Javascript (1) : Survol du langage

Cette première série d'exercices sur Javascript a pour objectif de vous familiariser avec la syntaxe du langage tout en vous proposant quelques (ré)visions d'algorithmique.
Pour chaque concept évoqué (variables, expressions booléennes, alternatives, boucles), un ou plusieurs exemples vous sont donnés. Vous êtes invités à les tester et à les comprendre. Puis, vous avez plusieurs exercices du même acabit à résoudre.
N'hésitez pas à demander de l'aide aux chargés de TP présents dans la salle. Ne restez pas bloqués sur un problème.

Exercice 1 : WebStorm et JavaScript (1)

Ce premier TP sur JavaScript vise à découvrir le langage et sa syntaxe. Pour le moment, il n'est pas (encore) question d'interagir avec une page web. Vos programmes JavaScript seront ainsi exécutés, non pas dans le navigateur, mais dans l'environnement d'exécution fourni par WebStorm.

Nous allons commencer par un exemple très simple pour voir comment cela fonctionne.

a. Lancez l'application WebStorm

b. Dans le menu `File`, choisissez `New` puis dans le nouveau menu qui apparaît, sélectionnez `JavaScript file`.

c. Une boîte de dialogue s'ouvre qui vous demande le nom du fichier que vous voulez créer, saisissez ici `exercice0.js`.

d. Saisissez l'instruction : `console.log("Hello World!")`

e. Dans le menu `Run` choisissez `Run` puis, dans le menu qui s'ouvre, `exercice0.js`.
Ceci déclenche l'exécution du code Javascript écrit dans le fichier `exercice0.js`.
Vous devez maintenant voir apparaître en bas de votre fenêtre une console dans laquelle s'affiche la phrase "Hello World !" (sans les guillemets bien sûr).

f. Remplacez maintenant, dans le fichier `exercice0.js`, l'instruction précédente par le code suivant :

```
function display(message){
    console.log(message);
}
```

Ce code définit une fonction appelée `display`, qui prend un paramètre nommé `message` et qui, lorsqu'elle est appelée, affiche le contenu du paramètre `message` sur la console.
Exécutez le fichier `exercice0.js` comme précédemment. Un message s'affiche-t-il sur la console ? Comprenez-vous pourquoi ? Si vous ne comprenez pas, posez la question à un des encadrants de TP.

g. Ajoutez dans votre fichier après la définition de la fonction, l'instruction suivante :
`display("Hello World !")`
Exécutez à nouveau. Un message s'affiche-t-il sur la console ? Comprenez-vous pourquoi ?

h. Enfin, ajoutez les lignes suivantes, puis exécutez :

```
display(2);
display(2 < 3);
display(new Date());
```

Ceci pour vous montrer que JavaScript ne se soucie pas du type des valeurs, ce qui peut poser un certain nombre de problèmes sur lesquels nous reviendrons plus tard.

Dans les exercices suivants, chaque fois qu'il vous sera demandé d'afficher quelque chose dans la console de WebStorm, vous utiliserez, comme ici, l'instruction `console.log`.
De même lorsqu'on vous demandera d'exécuter votre ficher JavaScript vous utiliserez la commande `Run` du menu `Run`, en choisissant évidemment le bon fichier à exécuter.

Exercice 2 : Variables

a. *On vous prend par la main*

Recopiez le code suivant qui permet de créer une variable dont le nom est "unentier" et de lui affecter la valeur 3 :

```
let unentier = 3;
```

Saisissez ensuite :

```
console.log(unentier);
```

Exécutez votre programme.
L'expression "unentier" est évaluée et sa valeur (3) affichée dans la console de WebStorm.
Quel est le type de cette variable ?

```
console.log(typeof unentier);
```

b. *À vous de jouer :*

- déclarez une variable `a` sans l'initialiser. En réalisant les affichages utiles, répondez aux questions suivantes :
  - Quelle est la valeur de `a` ?
  - Quel est son type ?
- affectez la valeur 42 à `a`. Quel est maintenant le type de `a` ?
- créez une autre variable `b` égale à 24.
- écrivez la suite d'instructions qui permet d'échanger le contenu des variables a et b. Vérifiez que a et b ont bien les valeurs attendues après l'échange.
- créez une variable `c` qui contienne la somme de `a` et `b`. Vérifiez son type et sa valeur.
- affectez maintenant dans `c` le résultat de la soustraction de `a` et `b`. Vérifiez son type et sa valeur.
- créez une variable `s1` qui contienne *"Bonjour"*. Quel est son type ?
- créez une variable `s2` qui contienne votre prénom. Créez une variable `s3`, et affectez-lui le résultat de la concaténation des 2 chaînes de caractères. Quel est le type de `s3` ?
- Dans une nouvelle variable `s4`, réalisez une soustraction entre `s1` et `s2`. Quel résultat obtenez-vous ? De quel type est la variable `s4` ?
- Déclarez une nouvelle variable `car1` et affectez-lui la valeur *'a'*. De quel type est la variable `car1` [1] ?

