Algorithmique et JavaScript

Plan du cours

- Variables
- Écriture des valeurs de données en JavaScript
- Fonctions et notion d'objet
- Opérateurs
- Structures algorithmiques de base
  - Conditionnelle
  - Boucle
- Tableaux et objets JavaScript

Plan du cours

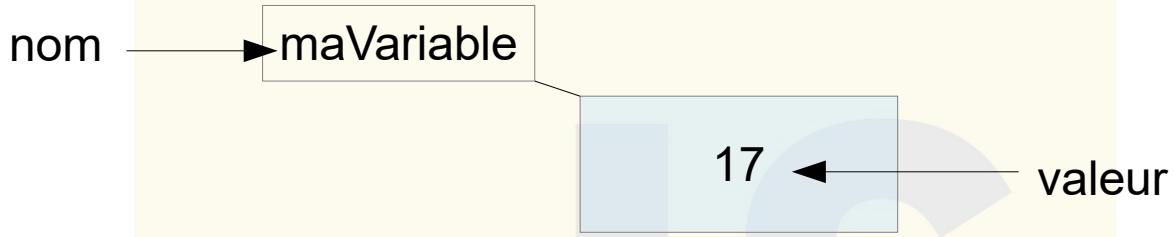
- Variables
- Écriture des valeurs de données en JavaScript
- Fonctions et notion d'objet
- Opérateurs
- Structures algorithmiques de base
  - Conditionnelle
  - Boucle
- Tableaux et objets JavaScript

Notion de variable (1)

- Les données utilisées par le programme sont susceptibles de **changer de valeur**, en fonction des traitements du programme et/ou des actions de l'utilisateur.
- On va les mémoriser dans des objets informatiques appelés « **variables** »
  - Les variables vont nous permettre d'accéder à une donnée par un **nom**, quelle que soit la valeur mémorisée dedans
  - Les variables ont un **type** qui définit le groupe de données dans lequel la variable peut prendre ses valeurs
    - Numérique → entiers, réels (décimaux), complexes
    - Alphanumérique → chaîne de caractères (texte)
    - Booléen → variable à 2 états possibles : VRAI ou FAUX
    - …

Notion de variable (2)

- Une variable peut être vue comme une « boîte » (en réalité un emplacement mémoire dans l'ordinateur) contenant une **valeur**, et identifiée par un **nom**
  - Ex : variable nommée maVariable contenant la valeur numérique 17



  - Remarque : la taille de la « boîte » (c.a.d de l'emplacement mémoire réservé par le programme) dépend du **type** de la variable
- On pourra lire ou écrire dans cette variable au cours du programme
- Attention : **une valeur écrite dans une variable « écrase » la valeur précédente qui est donc perdue**

Créer et affecter la valeur d'une variable en JavaScript

- Déclarer la variable

```
var quantite;    /*déclaration d'un variable nommée « quantité » */
```

  - Attention :
    - **point-virgule à la fin de chaque ligne d'instruction !**
    - Pas d'accent dans les noms de variable, commence par une lettre, en minuscule (par convention)
    - Mot-clé **var** pour tout type de variable : JavaScript adapte automatiquement le type à la valeur affectée à la variable (typage dynamique)

- Affecter une valeur à la variable

```
quantite = 10;  /* la valeur 10 est mise dans la « boîte »
                                    de la variable quantite */
```

Plan du cours

- Variables
- Écriture des valeurs de données en JavaScript
- Fonctions et notion d'objet
- Opérateurs
- Structures algorithmiques de base
  - Conditionnelle
  - Boucle
- Tableaux et objets JavaScript

Écriture des valeurs de données en JavaScript

- Chaînes de caractères (texte)
  - Écrites entre des guillemets **simples** (') **ou doubles** (")
  - Attention à la fermeture non-souhaitée d'une chaîne de caractères !
    - Ex : maChaine = 'Vive l'été !'; /*produit une erreur */

      maChaine = "Trop cool "LOL";-) "; /*idem*/
    - Il faudra écrire "Vive l'été !" et 'Trop cool "LOL";-)'
    - *Remarque : on peut aussi utiliser un \ devant le ' et le " pour qu'ils ne soient pas interprétés comme une fermeture de la chaîne :*
      - *maChaine = 'Vive l\'été !'; /* là, ça marche */*

