Les fonctions

[Mise à jour le 25/6/2021]

- **Sources** et compléments sur **MDN Web Docs**
  - Fonctions
- **Lectures connexes**
  - Wikis WebPEM : "Préparer un projet de site Web avec l'IDE VSCode"

1. Introduction

Il existe deux grands types de fonctions en JavaScript : les fonctions **natives** ou **prédéfinies** (qui sont en fait des méthodes) qu'il suffit d'appeler et les fonctions **personnalisées** à créer.

Une fonction est un **ensemble de traitements réutilisables** effectués à partir de **paramètres** et renvoyant un **résultat**.

Description

typeRetour **nomFonction**(*type*$_1$ param$_1$,... ,*type*$_n$ *param*$_n$, *[*,type *paramFacultatif=valeur])*
- typeRetour *est le type de donnée renvoyé par la fonction.*
- *param*$_1$,... ,type$_n$ sont les types des paramètres passés à la fonction.
- Lorsqu'un paramètre facultatif est utilisé, une valeur par défaut est appliquée.

Exemple

```
Integer parseInt(String chaîne [, String base])
```

2. Les fonctions natives (built-in)

- **Ressource** : Fonctions prédéfinies

Les fonctions natives sont intégrées au langage.

Exemple : *Integer* **parseInt**(*String* chaîne [,*String* base])

*.js

```
console.log(parseInt("150")); // Renvoie 150
console.log(parseInt("150.45")); // Renvoie 150
console.log(parseInt("150xxx")); // Renvoie 150
console.log(parseInt("xxx150")); // Renvoie NaN
console.log(parseInt("FF",16)); // Renvoie 255
```

3. Les fonctions personnalisées

3.1 Déclarer une fonction

- **Méthode 1** : courante

Syntaxe

*.js

```
function nomFonction(param1, ... , paramN){
    // Instruction(s)
}
```

- **Méthode 2** : avec le constructeur *function*

Syntaxe

*.js

```
nomFonction = function(param1, ... , paramN){
    // Instruction(s)
}
```

Le nom d'une fonction ne recevant pas de paramètres est écrit avec des parenthèses soit : nomFonction().

3.2 Renvoyer un résultat

Pour renvoyer un résultat, on utilise l'instruction ***return*** suivie de la valeur. *return* stoppe l'exécution de la fonction et peut être utilisé sans valeur.

Exemple

*.js

```
// Calcul de la somme des valeurs de 1 à n
function Somme1aN(n) {
   var somme = 0;
   for (i = 0; i <= n; i++) {
      somme += i;
   }
   return somme;
}
```

3.3 Utiliser une fonction

*.js

```
// Appel de la fonction ci-dessus
var n = 45;
var resultat = Somme1aN(n);
console.log("Somme=" + resultat + " pour n=" + n); // Affiche
Somme=1035 pour n=45
```

Une fonction est souvent déclarée dans un fichier JavaScript externe. Son intégration dans la page web se fait avec la balise <*script*> et l'attribut *src*.

3.4 Gérer les paramètres facultatifs

Les paramètres facultatifs sont définis à la fin de la liste de paramètres, après tous les paramètres obligatoires.

Exemple

*.js

```
// Le paramètre facultatif pluriel permet de gérer les exceptions
function getPluriel(nb, pluriel = "s") {
    return nb > 1 ? pluriel : "";
}
var nbChouette = 2, nbHibou = 3; nbGrandduc = 1;
console.log(nbChouette + " chouette" + getPluriel(nbChouette) + ", " +
nbHibou
+ " hibou" + getPluriel(nbHibou, "x") + " et " + nbGrandduc + " Grand-
duc"
```

```
+ getPluriel(nbGrandduc) + " me regardaient fixement !!!");

// Résultat dans la console : 2 chouettes, 3 hiboux et 1 Grand-duc me
regardaient fixement !!!
```

4. Variables globales et variables locales

voir Variables et constantes

5. Paramètres de fonctions

Les paramètres sont dits :
- **formels** (ou **arguments**) lorsqu'ils apparaissent au niveau de la définition d'une fonction (d'une méthode), par opposition aux paramètres
- **effectifs** qui sont listés dans un appel de fonction (méthode).

L'ensemble des paramètres passés à une fonction est contenu dans le tableau ***arguments[]***, accessible à l'intérieur de la fonction.

Exemple

*.js

```
```

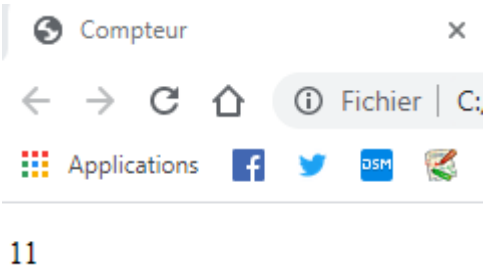
6. Fonctions anonymes

Les fonctions anonymes sont, comme leur nom l'indique, des fonctions qui ne vont pas posséder de nom. Généralement, on utilisera les fonctions anonymes lorsque le code de la fonction n'est appelé qu'à un endroit dans le script et n'est pas réutilisé.

- **Minuteur setInterval()**

La fonction setInterval() attend deux paramètres:
- Le nom de la fonction a exécuter à intervalle régulier.
- Le délai en millisecondes de l'intervalle de répétition.

Exemple



*.js

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Compteur</title>
</head>

<body>
    <p id="compteur">0</p>
    <script type="text/javascript">
        var nb = 0;
        setInterval(function () {
            nb++;
            document.getElementById("compteur").innerHTML = nb;
        }, 1000);
    </script>
</body>
```

- **Fonction auto-exécutée**

Syntaxe

```
(function(param1, param2,...,paramN){
    // Traitements
})(p1, p2, ...,pN);
```

Cette syntaxe remplace l'écriture traditionnelle de déclaration de la fonction puis de son appel.