ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

TP Série 2 : Fonctions, Fonctions fléchées, Fonctions anonymes, Fonctions Auto-invoquées

Objectif : Comprennent la différence entre les fonctions classiques, les fonctions fléchées et les fonctions anonymes.

Exercice d'introduction : Déclarer et Appeler une Fonction de Base

- Déclarez une fonction `bonjour` qui prend un argument `nom`.
- La fonction doit afficher dans la console : `"Bonjour, ${nom}!"`.
- Appelez la fonction avec votre nom.
- Faites ceci pour une fonction :
    1. Normale (Regular)
    2. Fléchées (Arrow).
    3. Anonyme (Anonymous).
    4. Auto-invoquée **(IIFE, pronounced iffy, immediately invoked function expression)**

```
//Fonction(Procedure) normale
function bonjour01(nom) {
    console.log(`Bonjour, ${nom}!`);
}
bonjour01("Alice"); // Bonjour, Alice!

//Fonction Anonyme
let bonjour02  = function(nom) {
    console.log(`Bonjour, ${nom}!`);
}
let bonjour12 = bonjour02;
bonjour02("Alice"); // Bonjour, Alice!
bonjour12("Alice"); // Bonjour, Alice!

//Fonction fléchées/Expression Lambda
let bonjour03 = (nom) => {
    console.log(`Bonjour, ${nom}!`);
}
let bonjour13 = bonjour03;
bonjour03("Alice");  // Bonjour, Alice!
bonjour13("Alice");  // Bonjour, Alice!

//Fonction auto-invoquée
(function(nom) {
    console.log(`Bonjour, ${nom}!`);
})("Alice");
```

Exercice 1 : Déclarer et Appeler une Fonction avec une valeur de retour

- Déclarez une fonction **somme** qui prend deux arguments a et b.
- La fonction doit retourner la somme de **a** et **b**.
- Appelez la fonction avec les valeurs 5 et 3, et affichez le résultat.
- Refaites l'exercice avec les quatre types de fonctions cités dans l'introduction.

Exercice 2 : Déclarer et Appeler une Fonction avec une valeur de retour

- Créez une fonction qui prend un nombre x en argument et retourne le **carré** de x.
- Appelez carre avec le nombre 4 et affichez le résultat.
- Refaites l'exercice avec les quatre types de fonctions cités dans l'introduction.

Exercice 3 : Déclarer et Appeler une Fonction avec une valeur de retour

- Créez une fonction multiplier qui prend deux paramètres a et b.
- La fonction doit retourner **le produit** de a et b.
- Appelez multiplier avec les valeurs 5 et 7, et affichez le résultat.
- Refaites l'exercice avec les quatre types de fonctions cités dans l'introduction.

Exercice 4 : Déclarer et Appeler une Fonction qui prend un ensemble dynamique de paramètres

Syntaxe :

```
function concatDynamique(...nombres) { //les paramètres forment un tableau
    return nombres.join(";");//retourne un string qui concatène l'ensemble des éléments du tableau
}
let myStr = concatDynamique(1, 5, 9, 13);
```

- Créez une fonction sommeDynamique qui prend plusieurs paramètres et retourne la somme de tous les éléments.
- Appelez sommeDynamique avec les valeurs 5, 7, 8, 10, 169 et affichez le résultat.
- Refaites l'exercice avec les quatre types de fonctions cités dans l'introduction.

Exercice 5 : Fonction avec Paramètres par Défaut ***(Rest Parameter)***

- Créez une fonction equationDynamique qui prend en paramètre deux entier a et b minimum, ainsi qu'un ensemble d'entier et retourne l'équation suivante :

$$a^{**}b + \sum n.$$

- Appelez equationDynamique avec 2 valeurs, ensuite 3 ensuite 4… jusqu'à 10 paramètres, et affichez le résultat.
- Refaites l'exercice avec les quatre types de fonctions cités dans l'introduction.

Exercice 6 : Fonction avec Paramètres par Défaut

Syntaxe :

```
function test(firstParam= "default value") {

}
```

- Déclarez une fonction direBonjour qui prend un paramètre nom avec une valeur par défaut de "inconnu".
- La fonction doit afficher "Bonjour, ${nom}!".
- Appelez direBonjour sans paramètre, puis avec le paramètre "Paul".
- Refaites l'exercice avec les quatre types de fonctions cités dans l'introduction.