Écriture des valeurs de données en JavaScript

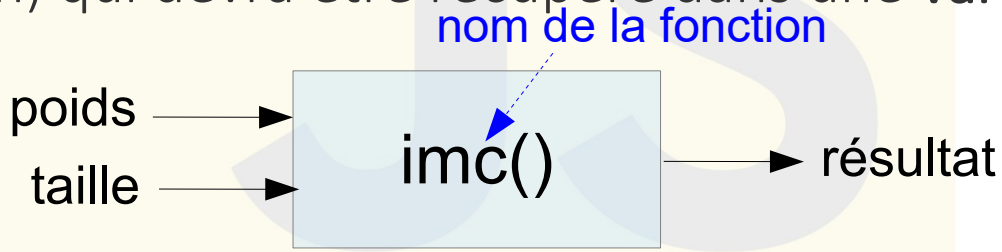
- Nombres
  - Écrits tels quels, sans rien autour : 17 3.145
  - Attention : les nombres décimaux s'écrivent avec un **point** et non une virgule (notation anglo-saxonne)
- Booléens (donnée à 2 états)
  - Valeurs : **true** (VRAI) et **false** (FAUX)
- Une valeur spéciale : null
  - « **null** » représente la valeur d'une donnée « vide » (non renseignée)

Plan du cours

- Variables
- Écriture des valeurs de données en JavaScript
- **Fonctions et notion d'objet**
- Opérateurs
- Structures algorithmiques de base
  - Conditionnelle
  - Boucle
- Tableaux et objets JavaScript

Fonctions

- Rappel :
  - Une fonction est un **sous-algorithme** qui contient une **série d'instructions** exécutées lorsqu'on appelle la fonction
    - On appelle la fonction grâce à son **nom** suivi de **parenthèses**
    - Elle peut éventuellement utiliser des données qu'on lui passe entre les parenthèses (entrée de la fonction)
    - Elle peut éventuellement retourner un résultat (sortie de la fonction) qui devra être récupéré dans une **variable**



    - Appel et récupération du retour :

```
monImc1 = imc(55, 1.69);
```

Fonctions

- Définition d'une fonction dans le code avec le mot clé ***function*** :

```
function imc(poids, taille){/*début de la fonction*/
    var result;
    result = poids/(taille*taille);
    return result; /* return obligatoire si la
                      fonction renvoie un résultat */
}/*fin de la fonction*/
```

Méthodes et propriétés d'un objet

Exemple d'objet

Sportif
prénom poids taille puissance
sentrainer(*nbJours*) faireUnRégime(*nbJours*)

- Objet = structure de programmation avec un nom, des données (propriétés) et des fonctionnalités (méthodes)
  - Il existe de nombreux objets prédéfinis en JavaScript (chaînes de caractères, tableaux, objets représentant un élément HTML, etc...)
- Une méthode est une fonction appelée à partir d'un objet, pour utiliser ou manipuler cet objet
  - Appel d'une méthode (syntaxe) :

```
monSportif.sentrainer(30)
```

- Il est aussi possible d'accéder aux propriétés d'un objet
  - Accès à une propriété (syntaxe) :

```
monSportif.puissance
```

Plan du cours

- Variables
- Écriture des valeurs de données en JavaScript
- Fonctions et notion d'objet
- **Opérateurs**
- Structures algorithmiques de base
  - Conditionnelle
  - Boucle
- Tableaux et objets JavaScript

Les opérateurs mathématiques

- Opérateur d'affectation : **=**
- Opérateurs classiques : **+**, **-**, *****, **/**
- Division entière
  - Pas d'opérateur direct
  - On utilisera une conversion avec la fonction `parseInt()` de JavaScript qui renvoie la partie entière d'un nombre :
    - Ex : **var** result = parseInt(13/5) ; // result vaut 2 ici
- Reste de la division entière (modulo) : **%**
  - Ex : **var** reste = **13 % 5 ;** // reste vaut **3** (parce que 13 = 2*5 + 3)

