TP 1

Programmation JavaScript

Ce TP est un TP JavaScript « pur ». À l'exception d'une page HTML au contenu vide, aucun code HTML ne sera demandé. Évidemment, aucun code CSS n'est demandé au cours de ce TP.

Pour tout ce TP, on créera une page HTML minimale (c'est-à-dire contenant le minimum de contenu pour être un code HTML valide), par exemple `TPJavaScriptPur.html`, liée à un fichier externe, par exemple `TPJavaScriptPur.js`. Il suffit donc de lancer `TPJavaScriptPur.html` pour exécuter le JavaScript. Ne pas oublier de mettre quelques `console.log()` et observer le résultat dans la console du navigateur (voir le cours).

Sous Firefox, la console s'affiche avec F12, ou avec CTR+Maj+K. Sous Chrome ou Chromium : CTR+Maj+I. Sous Edge : CTR+Maj+J.

EXERCICE

Exercice 1.1 (Écriture de fonctions simples)**.**

Question 1 : Écrire une fonction `minimum(a, b)` qui retourne le minimum de 2 nombres, et une fonction `maximum(a, b)` qui retourne le maximum de 2 nombres. La tester avec quelques exemples en utilisant `console.log()`.

Exemple de tests :

```
1 console.log("minimum(75,93) = " + minimum(75,93)); // affiche
    minimum(75,93) = 75
2 console.log("minimum(54,35) = " + minimum(54,35)); // affiche
    minimum(54,35) = 35
3 console.log("maximum(75,93) = " + maximum(75,93)); // affiche
    maximum(75,93) = 93
```

Question 2 : Écrire une fonction `afficheNombres()` qui affiche (en utilisant `console.log()`) tous les nombres de 1 à 100 ; de plus, pour les nombres divisibles par 3, afficher « `divisible par 3` » ; pour les nombres divisibles par 5, afficher « `divisible par 5` », et pour le nombre 93, afficher « `Seine-Saint-Denis !` ».

Premières lignes du résultat attendu :

```
1
2
3
divisible par 3
4
5
divisible par 5
```

6

...

Question 3 : Écrire une fonction `puissance(x, n)` qui retourne x^n. Le calcul devra être fait en utilisant une boucle.

Question 4 : Écrire une fonction `puissancerec(x, n)` qui retourne x^n. Le calcul devra être fait en utilisant une récursion.

■

EXERCICE

Exercice 1.2 (Écriture de fonctions complexes)**.**

Question 1 : Écrire une fonction `creerMultiplicateur(n)` qui reçoit un paramètre entier `n` et retourne une fonction qui multiplie son paramètre `x` par `n`.

Exemple d'appel :

```
// Exemple 1
let multipli4 = creerMultiplicateur(4);
console.log("Test : creerMultiplicateur(4)(3) = " + multipli4(3)); //
    affiche 12
// Exemple 2
console.log("Test : creerMultiplicateur(93)(22) = " +
    creerMultiplicateur(93)(22)); // affiche 2046
```

Question 2 : Écrire une fonction `creerSequence(init, step)` qui reçoit 2 paramètres : une valeur initiale `init` et une valeur d'incrément `step`, et retourne une fonction qui délivre à chaque appel les valeurs successives de la séquence démarrant à `init` et incrémentées de `step`.

Exemple d'appel :

```
let seq43 = creerSequence(4, 3); // une première fonction séquence
let seq21 = creerSequence(2, 1); // une deuxième fonction séquence
console.log("Test : creerSequence(4, 3) appel 1 : " + seq43()); // 4
console.log("Test : creerSequence(2, 1) appel 1 : " + seq21()); // 2
console.log("Test : creerSequence(4, 3) appel 2 : " + seq43()); // 7
console.log("Test : creerSequence(4, 3) appel 3 : " + seq43()); // 10
console.log("Test : creerSequence(2, 1) appel 2 : " + seq21()); // 3
```

Question 3 : Écrire une fonction `fibonacci(u0, u1)` qui permet de parcourir la suite de Fibonacci. La fonction reçoit 2 arguments qui sont les 2 valeurs initiales de la suite, et retourne une fonction qui, à chaque appel, délivre les valeurs successives de la suite. (Rappel : la suite de Fibonacci est la suite $u_n = u_{n-1} + u_{n-2}$.)

Exemple d'appel :

```
let fibolapins = fibonacci(1, 1);
console.log("fibolapins appel 1 : " + fibolapins()); // affiche 2
console.log("fibolapins appel 2 : " + fibolapins()); // affiche 3
console.log("fibolapins appel 3 : " + fibolapins()); // affiche 5
console.log("fibolapins appel 4 : " + fibolapins()); // affiche 8
```

Question 4 : Créer une variante (nommée `creerMultiplicateur2`) de la fonction `creerMultiplicateur` pour que, dans le cas où elle reçoit deux paramètres `n` et `x`, elle retourne `n×x`, et, dans le cas où elle reçoit un seul paramètre, elle retourne la fonction qui multiplie son paramètre par `n`.

