==========
LANGAGE JAVASCRIPT : LES BASES DU LANGAGE
==========

==========
LA STRUCTURE DE BASE D'UN PROGRAMME EN JAVASCRIPT
==========

Lorsque le code JavaScript se trouve dans un fichier séparé (extension .js),
la structure générale d'un programme écrit en JavaScript est relativement simple.

```
/* zone d'en-tête */

/* déclaration des variables globales */
Declaration1;
Declaration2;

/* définition des fonctions (mot clef function) */

/* CORPS DU PROGRAMME */
Instruction1;
Instruction2;

/* FIN */
```

Même s'il est possible de déclarer nos variables à n'importe quels moments,
il est conseillé de le faire en début de script pour une meilleure lisibilité du code.

CAS D'UN SCRIPT JAVASCRIPT DANS UNE PAGE WEB

Lorsque le JavaScript est écrit dans une page Web, la structure est différente.

Dans un premier temps nous allons travailler uniquement sur la partie JavaScript.
Cela donne la structure suivante :

```
<!DOCTYPE html>
<html>
<head>
   <title>TODO supply a title</title>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <script>
      // ici les variables globales à la page et les fonctions
   </script>
</head>
<body>
   <script>
      // ici les instructions
   </script>
</body>
</html>
```

Le langage HTML utilise des balises. Nous reviendrons dessus prochainement.

La balise permettant d'ajouter du JavaScript est :

<script> ... </script>

Cette balise s'ajoute à 2 endroits particulièrement :
• entre les balises <head></head> pour les variables globales et pour les fonctions
• entre <body></body> pour les instructions

==========
LES COMMENTAIRES
==========

Les commentaires permettent une relecture de programme pour le programmeur
lui-même mais aussi pour ceux qui vont relire le programme (travail en équipe).

Un commentaire d'entête est nécessaire pour indiquer :
• le nom du programmeur
• la date
• le nom du fichier
• le descriptif du programme

Les commentaires sont du texte simple et ne sont donc pas traités comme étant du code.
Ils ne seront donc pas interprétés.

Deux types de commentaires sont possibles :

• // indiquent au compilateur que le reste de la ligne est un commentaire.
• /* .... */ indiquent au compilateur qu'à partir de /* le texte est un commentaire
jusqu'au symbole */ (permet des commentaires sur plusieurs lignes).

Exemple :

```
/*
 * File: HelloWorld.js
 * Author: sgibert
 *
 * Created on 14 septembre 2018, 20:19
 *
 * Description du programme :
 * explique ici ce que fait ton programme
 */
```

Il n'est pas utile d'associer une ligne de code avec un commentaire
si ce dernier ne fait que dire ce que fait le code.

Exemple inutile :

```
// je déclare une variable x=3
var x=3;
```

Il vaut mieux expliquer à quoi sert la variable x dans notre code.

==========
LES DECLARATIONS ET INSTRUCTIONS
==========

Une instruction ou une déclaration se termine toujours par un point-virgule (;).

C'est le cas de nombreux langages de programmation.

Mais attention : le langage JavaScript n'est pas aussi strict au niveau
de sa syntaxe et certaines instructions autorisent l'absence de ; de fin.

Pour faire simple, cela ne coûte rien d'ajouter systématiquement
ce marqueur de fin d'instruction et sera donc la règle pour nous.

==========
LES DECLARATIONS DE VARIABLES
==========

Rappel du cours d'algorithmique :

En résumé :
variable = NOM + TYPE + VALEUR

JavaScript est dit "faiblement typé" contrairement à d'autres langages
dits "fortement typés" comme par exemple le langage C.

Avec ces langages il est nécessaire de préciser le type
au moment de la déclaration d'une variable.

Exemple en C :

```
----------
int a=3;       // a est un entier et restera un entier pendant tout le programme
float b=3.5;   // b est un réel et restera un réel pendant tout le programme
char c='a';    // c est un caractère et restera un caractère pendant tout le programme
----------
```

Avec JavaScript une variable est déclarée avec la commande var ou let,
mais sans indiquer son type.

Le type sera fixé à l'utilisation.
La notion de type est bien existante mais jamais indiquée de façon explicite dans le code.