1. Il n'existe pas en Javascript de type spécifique pour coder un caractère. Pour délimiter une chaîne de caractères, on utilise indifféremment des apostrophes et des guillemets.

- Les instructions suivantes déclarent une variable `d0`, lui affectent comme valeur un "objet" Date mémorisant la date de naissance de Tim Berners Lee[2], puis affichent une chaîne de caractères représentant cette date dans la console.

```
let d0 = new Date(1955,5,8);

console.log(d0.toString());
```

  - En regardant, le code essayez de deviner quel jour est né Tim Berners Lee.
  - Est-ce que l'affichage provoqué par l'exécution du script corrobore votre intuition ?
  - Expliquez à quoi correspondent les 3 paramètres utilisés pour créer un objet `Date`.
  - Déclarez une nouvelle variable `d1` et affectez-lui la date de votre naissance.
  - Créez une nouvelle variable `d2` qui contienne la date du jour. Réalisez "l'addition" de `d1` et `d2`. Affichez le résultat. Expliquez-le. Ajoutez maintenant un nombre entier à l'une de vos dates. Affichez le résultat. Expliquez-le... ou demandez à un chargé de TP si vous ne comprenez pas...

Exercice 3 : Expressions booléennes[3]

a. *On vous prend par la main*

- La fonction définie ci-dessous renvoie `true`, autrement dit "vrai" si la valeur passée en paramètre est strictement plus grande que la valeur 2, et `false` (="faux") sinon.

```
function isGreaterThan2(a){
  return (a > 2);
}
```

Recopiez le code précédent puis affichez le résultat de chacun des appels de fonction ci-dessous :

```
isGreaterThan2(3);
isGreaterThan2(1);
isGreaterThan2(2.5);
```

- La fonction définie ci-dessous renvoie `true`, autrement dit "vrai" si la valeur passée en paramètre est située après 'A' et avant 'Z' dans l'ordre lexicographique (autrement dit, l'ordre du dictionnaire), et `faux` sinon.

```
function isBeginningWithUpperCase(valeur){
  return (valeur >= 'A') && (valeur <= 'Z');
}
```

Recopiez le code précédent puis affichez le résultat de chacun des appels de fonction ci-dessous :

```
isBeginningWithUpperCase('E');
isBeginningWithUpperCase('a');
isBeginningWithUpperCase(3);
```

D'après vous que donne l'évaluation de l'instruction ci-dessous :

```
isBeginningWithUpperCase("Bonjour");
```

2. L'inventeur du web au cas où vous auriez oublié.

3. Rappel : une expression booléenne est une expression susceptible de prendre deux valeurs, vrai (*true*) ou faux (*false*)

Affichez le résultat dans la console.

Si l'on souhaite maintenant écrire une fonction retournant `true` si le paramètre est un caractère en majuscule et `false` sinon, il faut ajouter des vérifications supplémentaires au code précédent pour s'assurer que la valeur passée en paramètre est bien un caractère. Par exemple, on peut d'abord vérifier que le paramètre est bien une chaîne de caractères de longueur 1 (donc un caractère) et ensuite réaliser la comparaison.

```
function isUppercase(lettre){
  return (typeof lettre == "string")
         && (lettre.length == 1)
         && (lettre >= 'A')
         && (lettre <= 'Z');
}
```

Recopiez et évaluez le code précédent puis réalisez des tests en appelant cette fonction avec différents types de paramètres : des lettres majuscules ou non, des chaînes de caractères, des nombres...

