Les variables

Table des matières

I. Contexte

Durée : 1 h

Environnement de travail : Repl.it

Prérequis : Bases sur l'objet `console`

Contexte

Durant le développement d'une application, il est nécessaire de stocker des valeurs, afin de les manipuler, d'éviter d'avoir à les réécrire plusieurs fois, ou simplement de les réutiliser plus tard. Afin de simplifier ces traitements, nous allons utiliser des variables.

II. Définition et utilité d'une variable

Objectifs

- Comprendre à quoi sert une variable
- Apprendre les règles de nommage d'une variable

Mise en situation

Nous allons voir à quoi servent les variables et les bonnes pratiques en ce qui concerne leur nommage.

Utilisation d'une variable

Une variable peut être vue comme une sorte de conteneur servant à stocker des informations de manière temporaire afin de pouvoir la réutiliser ultérieurement.

Elle peut évoluer au fil du temps, d'où le nom "variable", et permet de stocker de manière simple et efficace différents types d'informations, et de les manipuler. On définit une variable à l'aide des mots-clés `var`, `let` ou `const`.

Exemple

Dans un programme, on souhaite récupérer le nom de l'utilisateur courant afin de l'afficher à l'écran. Utiliser une variable va nous permettre de traiter et manipuler le résultat sans avoir besoin de le connaître à l'écriture du script.

Dans cet exemple, on crée une variable “nomUtilisateur” avec le mot-clé let (la notion de mots-clés sera détaillée plus bas dans ce cours). Pour qu’elle stocke un nom d’utilisateur personnalisé, nous allons utiliser la méthode prompt(). La méthode prompt() ouvre une boîte de dialogue sur votre navigateur qui permet à l’utilisateur de saisir une valeur ; cette dernière sera ensuite stockée dans la variable.

```
1
2 // On stocke le nom donné par l'utilisateur dans une variable nommée nomUtilisateur
3 let nomUtilisateur = prompt("Quel est votre nom ?");
4
5 // Affichera le nom vous avez entré dans la boîte de dialogue
6 console.log(nomUtilisateur);
```

Stocker une valeur dans une variable permet aussi d'éviter de la réécrire plusieurs fois, et facilite ainsi toute évolution future.

Exemple

Dans un blog, le nombre maximum de posts affichés sur un écran passe de 4 à 8. Il suffira de modifier la variable dans laquelle cette valeur est stockée afin que le changement soit effectif partout.

```
let itemsPerPage = 4;
console.log(itemsPerPage);
// Affichera 4;

itemsPerPage = 8;
console.log(itemsPerPage);
// Affichera 8


```

La variable affiche la valeur qu'elle stocke. Il n'y a donc pas besoin de la réécrire si elle est utile à plusieurs endroits et, en cas de changement, il suffit de la modifier une seule fois à un endroit.

Syntaxe **À retenir**

- Une variable peut donc être vue comme un nom symbolique permettant de faire référence à une valeur donnée qui peut être modifiée et manipulée. Elle doit être nommée de manière claire afin de faciliter la lecture du code.

Complément

Variable

III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[1]

Question

[solution n°1 p.19]

Dans le contexte d'une page "Profil utilisateur", créons un ensemble de variables qui portent les données suivantes :

- Nom
- Prénom
- Âge
- Adresse e-mail
- Numéro de téléphone

Indice :

Le mot-clé à placer devant une variable est **`let`**, et chaque variable devrait être définie sur une ligne séparée.

1 https://repl.it/

IV. Déclaration et assignation d'une variable

Objectifs

- Déclarer et initialiser une variable
- Comprendre la différence entre `var` et `let`
- Maîtriser la réaffectation et l'assignation par transitivité

Mise en situation

Afin de pouvoir utiliser notre variable, nous allons voir comment la déclarer et lui affecter une valeur.

