprenez ce qui est fait. Vous remarquerez en particulier les méthodes sur les tableaux. N'oubliez pas d'insérer le fichier championnat.js.

2. Codez la fonction jouer_journee(i) dont l'exécution fera jouer la journée n°i du planning (attention, la journée n°1 du planning correspond à l'élément d'indice 0 du tableau journees de this).

3. Codez la fonction afficher_journee(i), dont l'exécution lancera l'affichage de la journée n°i du planning. Même remarque que pour la question précédente.

4. Codez la fonction afficher_classement(). Pour cela, vous remplirez :
- la d'identifiant titres qui donne les items de chaque colonne, à savoir : nom, points, G, N, P, buts pour, buts contre et différence (buts pour – buts contre)
- les identifiés "1", "2", …, "8". Chacune de ces div recevra l'affichage de l'équipe dont le classement correspond à l'identifiant de la div.

5. Codez la fonction classer_equipes(). Sa mission est de mettre à jour l'attribut classement des 8 équipes après avoir calculé leur évaluation.

Aide 1 : il peut être utile d'avoir à trier par ordre croissant un certain tableau de nombres. Si nous devons trier un tableau t, alors cela peut se faire simplement en JavaScript par :

```
t.sort(function(a,b) {return a-b;})
```

Explication : t.sort() trierait t en considérant ces éléments comme des chaînes de caractères. Le tri serait alphabétique. Autrement dit, 112 serait considéré comme « inférieur » à 13.

Le paramètre donné à la méthode sort permet d'imposer un autre critère de tri. Pour le cas présent, deux éléments a et b seront triés dans l'ordre a < b si la fonction paramètre retourne un résultat négatif. Comme cette fonction a été codée pour retourner a-b, on aura le tri a < b si a-b < 0 ce qui est cohérent avec l'ordre attendu entre les nombres.

Retenez que l'instruction t.sort(function(a,b) {return a-b;}) permet de trier le tableau de nombres t par ordre croissant.

De même, l'instruction t.sort(function(a,b) {return b-a;}) permet de trier un tableau de nombres par ordre décroissant. Ceci peut servir...

Aide 2 : t.indexOf(num) retourne l'indice de num dans le tableau t.

Avec tout ceci, vous devriez vous en tirer.

6. Il serait bon, de nouveau, de tester tout ça dans la console. Voici un exemple de code à insérer après tous les fichiers, et qui peut vous aider :

```
<script type="text/javascript">
  let eq1 = new Equipe("PSG");
  let eq2 = new Equipe("FCN");
  let eq3 = new Equipe("ASM");
  let eq4 = new Equipe("RCS");
  let eq5 = new Equipe("HAC");
  let eq6 = new Equipe("RCL");
  let eq7 = new Equipe("TFC");
  let eq8 = new Equipe("EAG");
  let chp = new Championnat(eq1,eq2,eq3,eq4,eq5,eq6,eq7,eq8);
</script>
```

Entrez ensuite dans la console les instructions suivantes :

```
chp.classer_equipes();
chp.afficher_classement();
chp.afficher_journee(1);
chp.jouer_journee(1);
chp.classer_equipes();
chp.afficher_classement();
chp.afficher_journee(2);
chp.jouer_journee(2);
chp.classer_equipes();
chp.afficher_classement();
```

etc

7. Il pourrait être agréable d'avoir une disposition de table au niveau de la `<div id="titres">` et des diverses `<div id="1">, ..., <div id="8">` pour avoir un bon affichage du classement (voir image plus haut). Si vous avez le temps, c'est le moment. C'est possible en incluant « brutalement » les balises adéquates au niveau des divers `innerHTML` rencontrés.

EXERCICE 5 - le scénario

Et maintenant, le scénario du déroulement du jeu. Vous pouvez supprimer les instructions tests utilisées précédemment (voir haut de cette page). Incluez le fichier `scenario.js` à la suite des 4 autres fichiers. Ce fichier sera une suite d'instructions. Pour le moment, il contient des déclarations de variables :

- des variables qui font le lien avec l'interface html ;
- d'autres variables comme tabEq et chp;

tabEq sert à remplir les input et la liste des équipes engagées avec des valeurs par défaut, et chp prendra comme valeur un Championnat (plus tard).

État initial

1. Au début, certains éléments seront en display : inline, d'autres en display : none. Tous ces styles sont amenés à passer d'un état à l'autre en fonctions d'événements clic divers, et ceci restera à programmer.