Les opérateurs mathématiques

- Incrémentation/décrémentation : **++** et **--**
  - **compteur++;** équivalent à **compteur = compteur+1;**
  - **compteur--;** équivalent à **compteur = compteur-1;**
- *Remarques* :
  - On ne peut faire des opérations qu'**entre des données homogènes** (c.a.d. de même type)
    - Ex : "bonjour"-10 est une instruction **incorrecte**
  - il existe des raccourcis syntaxiques pour effectuer certaines opérations et ré-affectation en même temps
    - Ex : **maVar += 10;** est équivalent à **maVar = maVar + 10;**

Les opérateurs sur les chaînes de caractères (texte)

- Opérateur **+** sur du texte
  - Le **+** utilisé entre chaînes de caractères est un opérateur de **concaténation** (chaînes mises bout à bout)
    - Ex : alert("Bonjour "+prenom+" !");
- Transformation d'un texte en nombre avec la fonction *Number()* de JavaScript
    - On peut convertir une chaîne de caractères représentant un nombre en une donnée numérique grâce à la fonction ***Number( )***
      - Ex : Number("12") renverra la donnée numérique 12
    - Parfois utile avec la fonction *prompt()* qui ne renvoie que du texte (attention aux opérations effectuées dessus ensuite!)

Les opérateurs sur les chaînes de caractères (texte)

- Autres opérations sur les chaînes de caractères
  - Il existe de nombreuses fonctions (appelées "**méthodes**") que l'on peut utiliser sur les chaînes de caractères pour les manipuler
  - Exemple : avec `maChaine` une variable contenant une chaîne de caractères
    - découpage selon un caractère : `maChaine.split(',');`
    - extraction d'une sous-chaîne : `maChaine.substring(1,3);`
    - remplacement d'une sous-partie :

      ```
      maChaine.replace('chien', 'chat') ;
      ```

    - ...
  - Voir la doc en ligne : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/String#M%C3%A9thodes_2

Les opérateurs de comparaison et les conditions

- Opérateurs de comparaison
  - Comparaison de valeur :
    - Inférieur, inférieur ou égal : **<, <=**
    - Supérieur, supérieur ou égal : **>, >=**
    - Égalité : **==** (attention à ne pas utiliser le simple = !)
    - Différence : **!=**
  - Remarque : comparaison de valeur ET de type :
    - **===** : teste l'égalité de valeur ET de type
    - **!==** : teste la différence de valeur OU de type

Les opérateurs de comparaison et les conditions

- Conditions
  - Une condition (simple) est un **test** qui compare 2 éléments à l'aide d'un **opérateur de comparaison** et qui renvoie **VRAI** (true) si le test est vérifié, ou **FAUX** (false) si le test n'est pas vérifié.
    - Ex :
      - ```
        var a = 10;
        console.log(a>=5) ;  /* affiche true dans la console */
        ```
      - ```
        var x = 4;
        var y = 3;
        console.log(x==y) ;  /* affiche false dans la console */
        ```
    - Remarque : le double égal == compare uniquement les valeurs, le triple égal === compare à la fois les valeurs et le type de donnée
      - 5 == "5" renvoie true mais 5 === "5" renvoie false

Les opérateurs logiques

- Servent à combiner des valeurs booléennes (VRAI/FAUX), en particulier des conditions
- Le ET logique : **&&**
  - Ex : (note >10) && (note<=12)
  - Le résultat de la combinaison est vrai lorsque **les deux** opérandes sont vraies

val1 && val2	val2 VRAI	val2 FAUX
val1 VRAI	**VRAI**	**FAUX**
val1 FAUX	**FAUX**	**FAUX**

    - *val1* et *val2* sont forcément des valeurs booléennes, par exemple les résultats de 2 tests (conditions)

Les opérateurs logiques

- Le OU logique : ||
  - Ex : (poids >=30) || (taille>110)
  - Le résultat de la combinaison est vrai si **une des deux** opérandes est vraie

val1 || val2	val2 VRAI	val2 FAUX
val1 VRAI	**VRAI**	**VRAI**
val1 FAUX	**VRAI**	**FAUX**