Déclarer une variable

Avant d'initialiser une variable, il faut d'abord la déclarer. Pour cela, il suffit d'utiliser l'un des mots-clés permettant de déclarer une variable, suivi du nom qui lui est donné

Il est possible de déclarer des variables de plusieurs façons :

- En déclarant la variable et en lui assignant une valeur aussitôt via l'opérateur "=", qui n'agit alors pas comme un opérateur d'égalité, mais bien comme un opérateur d'assignation.
- En déclarant la variable sans lui attribuer de valeur directement. Elle aura alors pour valeur `undefined`, jusqu'à ce qu'une autre valeur lui soit attribuée.

Exemple

```
var maVariable = "ma variable";
```

Dans cet exemple, on a affecté la valeur à droite de l'opérateur d'assignation (=) à la variable déclarée à gauche.

```
var maVariableSansValeur;

console.log(maVariableSansValeur) // Le résultat affiché sera undefined
maVariableSansValeur = 10;
console.log(maVariableSansValeur) // Le résultat affiché sera 10

```

Dans cet exemple, on a tout d'abord déclaré la variable sans lui assigner de valeur. Elle valait donc `undefined` (non défini). On lui a ensuite assigné la valeur 10 : à ce moment-là, il ne faut pas mettre le mot `var`, car l'ordinateur comprendrait alors que nous souhaitons déclarer à nouveau la variable.

Avant d'essayer d'accéder à une variable, il faut qu'elle ait été déclarée au préalable.

Méthode **Nommage d'une variable**

Une variable se compose d'une clé unique et d'une valeur. Cette clé ne peut pas être constituée de n'importe quel caractère alphanumérique, en effet, elle doit commencer par une lettre, un underscore ou un signe dollar.

Les noms de variables sont sensibles à la casse, ce qui signifie que la variable nommée myVariable est différente d'une autre variable nommée myvariable. Il est impossible d'avoir deux variables avec le même nom dans le même bloc ou dans le bloc parent et le bloc enfant.

Le nommage est libre, à l'exception de certains noms réservés, tels que boolean, abstract, return, var, etc. Pour rendre le code plus facile à lire, il doit être aussi significatif que possible.

Par exemple, pour la variable qui servira à stocker la date de naissance de l'utilisateur, il est judicieux de la nommer dateNaissance au lieu de variableAtHasard, afin qu'en lisant le code, on puisse facilement comprendre à quoi elle fait référence.

De manière générale, les variables sont nommées en utilisant la syntaxe camelCase, ce qui signifie que la première lettre du premier mot est en minuscule, et la première lettre de chaque mot suivant est en majuscule.

Par conséquent, le nom de la variable sera composé de la première lettre minuscule suivie d'une lettre majuscule, par exemple : dateNaissance.

Remarque **La notion de bloc**

On appelle bloc de code tout code étant encapsulé et autonome. Nous verrons plus tard les fonctions JavaScript, mais en substance, il est possible de créer des parties de code autonomes (blocs) au sein d'un même fichier ou d'un même bloc. Vous trouverez dans le paragraphe suivant une explication plus détaillée.

Exemple

Voici quelques exemples de noms de variables syntaxiquement corrects et incorrects :

```
let nommageOk; // Nom OK, caractères alphanumériques
let nommage-KO; // Nom KO, tiret non compris dans les caractères autorisés
let #symboles; // Nom KO, dièse non compris dans les caractères autorisés
let _variableAvecTiretBas // Nom OK, caractères alphanumériques et tiret bas autorisés dans le nommage
let boolean // Nom KO, boolean est un nom réservé
let mavariableavecunnomok // Nom OK, mais il vaut mieux préférer le lowerCamelCase qui facilite la lecture du nom de la variable
```

Les mots-clés var et let

Différents mots-clés permettent de déclarer une variable :

- Le mot-clé `var` : il a une portée limitée à la fonction et non au bloc courant, et peut être déclaré sans valeur initiale.

