Exercices : JavaScript

Attention tout morceau de code écrit dans la console d'un navigateur ou sur le site JSBin sera pas sauvegarder. Il faut donc penser à sauvegarder son code JavaScript.

Pour sauvegarder son code, on peut l'enregistrer dans un fichier texte sous le blocnote (notepad), sublimetext (version gratuite), VS Code (gratuit) ou tout autre traitement de texte.

Exercice 1 : Exemples de fonctions

1) Donner une interprétation du code JavaScript `ScriptCube`

`ScriptCube`	`ScriptCercle`

`var a = 5;`
 	
`function aireCube(x) {`
`  aireFace = a**2;`
`  aireTotale = 6  * aireFace;`
`  return aireTotale;`
`}`
 	
`console.log(aireCube(a));`
	`var pi = Math.PI;`
	
	`function aireCercle(r){`
	`  return pi * r**2;`
	`}`
	
	`var monAire = aireCercle(5);`
	`console.log('Aire du cercle est ', monAire)`

2) Copier et faire fonctionner ce script suivant pour différentes valeurs de a.

3) Écrire une fonction `airePave` qui prend en paramètres 3 valeurs : longueur, largeur et hauteur et qui retourne l'aire d'un pavé droit.

4) En JavaScript, certaines variables ont des noms réservées. Par exemple la valeur de π s'obtient avec la commande `Math.PI` On peut stocker sa valeur dans une variable. `ScripteCercle` donne le calcul de l'aire d'un cercle en fonction de son rayon r. Écrire les fonctions en respectant leurs prototypes indiquées.

Fonction	`aireCylindre`	`aireSphere`	`aireCone`
Paramètres Précondition	r //rayon h // hauteur **Type** : entier ou flottant	r // rayon **Type**: entier ou flottant	r //rayon h // hauteur **Type** : entier ou flottant
Sortie Postcondition	**Type : flottant**	**Type : flottant**	**Type : flottant**
Description	Donne l'aire d'un cylindre en fonction de sa hauteur et de son rayon.	Donne l'aire d'une sphère en fonction de son rayon.	Donne l'aire d'un cône (base + surface de révolution) en fonction de h et r.

Exercice 2 : Instructions conditionnelles.

Partie A : Introduction

En JavaScript, les instructions conditionnelles sont utilisées avec la structure suivante (cf **ScriptCond1**)
Il n'existe pas de structure **elseif** comme sous Python (elif). Si on doit faire plusieurs tests différents, on peut les enchaîner comme dans **ScriptCond2**.

ScriptCond1

```
var age = 10;

if (age<18) {
  console.log("Tu es encore au lycée")
  }
else {
  console.log("Tu es bachelier bravo!")
}
```

ScriptCond2

```
var age = 20;

if (age<18) {
  console.log("Tu es encore au lycée.")
  }
else if (age <60){
  console.log("Plus que" , 60- age, "ans à
travailler.")
      }
else { console.log("Vive la retraite.")
     }
```

1) Faire des tests pour des ages différents sur les 2 scripts.

2) Changer les conditions sur les ages dans **ScriptCond1.** Modifier les textes en conséquences.

3) Changer les conditions sur les ages dans **ScriptCond2.** Modifier les textes en conséquences.

4) On rappelle que l'age de la retraite a été modifiée. Rechercher et mettre à jours **ScriptCond2** pour le faire correspondre aux données actuelles du gouvernement.

Partie B : Taux d'imposition

Le taux d'imposition en France fonctionne sur le principe de tranche. Le tableau suivant donne un exemple pour un salaire annuel de 45 000€ la valeur de l'impôt brut. Si le salaire dépasse la plage de la tranche, on calcule son impôt sur la taille de la tranche totale. Ici 45 000 est supérieur à 30 000 donc on calcule son impôt dans la catégorie B sur une base de 30 000 – 10 000 = 20 000 €. Pour la catégorie D, on a 0 € car n'atteint pas dans cette catégorie. Son impôt sera donc de 2 000 +3 000 = 5000 €

Catégorie de tranches	Taux	Somme
A : inférieur à 10 000 €	**0 %**	0 €
B : entre 10 000 € et 30 000 €	**10 %**	$20\ \ 000\times\frac{10}{100}=2\ \ 000€$
C : entre 30 000 € et 75 000 €	**20 %**	$15\ \ 000\times\frac{20}{100}=3\ \ 000€$
D : supérieur à 75 000 €	**30 %**	0€

1) Coder une fonction **maCategorie** qui prend en paramètre un salaire annuel et renvoie la catégorie lui correspondant.

2) Coder une fonction **monImpot** qui prend en paramètre un salaire annuels puis renvoie son impôt.

Exercice 3 : Les boucles

Jean Caisse épargne de l'argent sur un compte. Chaque mois il dépose 500 €. Au bout de 12 mois, sa banque lui reverse 2 % de la somme contenue dans se compte.

ScriptBoucle1

```
var somme = 0;
var mois = 14;

for (let i = 1 ; i< mois ; i++) {
      somme = 500 + somme;
      if (i % 12 === 0 ) {
            somme= somme *1.02;
      }
}

console.log("J'ai",somme,"€ après",mois,"mois")

function capital(mois){
      // calcul de mon capital
}
```

ScriptBoucle2

```
var seuil = 10000;
var somme = 0;
var mois = 14;
var nb_invest = 0 ;

function investissement(nb_mois) {

      for (let i = 1 ; i< nb_mois ; i++) {
            somme = somme +500 ;
// On a 2 tests à faire et incémenter nb_invest
      }
      return nb_invest ;
}

console.log(investissement(60)) ;
```

1) Faire le calcul des sommes obtenues après 6 mois, 12 mois, 18 mois et 30 mois.

2) Faire fonctionner le script **ScriptBoucle1** pour différente valeur de la variable mois. Cela concorde-t-il avec les résultats de la question 1.

3) Expliquer l'instruction contenu dans le bloc if `i % 12 === 0` .

4) Compléter le fonction capital pour qu'elle renvoie la somme contenue dans son compte après n mois.

5) Si la somme du compte dépasse 10 000 €, Jean utilise ces 10 000 € pour les investir dans des obligations bancaires plus lucratives. Dans **ScriptBoucle2**, code la fonction investissement qui prend en paramètre le nombre de mois et renvoie le nombre d'achat d'obligations bancaires.

Problème :

Jean-Michel Gaming joue régulièrement sur son jeux de carte virtuel sur mobile : **Python-Cards**.

Chaque jours il peut collecter **35 Pièce d'or** (on abrégera en PO) et les dépenser dans des lots de cartes dont le prix varie. Le tableau suivant donne les variation de prix en fonction du nombre d'achat.

Paquet 1	Paquet 2	Paquet 3	Paquet 4	Paquet 5
100 PO	115 PO	135 PO	165 PO	190 PO

A partir du **6ième paquet** le prix revient à 100 PO et on refait le cycle.

1) Coder la fonction **mesPO** qui prend en paramètre un nombre de semaines et renvoie les PO obtenus.

2) Coder la fonction **mesPaquets** qui prend en paramètre une valeur en PO et renvoie le nombre de paquets obtenus.

3) Coder la fonction **mesProchains** qui prend en paramètre mes PO restants, le numéro du paquet et le nombre de nouveaux paquets que l'on veut obtenir et renvoie le nombre de jours nécessaires pour obtenir ces nouveaux paquets.