Exercice 7 : Fonction avec deux Paramètres par Défaut

- Déclarez une fonction salutation qui prend un paramètre nom avec une valeur par défaut de "inconnu", et un message qui le précède avec une valeur par défaut de "Bienvenue".
- La fonction doit afficher "${message}, ${nom} ! "
- Appelez direBonjour sans paramètre, puis avec le paramètre "Bienvenue", puis avec "Bienvenue" et "Paul".
- Refaites l'exercice avec les quatre types de fonctions cités dans l'introduction.

Exercice 8 : Générateur de fonctions, les fermetures ***(Closures)***

Syntaxe :

```
function createFunction(fixedParam) { //1er param (fixe)
    return function(custumizableParam) { //2eme param(dynamique)
        console.log(`${fixedParam} ==> ${custumizableParam}!`);
    }
}

const myFunctionA = createFunction("A")
myFunctionA("B") // A ==> B
myFunctionA("C") // A ==> C
myFunctionA("D") // A ==> D

const myFunctionZ = createFunction("Z")
myFunctionZ("E") // Z ==> E
myFunctionZ("F") // Z ==> F
myFunctionZ("G") // Z ==> G
```

- Déclarez une fonction creerSalutation qui retourne une nouvelle fonction de salutation personnalisée.
- creerSalutation prend un message en argument et retourne une fonction qui prend un nom.
- La fonction retournée utilise le message passé initialement pour créer une salutation.
- En appelant creerSalutation, vous créez des versions de la fonction de salutation personnalisée en fonction du message donné.
- **Refaite le même exercice avec des fonctions fléchées.******

Exercice 9 : Générateur de fonctions, les fermetures ***(Closures)***

D'après ce que vous avez appris dans ce TP, créez une fermeture pour calculer :

- le produit d'un chiffre "a" fixe, puis dynamiquement un chiffre "b".
- la somme d'un chiffre "a" fixe, puis dynamiquement un chiffre "b".
- la soustraction d'un chiffre "a" fixe, puis dynamiquement un chiffre "b".
- la division d'un chiffre "a" fixe, puis dynamiquement un chiffre "b".
- la puissance d'un chiffre "a" fixe, puis dynamiquement un chiffre "b".
- Refaite l'exercice avec des fonctions fléchées.

Exercice 10 : Callback Functions

- Créez une fonction qui prend en paramètre une autre fonction et retourne son retour en lui passant tout les paramètres "possibles".

- Créez une fonction qui prend en paramètre deux autres fonctions et retourne aléatoirement le retour de l'une des deux en leurs passant tous les paramètres "possibles". ***Nb : utilisez Math.random() % 2 == 0.***
- Créez une fonction qui prend un autre fonction en parametre et l'execute apres 2 seconds (2000ms). ***Nb :utilisez setTimeout(()=>{}, 2000) ;***
- Créez une fonction executerApresAction qui prend un nom et un *callback*. Cette fonction doit afficher un message du type "Bonjour, ${nom}!". Ensuite, elle doit exécuter le *callback* pour effectuer une action supplémentaire (console.log("Bienvenue dans notre programme!");).
- Créez une fonction calculatrice, qui prend 2 nombre en paramètres et un troisième parametre (callback function), et appelle le callback sur les deux paramètres avec le même ordre re retourne son retour.
  Nb : calculatrice(5, 3, addition); // Devrait retourner 8
  calculatrice(10, 7, soustraction); // Devrait retourner 3
  //addition et soustraction sont des fonctions.
- Créez une fonction qui prend un ensemble de fonctions en paramètre et les exécute l'une après l'autre (utilisez une boucle et le Rest Parameter).
- **Créez une fonction rechercher qui prend en parametre un tableau d'entiers et une fonction de callback, le callback joue le rôle d'un filtre et retourne le premier élément qui satisfait sa condition. L'objectif est de trouvez un element dans le tableau en utilisant des filtres sous forme de fonctions-callback.**********

  Let utilisateurs_age = [10, 20, 22, 30, 44, 60, 45, 45] ;
  const utilisateur30 = rechercher(utilisateurs, estAgeDe30) ;