- Le NON logique : !
  - Ex : !(age<12)
  - Inverse la valeur d'une valeur booléenne

val1	VRAI	FAUX
!val1	**FAUX**	**VRAI**

Les opérateurs logiques

- Conditions combinées : combinaison(s) de conditions simples grâce aux opérateurs logiques
  - Exercice :
    - On dispose de 3 variables
      - `solSec` : variable booléenne (`true`/`false`) indiquant si le sol est sec
      - `temperature` : variable numérique indiquant la température
      - `vitesseVent` : variable numérique indiquant la vitesse du vent
    - On condière qu'il y a un risque de feu si :
      - Le sol est sec et la température dépasse 35°
      - Ou si le sol est sec, la température dépasse 20° et la vitesse du vent dépasse 40km/h
    - Écrire en Javascript l'expression indiquant qu'il y a un risque de feu et mémoriser le résultat dans une variable `risqueFeu`. De quel type sera cette variable ?

Plan du cours

- Variables
- Écriture des valeurs de données en JavaScript
- Fonctions et notion d'objet
- Opérateurs
- **Structures algorithmiques de base**
  - **Conditionnelle**
  - **Boucle**
- Tableaux et objets JavaScript

Conditionnelle

- Branchement conditionnel (« si ») : if... else if...else
  - Exécute un bloc d'instructions si une condition est vérifiée, éventuellement un autre bloc d'instructions sinon

```
if (condition1) {
    ... ;
    /* bloc d'instruction exécuté si la condition1 est vérifiée */
}
else if (condition2) {
    ... ;
    /* optionnel : bloc d'instruction exécuté si la condition2 est vérifiée et pas la 1*/
}
else {
    ... ;
    /* optionnel : un bloc d'instruction exécuté si aucune condition au dessus n'est pas vérifiée */
}
```

optionnel

Boucle

- Boucle (ou répétition) : structure de contrôle qui sert à **répéter** un bloc d'instructions
- La boucle while :
  - Le bloc d'instructions de la boucle est répété **tant qu**'une condition est vérifiée
  - Attention : il faut absolument que la condition soit vérifiée à un moment, sinon boucle infinie !

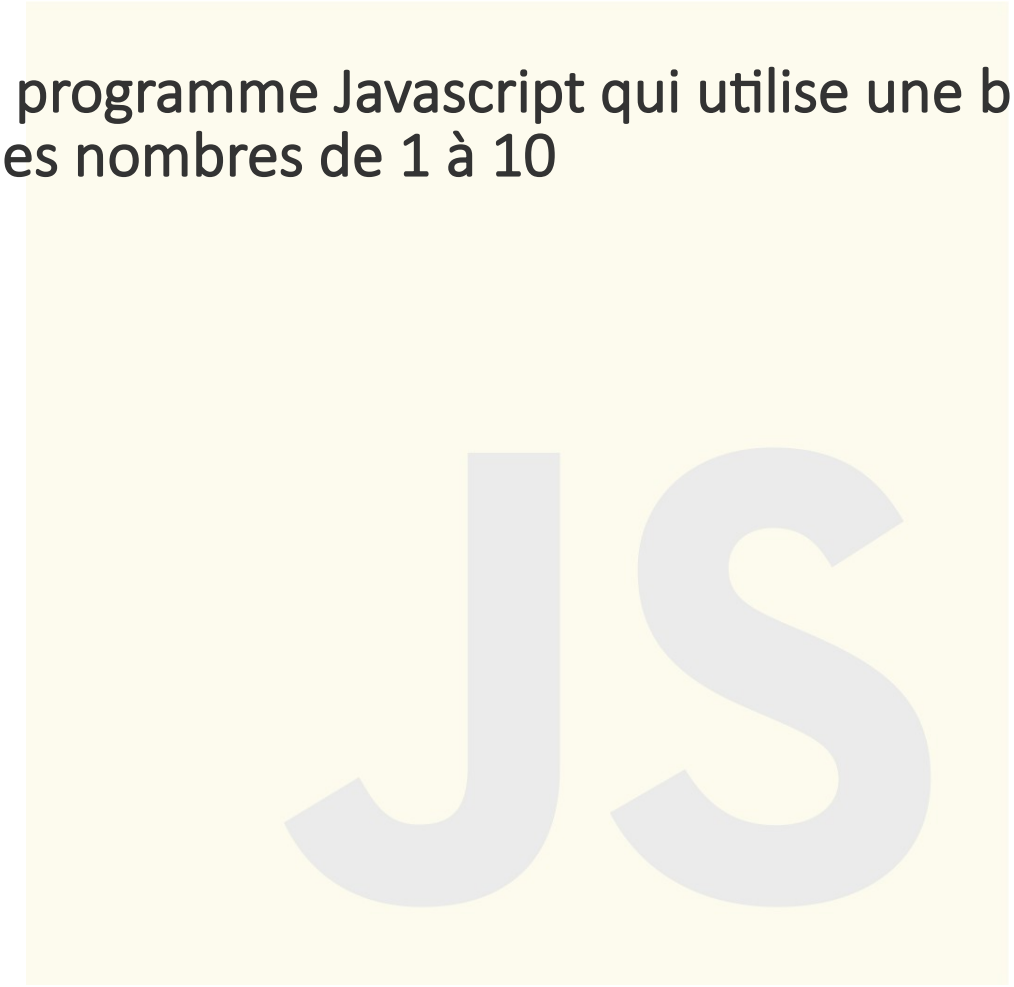
```
while (condition) {
    … ;
    /* bloc d'instructions répété tant que  la condition est vérifiée */
}
```