Exemple d'appel :

```
let multipli5 = creerMultiplicateur2(5);
console.log("creerMultiplicateur(5)(2) = " + multipli5(2)); // affiche 10
console.log("creerMultiplicateur2(3,4) = " + creerMultiplicateur2(3,4));
    // affiche 12
```

Question 5 : Écrire une fonction `formatter(numero_initial)` qui construit une fonction de formatage de messages en ajoutant un numéro de message (initialement : `numero_initial`) incrémenté à chaque appel. On pourra l'utiliser ainsi :

```
let f = formatter(10);            // on numérote à partir de 10
console.log(f('il fait beau'));   // affiche "10: il fait beau"
console.log(f('il fait chaud')); // affiche "11: il fait chaud"
```

Question 6 : Écrire une fonction `logger`, qui reçoit en paramètre une fonction de formatage et une fonction d'écriture de messages, et retourne une fonction de log qui reçoit un message en paramètre, le formate et l'écrit avec les fonctions passées en paramètres.

Exemple d'appel :

```
let nb = 1;
let monlogger = logger(
    (chaine) => "message " + nb++ + " : " + chaine,
    console.log);
monlogger('coucou');  // Affiche sur la console 'message 1 : coucou'
monlogger('ça va ?'); // Affiche sur la console 'message 2 : ça va ?'
```

■

EXERCICE

Exercice 1.3 (Tableaux)**.**

Question 1 : Écrire la fonction `range(a,b)` qui reçoit deux paramètres entiers et retourne un tableau contenant tous les entiers de a à b (si $a \leq b$) ou de b à a (si $a > b$).

Question (optionnelle) 2 : S'assurer que la fonction `range(a,b)` ne contient *aucun* test conditionnel (syntaxe `if`).

Question 3 : Écrire une fonction `somme` qui reçoit un tableau d'entiers et retourne la somme des éléments. Écrire une version utilisant une boucle `for`, puis une version utilisant une boucle `for ... of`, puis une version utilisant la construction `forEach()`.

Question 4 : Écrire une fonction `moyenne` qui reçoit un tableau d'entier et retourne la moyenne des éléments. Utiliser la construction `forEach()`.

Question 5 : Écrire une fonction `selectionne` qui reçoit un tableau de chaînes de caractères `t` et une chaine de caractères `pattern`, et qui retourne un tableau composé des chaînes de `t` qui comportent le motif `pattern`, mis en majuscules.

Exemple d'appel :

```
console.log(selectionne(["ile-de-france", "galilee", "universite"],
    "ile")); // affiche ["ILE-DE-FRANCE", "GALILEE"]
```

Note : on peut tester l'inclusion entre deux chaînes avec la fonction suivante :

```
bottedefoin.includes(aiguille); // rend vrai si bottedefoin inclut
    aiguille
```

■

EXERCICE

Exercice 1.4 (Objets)**.**

Question 1 : Consulter la documentation JavaScript sur l'objet `Date` pour écrire une fonction `semaine()` qui retourne un message dépendant du jour courant :

- `"Dur le lundi !"` si on est lundi ;
- `"Youhou, c'est le week-end !"` si on est samedi ou dimanche ;
- `"Bientôt le week-end"` si on est vendredi ;
- `"Au boulot !"` si on est un autre jour de la semaine.

Question 2 : Écrire une fonction `statsTableau()` qui reçoit un tableau d'entiers, et retourne un objet contenant 3 propriétés : le nombre d'éléments du tableau, la somme des éléments et la moyenne des éléments. Penser à utiliser les fonctions réalisées dans l'exercice réalisé plus tôt dans ce TP.

Exemple d'appel :

```
let tableau = [12, 11, 15, 9, 10, 15];
console.log(statsTableau(tableau));
// Affiche : Object { nombre: 6, moyenne: 12, somme: 72 }
```

■

EXERCICE

Exercice 1.5 (Création d'une classe)**.**

Question 1 : Créer une classe `Eleve` : cette classe doit produire des objets avec les propriétés suivantes :

- numéro (champ `numero`),
- nom de famille (champ `nom`),
- prénom (champ `prenom`),
- date de naissance (champ `ddn`),
- adresse électronique (champ `courriel`),
- liste de notes (champ `notes`).

La valeur de la propriété `ddn` est construite avec le constructeur de dates. La valeur de la propriété `notes` est initialement un tableau vide.

Question 2 : Ajouter une méthode `age` qui calcule l'âge de l'élève. Ajouter une méthode `affiche` qui affiche les nom, prénom et date de naissance de l'élève. Le nom doit s'afficher en majuscules ; la date doit s'afficher sous la forme `jj/mm/aaaa`.

Question 3 : Ajouter une méthode `ajoutNote` qui ajoute à l'élève une note dans une matière. La méthode reçoit deux paramètres `matiere` (matière) et `note` (note), en fait un objet contenant deux propriétés, et ajoute cet objet dans le tableau de notes de l'élève; on suppose que l'on a toujours une seule note par matière.

Question 4 : Ajouter une méthode `moyenne` qui calcule la moyenne de l'élève (toutes les matières ont le coefficient 1).

Question 5 : Créer une méthode statique `annivMois` dans la classe `eleve` qui reçoit un tableau d'élèves et un numéro de mois, et retourne un tableau contenant la liste des élèves dont l'anniversaire a lieu durant le mois indiqué. Créer quelques élèves et tester cette méthode.

■

Version du TP : 29 avril 2024