

Séquence 7 : Les fonctions

I. Généralités

Nous avons déjà vu beaucoup de fonctions : `addChild()`, `addEventListener()`, `trace()` ...

Ce sont des fonctions pré-définies par AS3.

Nous avons aussi la possibilité de créer nos propres fonctions !

Intérêt des fonctions

Une fonction est une portion de code que l'on peut appeler au besoin (c'est une sorte de sous-programme).

L'utilisation des fonctions évite des redondances dans le code : on obtient ainsi des programmes plus courts et plus lisibles.

Par exemple, nous avons besoin de convertir à plusieurs reprises des degrés Celsius en degrés Fahrenheit :

```
public function Main():void
{
    trace(100.0 * 9.0 / 5.0 + 32.0);
    trace(37.0 * 9.0 / 5.0 + 32.0);
    trace(233.0 * 9.0 / 5.0 + 32.0);
}
```

La même chose en utilisant une fonction :

```
private function Main():void
{
    conversion(100);
    conversion(37);

    var temperature : Number = 233;
    conversion(temperature);
}

private function conversion(degCelsius : Number):void
{
    trace(degCelsius * 9.0 / 5.0 + 32.0);
}
```

Syntaxe :

La définition d'une fonction s'effectue au travers d'une syntaxe bien précise qui permet de spécifier le nom d'appel de la fonction ainsi que les instructions qui la composent.

Syntaxe

Pour définir une fonction, il convient d'utiliser la syntaxe suivante :

```
//❶ En-tête de la fonction
private function nomDeLaFonction(parametre1:type, parametre2:type, ...):type
{
    //❷ Définition du corps de la fonction
}
```

❶ Le nom d'une fonction est défini grâce à un en-tête de fonction qui est composé :

- Du mot-clé `function` qui débute l'en-tête afin de préciser au lecteur Flash que les instructions suivantes concernent la définition d'une nouvelle fonction.
- Du nom de la fonction, choisi par le développeur afin d'identifier cette fonction.

Remarque

Le nom d'une fonction est choisi de façon à représenter et résumer tout ce qui est réalisé par son intermédiaire.

Bien évidemment, il est fortement déconseillé de choisir un nom identique à celui d'une fonction définie par le langage.

- De parenthèses ouvrante et fermante encadrant une suite de paramètres séparés par des virgules. Le type de chacun des paramètres doit être spécifié, comme pour une déclaration de variables. L'utilisation des paramètres est décrite plus bas, à la section « Les paramètres d'une fonction » de ce chapitre.

- D'un type précédé de « : » afin de préciser le type de la valeur retournée par la fonction.

```
private function conversion(degCelsius : Number ) : void //cette fonction ne retourne rien
    {trace(degCelsius * 9.0 / 5.0 + 32.0); }
```

```
private function conversion(degCelsius : Number ) : String //cette fonction retourne une chaîne de caractère
    {trace(degCelsius * 9.0 / 5.0 + 32.0);
    return "leTexte"; }
```

Une fois le nom de la fonction défini, il est nécessaire de l'associer aux instructions décrivant le comportement de la fonction. Pour cela, l'en-tête de la fonction est placé au-dessus du corps de la fonction, déterminé par les accolades ouvrante et fermante ({ et }). Le corps d'une fonction est essentiellement composé de déclarations de variables et d'instructions d'affectation, de tests, de répétitions, d'appels à des fonctions, etc.

II.Exemples

Exemple n°1

```
private function maPremiereFonction() : void //cette fonction n'a pas de paramètre et ne retourne rien
    {
        trace("Bonjour");
    }
```

Une fois la fonction définie, nous pouvons l'appeler dans la fonction *Main()*:

```
MaPremiereFonction(); // ne pas oublier les parenthèses ()
```

Exemple n°2

La fonction suivante simule le comportement d'un dé à 6 faces.

Pour cela, on utilise la fonction *random()* du module *Math*.

```
private function tirageDe():int //cette fonction n'a pas de paramètre et retourne _____
    { var valeur : int ;
      valeur = Math.floor(Math.random() * 6) + 1;
      return valeur;
    }
```

Appeler cette fonction plusieurs fois dans la fonction *Main()* :

```
trace( tirageDe() );
```

III.La visibilité des variables

Après étude des programmes énoncées ci-dessus, nous observons qu'un script est constitué d'instructions, dont des instructions de déclaration de variables et de fonctions. Il existe, de fait, une notion d'extérieur et d'intérieur aux fonctions.

Par ailleurs, une fonction ne peut pas utiliser, dans ces instructions, une variable déclarée dans une autre fonction.

De plus; pour avoir une variable visible depuis toutes les fonctions, il faudra la déclarer en dehors de toute fonction. Les variables ainsi déclarées sont appelées variables « globales ».