Boucle while

- Exercice 1 : if et while
  - Créer un programme Javascript qui demande à un utilisateur de fournir une note au clavier et indique la mention correspondant :
    - Note inférieure à 10 → Recalé
    - Note entre 10 et 12→ Passable
    - Note entre 12 et 14→ Assez bien
    - Note entre 14 et 16→ Bien
    - Note supérieure à 16 → Très bien
  - Le programme devra vérifier que la note fournie est comprise entre 0 et 20 et demander une nouvelle note tant que ce n'est pas le cas

Boucle while

- Exercice 2 : 10 tours de boucles avec while
  - Créer un programme Javascript qui utilise une boucle while pour afficher les nombres de 1 à 10



Boucle

- La boucle for
  - Répéter une boucle un nombre de fois défini *N*

```
for (var i = 0 ; i < N; i++) {
    … ;
    /* bloc d'instructions répété N fois */
    /* i est une variable qui sert de compteur de boucle */
}
```

  - *N* doit être un entier ou une variable contenant un entier

Boucle for

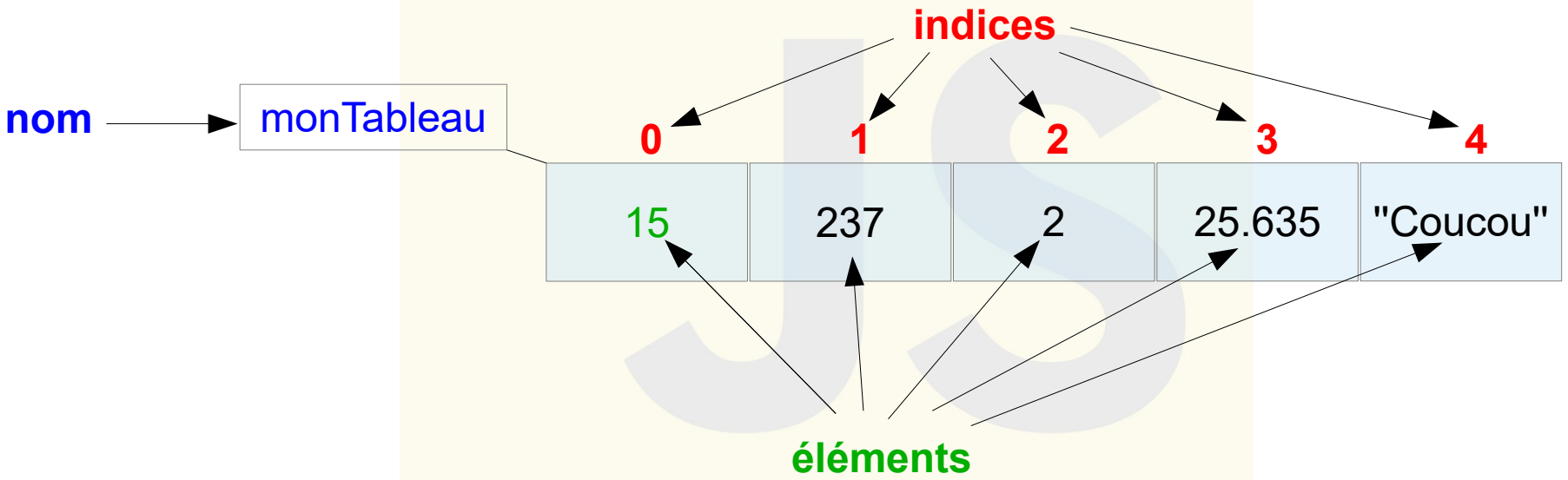
- Exercice : 10 tours de boucles avec for
  - Créer un programme Javascript qui utilise une boucle for pour afficher les nombres de 1 à 10