b. À vous de jouer

1. (*) Écrivez la fonction javascript `isBetween(a,min,max)` qui n'utilise aucune alternative, autrement dit aucun test, et qui renvoie vrai si la valeur du paramètre `a` est dans l'intervalle $[min, max]$.
   Affichez le résultat des appels de fonctions ci-dessous et expliquez les affichages obtenus :
   - isBetween(5,1,4) ;
   - isBetween(3,1,4) ;
   - isBetween('e','a','z') ;
   - isBetween('E','a','z') ;
   - isBetween('.','A','z') ;
   - isBetween('[', 'A', 'z') ;
   - isBetween('g','A','z') ;
   - isBetween('e', 'a', 'Z') ;
   - isBetween('chien', 'chat', 'souris') ;
   - isBetween('chat', 'chien', 'souris') ;

   En faisant appel à la fonction précédente, écrivez les fonctions `isLetter(a)`, `isDigit(a)` et `isAlphaNumeric(a)` qui renvoient vrai si a est respectivement une lettre (minuscule ou majuscule) non accentuée, un chiffre entre 0 et 9, ou l'un ou l'autre.
   Affichez les résultats des instructions ci-dessous. Est-ce que leur résultat correspond à ce que vous attendiez ?
   - isDigit(3) ;
   - isDigit(45) ;
   - isDigit("3") ;
   - isDigit("34") ;
   - isLetter("a") ;
   - isLetter("ab") ;
2. (**) Écrivez la fonction `isLeapYear(year)` qui prend en paramètre une année et retourne vrai si elle est bissextile, faux sinon. Pour mémoire, une année est bissextile si elle est multiple de 400 ou bien multiple de 4 et pas de 100. Cette fonction sera réalisée sans test avec une expression booléenne.

3. (***) Écrivez la fonction `belongsToRectangle(x,y,a1,b1,a2,b2)` qui renvoie vrai si le point (x, y) appartient au rectangle dont le coin supérieur gauche (resp. le coin inférieur droit) a comme coordonnées $(a1, b1)$ (resp. $(a2, b2)$).

Exercice 4 : Conditionnelles

a. `if-then-else`

1. *On vous prend par la main*

   La fonction ci-dessous prend en paramètre un prix et renvoie un nouveau prix calculé avec une remise. Si le prix initial est supérieur à 1000 euros, la remise est de 40%, s'il est compris entre 500 et 1000 euros, elle est de 20%, entre 100 et 500 euros de 10%. Au-dessous de 100 euros, aucune remise n'est pratiquée.

```
function computeDiscount(price){
  let newPrice;

  if (typeof price != "number"){
    return -1; // ERREUR
  }

  if (price >= 1000){
    newPrice = price * 0.6;
  }
  else if (price >= 500){
    newPrice = price * 0.8;
  }
  else if (price >= 100){
    newPrice = price * 0.9;
  }
  else{
    newPrice = price;
  }

  return newPrice;
}
```

   Recopiez cette fonction et appelez-la avec diverses valeurs (des nombres, des chaînes de caractères, etc.) pour observer son comportement.

2. *À vous de jouer*

   i. (*) En utilisant des `if`, écrivez une fonction `getMaximum(a,b,c)` qui renvoie le maximum des 3 nombres passés en paramètre.

   ii. (**) Écrivez, en utilisant des alternatives, la fonction `nextSecond(h,m,s)` qui renvoie sous forme d'une chaine de caractères, l'heure qu'il sera dans 1 seconde.

   Par exemple :

   `nextSecond(14,23,4)` renverra la chaîne de caractères "14 :23 :5",

   `nextSecond(14,23,59)`, la chaîne "14 :24 :00"

   et `nextSecond(23,59,59)`, la chaîne "00 :00 :00".

b. `switch-case`

1. *On vous prend par la main*

   La fonction ci-dessous renvoie le nom de la direction ("NORTH", "EAST", "SOUTH", "WEST") dont l'initiale est passée en paramètre (en majuscule ou minuscule) à la fonction.

```
function giveDirection(d){
  let direction;
  switch (d){
    case 'n' :
    case 'N' : direction = "NORTH"; break;
    case 'e' :
    case 'E' : direction = "EAST"; break;
    case 's' :
    case 'S' : direction = "SOUTH"; break;
    case 'w' :
    case 'W' : direction = "WEST"; break;
    default : direction = "ERROR";
  }
  return direction;
}
```

Recopiez cette fonction et appelez-la avec diverses valeurs (caractères, nombres, etc.) pour observer son comportement.