Le JavaScript a 8 types :

- Number
- String
- Boolean
- Null
- Undefined
- BigInt
- Object
- Symbol (nouveau)

Pour plus de précision :
https://developer.mozilla.org/fr/docs/Web/JavaScript/Structures_de_donn%C3%A9es

Exemple en JavaScript :

```
---------------------------------------------------------------------
var a=3;        // a est un Number
a="coucou";     // a est maintenant un String
---------------------------------------------------------------------
```

Dans ce programme la variable a est déclarée comme étant un Number.
Mais au cours du programme elle devient un String.

Bien que cela soit possible, on essaiera au maximum d'éviter ce genre de mélange.

==========
MEMOIRE VIVE
==========

Sur un PC la mémoire vive est une mémoire de travail.

Elle stocke de façon temporaire des données et des programmes.
Lorsque le PC est éteint le contenu de cette mémoire est supprimé.

On parle alors de mémoire volatile,
contrairement au disque dur qui stocke des données de façon permanente.

Cette mémoire est très rapide et communique directement avec le processeur.

En JavaScript :

La mémoire est allouée lors de la création des objets
puis libérée automatiquement lorsque ceux-ci ne sont plus utilisés.

Cette libération automatique est appelée :

- garbage collection (anglais)
- ramasse-miettes (français)

==========
CHOIX DU NOM
==========

Bien qu'il soit possible de donner des noms de variables avec des caractères
accentués en JavaScript, on se limitera aux caractères suivants :

- a..z, A..Z
- 0..9 → interdit au début du nom
- _ (underscore) → autorisé au début du nom

On ne peut pas utiliser :

- un tiret -
- un point .
- un mot clef du langage comme nom de variable
  (new, if, typeof, ...)

On utilise l'alternance de minuscules, majuscules ou _
pour améliorer la lecture du nom.

Rappel : choisir un nom qui caractérise au mieux la variable.

Exemples :

nbrDeCaisses
poste_1
poste_123

Le langage JavaScript est sensible à la casse.

Ainsi :
Poste_1 ≠ poste_1

==========
DECLARATION DE VARIABLES ET CONSTANTES
==========

Pour pouvoir être utilisable une variable doit d'abord être déclarée.

2 mots clefs sont utilisés :

- var
- let

Une variable peut être déclarée à n'importe quel moment.

Avec var :

Quel que soit l'endroit de la déclaration (hors fonction),
la variable est globale.

Même déclarée dans un bloc d'instruction,
la variable est globale.

Avec let :

La variable est locale si déclarée dans un bloc { }.

Locale signifie :
accessible uniquement dans ce bloc.

Si déclarée en dehors d'un bloc,
la variable est globale.

Que choisir entre var et let ?

Nous utiliserons let.
C'est ce qui est recommandé.

Pour aller plus loin :
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/let

---

Rappel : le nom d'une variable est unique dans votre programme.

---

const :

Permet de déclarer une constante.

La constante doit être initialisée au moment de sa création.

Son comportement (global ou local) est le même que let.

Syntaxe de base :

let a;

Exemple :

```
---------------------------------------------------------------------
<script>
let a;          // a est ici undefined
let b=4;        // b est un Number vaut 4
let c,d=3,e;    // 3 variables
const f=3;      // f est une constante
var g="Bonjour"; // g est un String
</script>
---------------------------------------------------------------------
```

==========

LES DIFFERENTS TYPES ET LEUR PLAGE DE VALEUR

==========

Tableau récapitulatif :

Type	Type Algo	Taille	Plage de valeurs
Number	Entier ou Réel	8 octets	-(2^53 -1) à 2^53 -1 (entiers) 5e-324 à 1.7976931348623157e+308 (réels)
BigInt	Entier	?	Au-delà de la limitation Number
String	Chaine/Caractère	2 octets	Chaîne de caractères
Boolean	Booléen	1	true ou false
Null	?	?	null
Undefined	?	?	undefined

==========
DETAILS SUR LES DIFFERENTS TYPES
==========

• Number

Représente un entier ou un réel.
Peut être négatif (préfixé par -).
Le signe + est possible mais inutile.

