Les fonctions

Les fonctions sont notre moyen d'action en programmation, c'est avec elles que vous pourrez afficher à l'écran, récupérer des données de l'utilisateur et bien d'autres choses. Avant tout on doit distinguer 2 types de fonctions : celles qui sont présentes de base dans le langage (« built-in » qui signifie « intégrées » en Français) et celles que vous pourrez définir vous-mêmes.

Dans la majorité des cas une fonction possède, comme les variables, un nom. (Pour l'exemple on utilisera « console.log », fonction qui permet d'afficher des données dans la console en Javascript) Mais cela ne suffit pas à exécuter une action car « console.log » est la fonction mais on ne lui dit pas d'agir ! Pour lancer cette fonction on doit *l'appeler* et pour faire cela on ajoute juste des parenthèses (« () ») après le nom de la fonction. On pourrait donc dire, si on reprend le cas de notre fonction fictive, d'afficher à l'écran en tapant « console.log() ».

Oui mais afficher quoi ? Car pour l'instant c'est bien beau de demander d'afficher à l'écran mais si on ne précise pas quoi afficher on ne verra rien à l'écran. C'est là qu'intervient une autre capacité des fonctions : elles peuvent prendre ce qu'on appelle des *arguments*.

Les arguments sont des informations qu'on va pouvoir donner à la fonction pour qu'elle effectue son boulot. Ces informations sont simplement des valeurs et on peut aussi évidemment utiliser des variables. Le nombre et types d'arguments nécessaires dépend par fonction mais dans tous les cas ils se placent, lors de l'appel à la fonction, entre les parenthèses et sont séparés par des virgules.

Ex : **console.log("Hello", 'World !') // Affiche dans la console "Hello World !"**

L'ordre dans lequel on rentre les arguments à de l'importance. Imaginez qu'une fonction puisse prendre deux arguments et qu'elle enlève la valeur du deuxième au premier.

Ex : **function calculateDifference(num1, num2) { // Architecture de calculateDifference**

```
    return num1 - num2;
}

calculateDifference(10, 4)
```

Dans notre cas « 10 - 4 » donne 6 mais si on avait inversé les arguments on aurait eu « 4 - 10 » qui donne -6 ce qui n'est pas du tout le même résultat !

Une dernière chose importante à savoir est qu'une fonction peut retourner une valeur. Lorsqu'elle a fini d'exécuter toutes ses actions une fonction peut, dépendamment des fonctions, renvoyer une information que l'on peut utiliser normalement. Voici un exemple pour illustrer :

Ex : **function returnTwo() { // Architecture de returnTwo**

    return 2;

}

returnTwo() + 3 // Une addition entre une fonction qui renvoie une valeur 2 et le chiffre 3, leur somme vaudra bien 5 (le nom d'une fonction n'a pas forcément de lien avec ce qu'elle renvoie, le nom a juste été choisi par clarté)

Pour l'exemple ci-dessus c'est comme si on avait « remplacé » la fonction par ce qu'elle renvoie : returnTwo() + 3 => 2 + 3 mais toutes les actions de la fonction « returnTwo », dans le cas où elle en avait, ont bien été effectuées.

On n'abordera pas la création de fonctions maintenant mais avant de passer au point suivant sachez qu'on ne peut pas créer de « nouvelles capacités », les fonctions que vous définirez vous-mêmes serviront à réutiliser le code et à apporter de la lisibilité.

Quizz

1) Comment dois-je faire pour exécuter une fonction dont le nom est « console.log » ?

2) Quelle est la bonne syntaxe pour faire appel à une fonction « calculateArea » (« calculAire ») qui prend 2 arguments, *width* et *height* (« largeur » & « hauteur »), qui vaudraient respectivement dans notre cas 7 et 9 ?

function calculateArea(width, height) {

    return width * height;

}

3) Quel est le résultat du calcul suivant ? « returnSum(13, 4) - 7 » (Sum = « Somme »)

```
function returnSum(num1, num2) {
    return num1 + num2;
}
```

4) Que renvoie l'appel à la fonction suivante ? « multiplyByTwo(returnSeven()) »

```
function multiplyByTwo(num) {
    return num * 2;
}
function returnSeven() {
    return 7;
}
```

5) Quel est l'erreur probable dans ce qui suit ? « calculatePerimeterSquare("sept") »

```
function calculatePerimeterSquare(num) {
    return num * 4;
}
```

6) Que se passera-t-il lorsque le code qui suit s'exécutera ?

```
var lastName = "Lavilliers"
var firstName = "Bernard"
console.log(firstName, lastName);
```