JS 03 : Les variables

La déclaration de variables

Les variables contiennent des données qui peuvent être modifiées lors de l'exécution d'un programme. On y fait référence par le nom de cette variable. Pour nommer une variable ou une fonction, le développeur définit des identificateurs. Un identificateur doit commencer par une lettre (alphabet ASCII) ou le signe _ et se composer de lettres, de chiffres et des caractères _ et `$` (à l'exclusion du blanc).
Pour rappel JavaScript est sensible à la casse.

Les variables peuvent se déclarer de deux façons :

- soit de façon **explicite**. On dit à JavaScript que ceci est une variable. La commande qui permet de déclarer une variable est le mot `var`. Depuis ES6, il est également possible de remplacer `var` par `let` dont le comportement est similaire, sauf que let n'est pas reconnu par les versions anciennes des navigateurs.

Par exemple :

```
var numAdherent = 1 ;
```

```
var prenomAdherent = "Jean" ;
```

- soit de façon **implicite**. On écrit directement le nom de la variable suivi de la valeur que l'on lui attribue et JavaScript s'en accommode.

Par exemple :

```
numAdherent = 2;
```

```
prenomAdherent = "Luc";
```

Attention : malgré cette apparente facilité, la façon dont on déclare la variable aura une grande importance pour la « visibilité » de la variable dans le programme Javascript. Voir à ce sujet la distinction entre variable locale et variable globale plus loin dans ce cours.

Pour la clarté de votre script et votre facilité, on ne peut que conseiller d'utiliser à chaque fois le mot-clé `var` pour déclarer une variable.

Portée des variables

Variables de portée globale

Les variables déclarées en début de script, en dehors et avant toute structure de code (fonction, condition etc.) fonctionnent comme des variables globales : on pourra donc les exploiter partout dans le script.

Exemple :

```
var myVar = 123;

maFonction();

function maFonction()
{
console.log("myVar fonction : "+myVar);
}

if (myVar > 1)
{
console.log("myVar condition : "+myVar);
}

console.log("myVar fin : "+myVar);
```

L'objet par défaut est l'objet `window`.
Par exemple, la déclaration implicite de la variable `maVariable` sous-entend qu'elle est une propriété de l'objet window. On pourrait l'appeler ainsi : `window.maVariable`.

Exercice

Testez l'exemple de code ci-dessus et observez ce qu'il se passe.

Variables de portée locale

Toute variable déclarée à l'intérieur d'une structure de code ne sera valide que dans cette structure.

Exemple :

```
var compteur = 2;

function maFonction()
{
var myVar = 456;
console.log("myVar : "+myVar);
}

if (compteur > 1)
{
let myVar2 = "Wazaa !";
console.log("myVar2 : "+myVar2);
}

/* Ici, myVar n'est pas disponible
* car déclarée dans la fonction
* sa portée ne concerne que le code de la fonction
```

```
*/
console.log("myVar : "+myVar);

/* Ici, myVar2 n'est pas disponible
* car déclarée dans le bloc de condition
* avec une portée uniquement pour ce bloc
*/
console.log("myVar2 : "+myVar2);
```

Exercice

Testez l'exemple de code ci-dessus et observez ce qu'il se passe.

Les types de données

JavaScript utilise 5 types de données :

- Nombre : Tout nombre entier ou avec virgule tel que `22` ou `3.1416`. Tout entier en octal ou hexadécimal tel que `0387`, `0xFFA8`.
- Chaîne : toute suite de caractères comprise entre guillemets ou côtes telle que "suite de caractères" ou 'suite de caractères'
- Booléen : les mots `true` pour vrai et `false` pour faux
- Objet : toute utilisation de variable par référence vers tout objet natif JavaScript (Array, Date, String …) ou tout objet du DOM.
- `null` : mot réservé qui ne représente pas de valeur. La valeur null est affectée volontairement à une variable.
- `undefined` : mot réservé qui est renvoyé par une variable référencée qui n'a pas encore été définie (la variable existe mais n'a reçu aucune affectation de valeur).

Exemples :

```
var myVar; // le type de la variable myVar est undefined
myVar = 324; // type number
myVar = "Bonjour"; // type string
myVar = true; // type boolean
myVar = new Array(); // type object
```

Notez que, contrairement au langage C ou C++, il ne faut pas déclarer le type de données d'une variable (pas besoin préciser `int`, `float`, `char` etc.).

Connaître le type d'une donnée

Une variable peut changer de type après une affectation. On peut vérifier le type en cours d'une variable avec l'instruction `typeof` :

```
console.log(typeof 42); // Affiche "number"
console.log(typeof 'blubber'); // Affiche "string"
console.log(typeof true); // Affiche "boolean"
```

```
var myVar;

console.log(typeof myVar); // Affiche "
```