2. *À vous de jouer*
   i. (*) Écrivez une fonction `getMonthName(m)` qui renvoie le nom en toutes lettres et en anglais du mois dont le numéro est passé en paramètre. Si le paramètre ne correspond pas au numéro d'un mois, une chaîne de caractères vide est renvoyée.
   ii. (*) Écrivez une fonction `getNumberOfDays(m,b)` qui prend en paramètre un numéro de mois (m) et un booléen (b) indiquant si l'année considérée est bissextile (alors b vaut true), ou non (alors b vaut false) et renvoie le nombre de jours que contient le mois m. Si le mois passé en paramètre est incorrect, la fonction renvoie -1.
   iii. (**) Pour envoyer une lettre pesant au plus 20 grammes, vous disposez de 3 tarifs différents. Le tarif "économique" à 0,68 euros, le tarif "vert" à 0,70 euros, et le tarif "normal" à 0,80 euros. Écrivez la fonction `stampCosts(quantite, code)` qui renvoie le coût du nombre de timbres passé en premier paramètre au tarif passé en second paramètre. Ce deuxième paramètre est un chiffre, 0 correspond à économique, 1 à vert et 2 à normal.

Exercice 5 : Boucles

a. *On vous prend par la main*

La suite de Syracuse d'un nombre entier N est construite par récurrence de la manière suivante :

$u_0 = N$

si u_k est pair, alors $u_{k+1} = \frac{u_k}{2}$

si u_k est impair, alors $u_{k+1} = 3 * u_k + 1$

La fonction ci-dessous prend en paramètre la valeur initiale de la suite et le numéro k du terme que l'on veut calculer et renvoie le $k^{ème}$ terme de la suite.

```
function computeSyracuse(init, k){
  let u = init;
  for (var i = 1 ; i <= k; i++){
    if (u % 2 == 0){
      u = u / 2;
    }
    else{
      u = 3 * u + 1;
    }
```

```
  }
  return u;
}
```

Recopiez et testez cette fonction.

Une conjecture affirme que, quelle que soit la valeur initiale de la suite, il existe un terme de la suite qui est égal à 1. L'indice du premier terme de la suite égal à 1 est appelé le temps de vol de la suite. La fonction ci-dessous renvoie le temps de vol de la suite, dont la valeur initiale est passée en paramètre.

```
function computeSyracuseFlight(init){
  let u = init;
  let cpt = 0;
  while (u > 1){
    if (u % 2 == 0){
      u = u / 2;
    }
    else{
      u = 3 * u + 1;
    }
    cpt = cpt + 1;
  }
  return cpt;
}
```

Recopiez et testez cette fonction.

b. *À vous de jouer*

1. (*) Écrivez une fonction `getNumberOfDigits(a)` qui, dans une boucle `while`, divise la variable a par 10 jusqu'à ce qu'elle atteigne une valeur strictement inférieure à 1. La fonction renvoie le nombre d'itérations qui ont été effectuées pour atteindre ce résultat.
2. (*) Écrivez une fonction `integerSum(a,b)` qui, en utilisant une boucle pour, calcule et renvoie la somme des entiers entre a et b.
3. (**) Écrivez deux fonctions qui prennent en paramètre un nombre n et renvoient la somme des n premiers termes de la série harmonique. La première calcule $1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}$. La seconde $\frac{1}{n}+\frac{1}{n-1}+\ldots+\frac{1}{2}+1$ [4].
   Testez ces deux fonctions avec les valeurs 4, 40, 400, 4000, 40000. Que constatez-vous ? Devinez-vous pourquoi ? [5]

Exercice 6 : Tableaux indexés par des entiers

Dans aucune des questions ci-dessous, on n'utilisera de tableaux auxiliaires.

a. *On vous prend par la main*

L'instruction ci-dessous permet de déclarer et initialiser un tableau de taille 4 contenant dans ses cases numérotées de 0 à 3 les valeurs : 10, 20, 30, et 40. Recopiez-la et évaluez-la.

```
let tab = [10,20,30,40];
```

L'instruction ci-dessous permet de modifier la première case du tableau, testez-la :

```
tab[0] = 100;
```

4. Oui, d'un point de vue mathématique, on dirait bien qu'on calcule la même chose, mais...
5. Si vous n'êtes pas sûrs, demandez à un des chargés de TP.

Modifiez la valeur de la deuxième case en y mettant la valeur 200. Puis modifiez la valeur de la troisième case en y mettant la chaîne de caractères : "Bonjour".