Exemple :

let a=+10;
let a=10;

Pour les entiers :

Plage :
-(2^53 -1) à 2^53 -1
soit :
-9007199254740991 à +9007199254740991

Au-delà :
JavaScript ne garantit plus la cohérence des calculs.

Exemple :

let a=2**53-1;
a=a+20;

Résultat incohérent possible.

Représentation informatique

En mathématiques :
+∞ et -∞ sont théoriquement illimités.

En informatique :
les nombres sont limités.

Sur 1 octet :
2^8 = 256 valeurs
intervalle : [-127 … 0 … 128]

La représentation est circulaire.

Pour les réels :

Plage :
5e-324 à 1.7976931348623157e+308

Au-delà :
Infinity ou -Infinity

Exemple :

```
let a=(2**2000);     // Infinity
let b=-(2**2000);    // -Infinity
```

• BigInt

Permet de dépasser la limite des entiers Number.

Il faut suffixer chaque valeur avec n.

Exemple :

```
let a=2n**53n-1n;
a=a+20n;
```

• String

Représente une chaîne ou un caractère.

```
let a="coucou";
let b='coucou';
```

Chaque caractère est codé sur 16 bits (Unicode).

Caractères spéciaux :

\' apostrophe
\" guillemets
\& &
\\ barre oblique inverse
\n nouvelle ligne
\r retour chariot
\t tabulation
\b espace
\f saut de page

==========
LES OPERATEURS
==========

3 grandes catégories.

==========
LES OPERATEURS ARITHMETIQUES
==========

Priorité des opérateurs comme en mathématiques.
Utiliser des parenthèses pour éviter les erreurs.

Opérateur	Signification
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo
**	Puissance

Exemples :

```
let a=3,b=5,c=a+b;
let a=3,b=50.5,c=a+b;
let a=true,b=50.5,c=a+b;
let a=true,b=true,c=a+b;
let a=5,b="test",c=a+b;
```

Si impossible :
NaN (Not A Number)

Exemple :

```
let a=5,b="test",c=a/b;
```

Division entière

```
let a=5,b=2,c=a/b;          // 2.5
document.write(parseInt(c)); // 2
```

Reste :

```
let a=5,b=2,c=a%b;          // 1
```

Division par 0 :

```
let a=5.5,b=0,c=a/b;    // Infinity
let a=5.5,b=-0,c=a/b;   // -Infinity
```

Attention :

```
2 ** 3 === 8
2 ^ 3  === 1  (XOR)
```

==========
LES OPERATEURS LOGIQUES
==========

Utilisés dans if, else if, for, while.

Résultat : Boolean

Opérateur	Signification
!	Non
&&	Et
||	Ou
==	Egal
!=	Différent
<	Inférieur
>	Supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal

Exemple :

```
let i=4,j=3;
let v;

v=(i<=j);
v=(i==4 && i>2);
```

==========
LES OPERATEURS D'AFFECTATION ET D'INCREMENTATION
==========

Opérateur	Signification
=	Affectation
+=	Addition
-=	Soustraction
*=	Multiplication
/=	Division
%=	Modulo
**=	Puissance
? :	Affectation conditionnelle
++variable	Pré-incrémentation
variable++	Post-incrémentation
--variable	Pré-décrémentation
variable--	Post-décrémentation

Exemples :

```
let i=3;
i++;
i--;
i+=1;
i*=5;
```

Effets de bord :

let a=5,b=1,c=0;
c=++b + a++;

let a=5,b=1,c=0;
c=b++ + a++;

Affectation conditionnelle :

Syntaxe :

<opérande> = <condition> ? <expression1> : <expression2>;

Exemple :

let i=3,j=4,k;
k=i>j ? 10 : -10;

==========
LES FONCTIONS D'ENTREE / SORTIE
==========

Permettent de dialoguer avec clavier et écran.

En environnement Web :

• window.alert(message);

• resultat = window.confirm(message);

message : texte affiché
resultat : Boolean (true si OK)

• resultat = window.prompt(message, defaut);

message : texte affiché
defaut : valeur par défaut
resultat : chaîne saisie ou null

==========
FIN DU DOCUMENT
==========