Plan du cours

- Variables
- Écriture des valeurs de données en JavaScript
- Fonctions et notion d'objet
- Opérateurs
- Structures algorithmiques de base
  - Conditionnelle
  - Boucle
- **Tableaux et objets JavaScript**

Les tableaux

- Qu'est-ce qu'un tableau en programmation ?
  - Séquence de « boîtes » pouvant contenir une série d'autres éléments
  - Chaque case du tableau est accessible par un **indice entier**



  - Attention : les indices commencent à 0 !!!

Les tableaux en JavaScript

- Déclaration et affectation
  - 2 façons :
    - `var person = ["John", "Doe", 46];` /* on utilise la notation entre crochets */
    - `var cars = new Array("Saab", "Volvo", "BMW");` /* on crée explicitement un nouvel objet tableau (Array) */
- Accès aux éléments du tableau
  - On y accède en lecture et écriture à partir du **nom du tableau** et grâce aux **indices** mis entre **crochets [...]**
    - **person[2]** renverra **46, cars[0]** renverra **"Saab"**
    - **person[2] = 52;** affecte la valeur **52** à l'élément d'indice **2** du tableau **person**, l'ancienne valeur est écrasée (perdue)

Les tableaux en JavaScript

- Longueur d'un tableau :
  - **person.length** est une propriété des tableaux qui renvoie le nombre d'éléments du tableau (i.e. dernier indice+1)
    - *Remarque : **.length** est aussi utilisable sur les chaînes de caractères pour renvoyer leur longueur*
- Ajouter un nouvel élément dans un tableau (en dernière position)
  - **cars.push("Mercedes") ;** /* rajoute l'élément "Mercedes" en dernière position du tableau **cars** */

Les tableaux en JavaScript

- Parcours des éléments d'un tableau
  - On utilise une boucle **`for`**
    - Le compteur de boucle `i` va servir d'indice dans le tableau pour accéder aux élément du tableau **un à un**
    - On utilise la longueur du tableau pour savoir combien de tours de boucle on doit faire

```
for (var i = 0 ; i < monTableau.length ; i++) {
    var elem =  monTableau[i];  // accès à l'élément courant d'indice i
    …
    …
}
```

Les tableaux en Javascript

- **Exercice 1 : affichage un à un des éléments d'un tableau**

  1) Créer un programme Javascript qui commence par déclarer un tableau contenant 10 nombres, dans la variable `mon_tableau`

  2) Utiliser ensuite une boucle `for` pour afficher tour à tour les éléments de ce tableau dans la console (sans utiliser la valeur 10 dans la déclaration de cette boucle).