Pour mieux visualiser ces propriétés, examinons le programme ci-dessous.

```
private var valeur:int;

public function Main():void
{
    initialise();
    double();
}

private function initialise():void{
    var valeurLocale : int;
    valeurLocale = 10 ;
    trace("ValeurLocale = " + valeurLocale + " dans initialise() ");

    valeur = 2
    trace("Valeur = " + valeur + " dans initialise() ");
}

private function double():void{
    valeur = valeur * 2;
    trace("Valeur = " + valeur + " dans double() ");
}
```

IV.Exercices

Exercice 1

1) Ecrire une fonction `carre ()` qui retourne le carré d'un nombre :

```
trace( carre(11.11111) ) ;
```

```
123.4567654321
```

2) Avec une boucle `for` et la fonction `carre ()`, écrire un script qui affiche le carré des nombres entiers de 1 à 100 :

```
12 = 1
```

```
22 = 4
```

```
32 = 9
```

```
...
```

```
992 = 9801
```

```
1002 = 10000
```

Exercice 2 Ecrire une fonction qui retourne l'aire de la surface d'un disque de rayon R.

Exemple :

```
trace( AireDisque(2.5) ) ;
```

```
19.6349540849
```

Exercice 3 :

Ecrire une fonction qui retourne la factorielle d'un nombre entier N.

On rappelle que : $N! = N \times (N-1) \times (N-2) \times \dots \times 3 \times 2 \times 1$

```
trace( factorielle(50)) ; // Resultat : 3.04140932017133e+64
```

Exercice 4 :

4.1. A l'aide de la fonction Math.random() qui renvoie un nombre pseudo-aléatoire n, où $0 \leq n < 1$.

Ecrire une fonction randint(entier) qui retourne un nombre entier N entre $0 \leq N < \text{entier}$.

Exemple randint(44) : retourne un nombre pseudo-aléatoire N, où $0 \leq N < 44$.

4.2. Ecrire une fonction qui retourne un mot de passe de longueur N (chiffres, lettres minuscules ou majuscules).

On donne :

```
var chaine : String = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
chaine.charAt( entier );
chaine.length ;
```

Exemple :

```
trace( genererMotDePasse(10) );
eSDXkjUVgP
trace( genererMotDePasse(6) );
F3hGXM
```

3) Quel est le nombre de combinaisons possibles pour un mot de passe à 10 ou 6 caractères ?

4) Quelle durée faut-il pour casser le mot de passe avec un logiciel capable de générer 1 million de combinaisons par seconde ?

Lien utile : www.exhaustif.com/Generateur-de-mot-de-passe-en.html

Exercice 5 : Ecrire une fonction qui retourne une carte (au hasard) d'un jeu de Poker à 52 cartes.

On utilisera la fonction choice() ou randint() du module random.

On donne :

```
var listeCarte : Array = [ '2s', '2h', '2d', '2c', '3s', '3h', '3d', '3c', '4s', '4h', '4d', '4c', '5s', '5h', '5d', '5c', '6s', '6h', '6d',
'6c', '7s', '7h', '7d', '7c', '8s', '8h', '8d', '8c', '9s', '9h', '9d', '9c', 'Ts', 'Th', 'Td', 'Tc', 'Js', 'Jh', 'Jd', 'Jc', 'Qs', 'Qh', 'Qd',
'Qc', 'Ks', 'Kh', 'Kd', 'Kc', 'As', 'Ah', 'Ad', 'Ac' ] ;
listeCarte[ entier ];
listeCarte.length
```

Exemple : trace(tirageCarte());

7h

trace(tirageCarte());

Kd

Exercice 6 Ecrire une fonction qui retourne une liste de N cartes **différentes** d'un jeu de Poker à 52 cartes.

On donne :

```
var listeCarte : Array = [ '2s', '2h', '2d', '2c', '3s', '3h', '3d', '3c', '4s', '4h', '4d', '4c', '5s', '5h', '5d', '5c', '6s', '6h', '6d',
'6c', '7s', '7h', '7d', '7c', '8s', '8h', '8d', '8c', '9s', '9h', '9d', '9c', 'Ts', 'Th', 'Td', 'Tc', 'Js', 'Jh', 'Jd', 'Jc', 'Qs', 'Qh', 'Qd',
'Qc', 'Ks', 'Kh', 'Kd', 'Kc', 'As', 'Ah', 'Ad', 'Ac' ] ;
var tirage : Array = [number] ;
listeCarte.splice( i , 1 );
```

Exemple :

```
trace( tirageNcarte(8) );
```

```
Ts, Qc, Ks, Ts, 8h, Qs, Jd, Jc
```

```
trace( tirageNcarte(25) );
```

```
5d, Ad, Ts, 2h, Qc, 7s, Ah, 9c, Ac, Kd, Ad, Qc, Ah, 9d, 8c, 8c, Js, 3c, 8c, Qd, As, 2c, 3h, Ac, As
```