Rappel: on peut accéder au display d'un élément elt par

```
elt.style.display = "…"
```

Programmez l'état d'affichage initial suivant :

- les fieldsetJ et fieldsetC sont en display : none.
- le fieldsetE doit être en display : inline.
- Parmi les enfants du fieldsteE :
  → la sera en display : inline.
  → la sera en display : none.

- Dans la balise legend du fieldsetE, seule l'image « plus » est affichée, l'autre est en display : none.

- Quand on lancera le championnat, le fieldsetC deviendra visible. Faites en sorte que dès à présent, le bouton journee_suivante soit caché, contrairement au bouton jouer_journee. Passez le display de jj à inline et celui de js à none.

2. Il faut aussi préremplir les input des noms d'équipes en cohérence avec la liste des équipes engagées par défaut. Autrement dit, il faut remplir ces input avec les valeurs du tableau tabEq. Programmez ceci.

3. Enfin, pour compléter votre état initial, affectez au innerHTML de ee la valeur texte suivante :

```
"équipes engagées : PSG - ASM - OL - OM - FCN - ASSE - MHSC - EAG"
```

Programmez tout ceci et testez en rafraîchissant la page

Gestion des événements click

4. On peut programmer la gestion d'un événement click de plusieurs façons (voir TD1). Dans notre cas, nous allons utiliser des « fonctions anonymes » :

```
pl.onclick = function() {
  pl.style.display = "none";
  mo.style.display = "inline";
  le.style.display = "inline";
  ee.style.display = "none";
}
```

Dans le code précédent, on affecte à l'attribut `onclick` de `pl` une valeur de type fonction, qui ne porte pas de nom particulier, et dont le contenu permet d'agir sur le `display` d'éléments. Un code équivalent aurait été :

```
function reaction_au_clic_pl() {
  pl.style.display = "none";
  mo.style.display = "inline";
  le.style.display = "inline";
  ee.style.display = "none";
}

pl.onclick =  reaction_au_clic_pl ;
```

Mais comme cette fonction ne sert que là, on peut la passer en fonction anonyme sans problème. Recopiez ce code, anticipez ce qui se passera quand on cliquera sur l'image `plus`. Actualisez la page et vérifiez que le comportement attendu se produit bien.

5. On va maintenant gérer le clic sur l'image « moins ». Adaptez le code précédent pour programmer le comportement suivant, quand on clique sur le moins :

- l'image « plus » doit réapparaître;
- l'image « moins » doit disparaître;
- la doit disparaître ;
- le `innerHTML` de la `<div id="equipesEngagees">` est recalculé pour afficher le même type de phrase que celle par défaut, mais cette fois ce sont les valeurs des `input` qui serviront ;
- cette `<div id="equipesEngagees">` doit apparaître ;

Programmez tout ceci, actualisez la page et vérifiez que les comportements attendus sont opérationnels.

6. Programmons l'action du bouton `lancer_championnat`. Celui-ci doit :

- appeler le constructeur de `Championnat`, les paramètres étant huit équipes construites à partir des 8 champs texte du `fieldsetE`. Vous utiliserez la variable globale `chp` déclarée au début :

```
chp = new Championnat(…);
```

- classer ces équipes ;
- afficher le classement ;
- actualiser la balise <legend id="numJ"> pour que son contenu soit de la forme « journée n°… » (avec l'attribut numJournee) ;
- afficher la journée correspondant à numJournee (c'est-à-dire afficher les 4 matchs) ;
- passer jj en display : inline et js en display : none ;
- passer fj et fc en display : inline, fe en display : none ;
- passer lc en display : none (pour éviter les relances maladroites du championnat)

Essayez de faire tout ça dans une fonction anonyme !

7. Passons à l'action du bouton jouer_journee. Celui-ci doit :

- faire jouer la journée d'indice numJournee ;
- passer jj en display : none ;
- si numJournee est inférieur à 14, passer js en display : inline (sinon, le championnat est terminé!) ;
- classer les équipes ;
- afficher le classement.

Pareil, fonction anonyme.

8. Et pour finir, l'action de journee_suivante. Celui-ci doit :

- augmenter numJournee d'une unité ;
- afficher la journée correspondant à cette nouvelle valeur de numJournee ;
- mettre à jour le contenu de la balise numJ ;
- passser jj en display : inline et js en display : none.

Pareil, fonction anonyme.

Actualisez tout ça, et jouez !