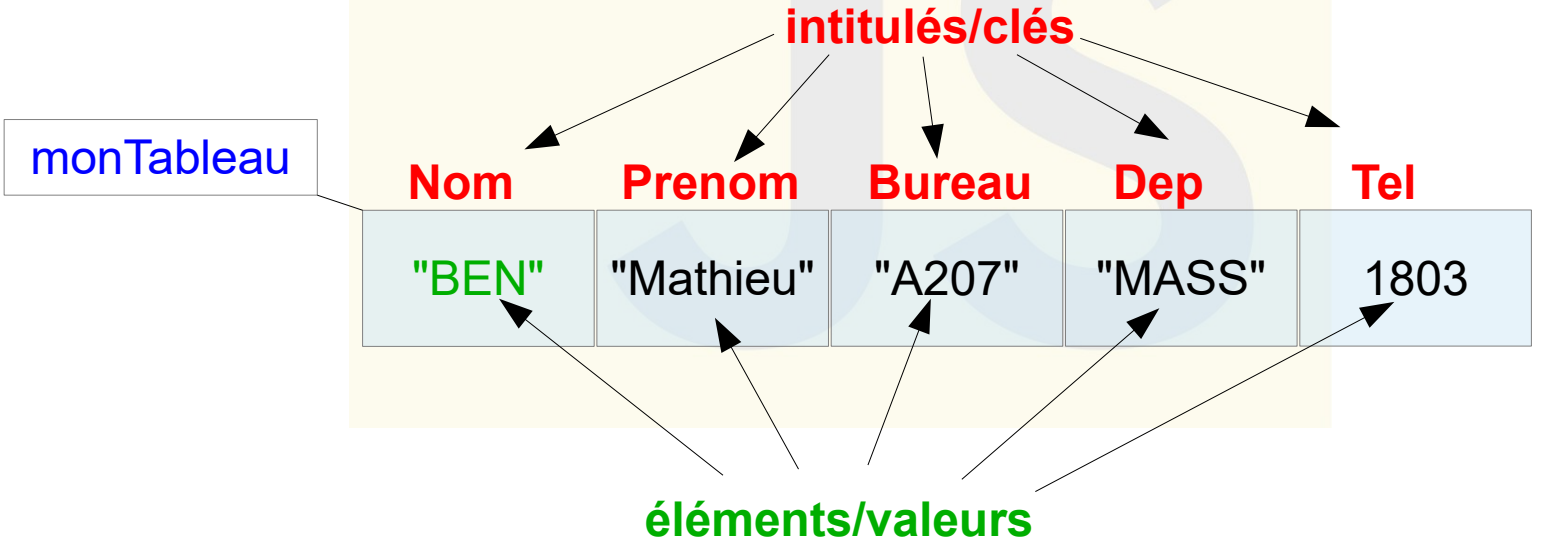
- **Exercice 2 : remplissage d'un tableau élément par élément**

  1) Créer un programme Javascript qui commence par déclarer un tableau vide dans la variable `mon_tableau`

  2) Utiliser ensuite une boucle `for` pour remplir le tableau `mon_tableau` par 6 mots entrés un à un au clavier par l'utilisateur.

Les tableaux associatifs ou objets

- Appelés « objets » *(Object)* en JavaScript
- Qu'est-ce qu'un tableau associatif ?
  - Ensemble de « boîtes » regroupées sous un même nom
  - Chaque case du tableau est accessible par un **intitulé (unique)** qui peuvent par exemple représenter des **propriétés** d'un objet
    - *Remarque : cet intitulé unique s'appelle une **clé**. Un tableau associatif contient donc un ensemble de paires clé/valeur*



Utilisation des objets en JavaScript

- Déclaration et affectation
  - 2 façons :
    - `var myCar = {type:"Fiat", model:500, color:"white"};` /* on utilise la notation entre accolades avec une série de paires clé:valeur */
    - `var myCar = new Object({type:"Fiat", model:500, color:"white"});` /* on crée explicitement un nouvel objet (Object) */
- Accès aux éléments (lecture / écriture)
  - 2 façons :
    - Avec des crochets : `myCar["type"]` / `myCar["type"] = "Volvo";`
    - Avec le point : `myCar.color` / `myCar.color="red";`

Les objets en Javascript

- **Exercice : affichage des propriétés d'un objet**

1) Créer un programme Javascript qui commence par déclarer un objet `mon_article` ayant les propriétés suivantes :
   - `taille : 40`
   - `couleur : "bleu"`
   - `marque : "Fashion"`
   - `prix : 55.99`
   - `tags : ["jean", "homme", "hiver"]`
2) Afficher une à une les valeurs des propriétés de l'objet `mon_article`
3) Afficher ensuite les éléments du tableau associé à la propriété `tags` de `mon_article`