En javascript, un tableau peut contenir des valeurs de différents types...

Javascript fournit plusieurs méthodes pour manipuler les tableaux (cf l'aide-mémoire fourni sur UPdago). Parmi elles, la méthode `join` crée une chaîne de caractères à partir du contenu du tableau et peut être utilisée de la manière suivante :

```
tab.join();
```

Recopiez et évaluez l'instruction précédente.

Essayez d'accéder à la valeur de la $5^{ème}$ case du tableau :

```
tab[4];
```

A priori, elle est "undefined".

Essayez maintenant d'affecter la valeur de votre choix à cette case puis utilisez la méthode `join` vue précédemment pour afficher ce que contient le tableau.

Continuons « vers l'infini et au-delà »[6] et essayez maintenant d'affecter une valeur de votre choix à la case numéro 200.

Affichez le tableau.

Que s'est-il passé ?

La fonction ci-dessous calcule la valeur maximum d'un tableau :

```
function computeMax(t){
  let max = t[0];
  for (let i = 1 ; i < t.length ; i++){
    if (t[i] > max){
       max = t[i];
    }
  }
  return max;
}
```

Recopiez et évaluez la fonction puis testez-la avec plusieurs tableaux, l'un contenant des nombres, un autre des chaînes de caractères, un troisième des valeurs de différents types.

b. *À vous de jouer*

1. Écrivez une procédure `function turnLeft(tab)` qui, en utilisant une boucle pour, fait "tourner" les valeurs contenues dans un tableau passé en paramètre d'un cran dans le sens des indices décroissants. Autrement dit, si le tableau contient initialement les valeurs $1, 2, 3, 4, 5$, il contiendra après application de la procédure les valeurs $2, 3, 4, 5, 1$.
2. Écrivez une procédure `function turnRight(tab)` qui, en utilisant une boucle pour, fait "tourner" les valeurs contenues dans un tableau passé en paramètre d'un cran dans le sens des indices croissants. Autrement dit, si le tableau contient initialement les valeurs $1, 2, 3, 4, 5$, il contiendra après application de la procédure les valeurs $5, 1, 2, 3, 4$
3. Écrivez une procédure `function permute(tab,k)` qui, en utilisant les deux procédures précédemment écrites, fait tourner les valeurs du tableau tab, k fois vers la droite lorsque k est positif et $-k$ fois vers la gauche lorsque k est négatif.

Exercice 7 : Objets

6. « Toy story », vous connaissez ?

a. *On vous prend par la main*

En javascript, outre les types primitifs et les tableaux indexés par des entiers, on manipule ce qu'on appelle des objets et qui sont des tableaux associatifs. Concrètement, les indices de ces tableaux ne sont pas des entiers mais des chaînes de caractères.

L'instruction ci-dessous définit la variable minmax comme un tableau comportant une case "min" ayant pour valeur 0 et une case "max" ayant pour valeur 10.

```
let minmax = {min : 0, max : 10};
```

Pour accéder aux éléments de l'objet `minmax`, on peut procéder de plusieurs manières. Recopiez et testez une à une les instructions ci-dessous :

```
console.log(minmax["min"]);

console.log(minmax["max"]);

console.log(minmax.min);

console.log(minmax.max);
```

On peut aussi itérer dans ces objets (comme on itère dans un tableau) en utilisant une version particulière de la boucle pour :

`for (indice in object) {... manipuler object[indice] ... }`

Recopiez et testez l'exemple ci-dessous :

```
let HermioneGrangerResults ={
  defenceAgainstDarkArts : 19,
  potions : 20,
  charms : 20,
  careOfMagicalCreatures : 19
};

function computeAverage(results){
  let s = 0;
  let nbSubjects = 0;
  for (let subject in results){
    s = s + results[subject];
    nbSubjects = nbSubjects + 1;
  }
  s = s / nbSubjects;
  return s;
}

computeAverage(HermioneGrangerResults);
```

b. *À vous de jouer*

1. En vous inspirant de l'exemple précédent, écrivez une fonction `getObjectProperties(o)` qui renvoie une chaîne de caractères composée d'autant de lignes que de propriétés de l'objet avec pour chaque propriété, son nom et sa valeur.
2. Pour tester votre fonction, affichez la chaîne de caractères renvoyée lorsqu'elle est appelée sur l'objet `minmax` et sur l'objet `HermioneGrangerResults` précédemment créés.