Les variables déclarées avec le mot-clé var peuvent être déclarées après la première utilisation. En effet, les déclarations de variables définies avec le mot-clé var sont traitées avant le reste du code en JavaScript. Après l'introduction de let, var n'a plus de sens, car la fonction de let est essentiellement la même que var, mais l'effet est meilleur. Préférez toujours utiliser let pour éviter les conflits dans le code.

- Le mot-clé `let` a une portée limitée au bloc courant (accolades les plus proches) et peut être déclaré sans valeur initiale.

Un bloc de code est, de façon générale, tout ce que vous verrez entre deux accolades ({ }).

La portée (ou scope en anglais) est la portion de code dans laquelle une variable existe. La portée est différente selon les mots-clés utilisés lors de la déclaration. Cette notion sera vue plus en détails en suivant le parcours Javascript, mais il est intéressant de connaître les grandes lignes avant d'y être confronté.

Retenez que let et const ont la même portée et que var a une portée légèrement supérieure, sans pour autant que cela soit toujours un avantage.

```
début portée de fonction  {
  début portée de bloc {
    //code
    //code
  fin de portée de bloc }
  //code
fin de portée de fonction }
```

La déclaration d'une variable sans le mot-clé `var` ou `let` créera une variable d'une portée globale non déclarée (au niveau de l'élément `window`), mais ce n'est pas une pratique recommandée, car elle peut avoir des effets inattendus.

Il est donc préférable de définir la variable en question au niveau du bloc dans lequel elle doit être accessible.

L'utilisation des mots-clés `var` et `let` n'est utile que lors de la déclaration des variables : il suffit d'utiliser leur nom pour les manipuler une fois qu'elles ont été déclarées.

Réaffectation d'une variable

Pour affecter une nouvelle valeur à une variable déjà initialisée, il suffit de se servir de nouveau de l'opérateur d'assignation "=".

La nouvelle valeur attribuée à la variable va écraser l'ancienne valeur.

```
var a = "Une chaine de caractère";

console.log("a : ", a) ;
// Affichera "a  : Une chaine de caractère"

a = 10;
console.log("a : ", a);
// Affichera "a  : 10", l'ancienne valeur a été écrasée.
```

Assignation par transitivité

Si on assigne une valeur à une variable en lui assignant une autre variable comme valeur, alors la variable en question prendra la valeur de la seconde variable au moment de l'assignation.

```
var a = 0;
var b = a;
console.log("a : ", a, "& b : ", b) ;
// Affichera "a  : 0 & b : 0", b vaut la valeur de a au moment où l'assignation est faite.

a = 10;
console.log("a : ", a, "& b : ", b);
// Affichera "a  : 10 & b : 0", b ne change pas de valeur avec a.
```

Syntaxe **À retenir**

- Il y a donc plusieurs manières de déclarer une variable, selon la portée souhaitée : `let` pour une portée propre au bloc, et `var` pour une portée propre à la fonction dans laquelle elle est déclarée.
- Afin de respecter les bonnes pratiques, il est préférable de choisir une méthode au début d'un projet et de s'y tenir par la suite, de manière à garder une cohérence dans le code produit.

Complément

Déclaration de variables

V. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[1]

1 https://repl.it/

Question 1

[solution n°2 p.19]

Nous souhaitons recueillir un peu plus d'informations sur notre utilisateur. Déclarez les variables qui serviront à stocker :

- L'adresse postale : 1, rue de la paix, 75000 Paris
- Le pays de naissance : France
- Nombre d'enfants : 2

Question 2

[solution n°3 p.19]

Heureux événement pour notre utilisateur, il vient d'avoir un nouvel enfant ! Mettez à jour la variable qui contient le nombre de ses enfants. Il a également déménagé au 32 rue de Vaugirard, 75000 Paris : modifiez son adresse postale.

VI. Échanger deux mots

Objectifs

- Comprendre le concept de variable
- Échanger le contenu de deux variables

Mise en situation

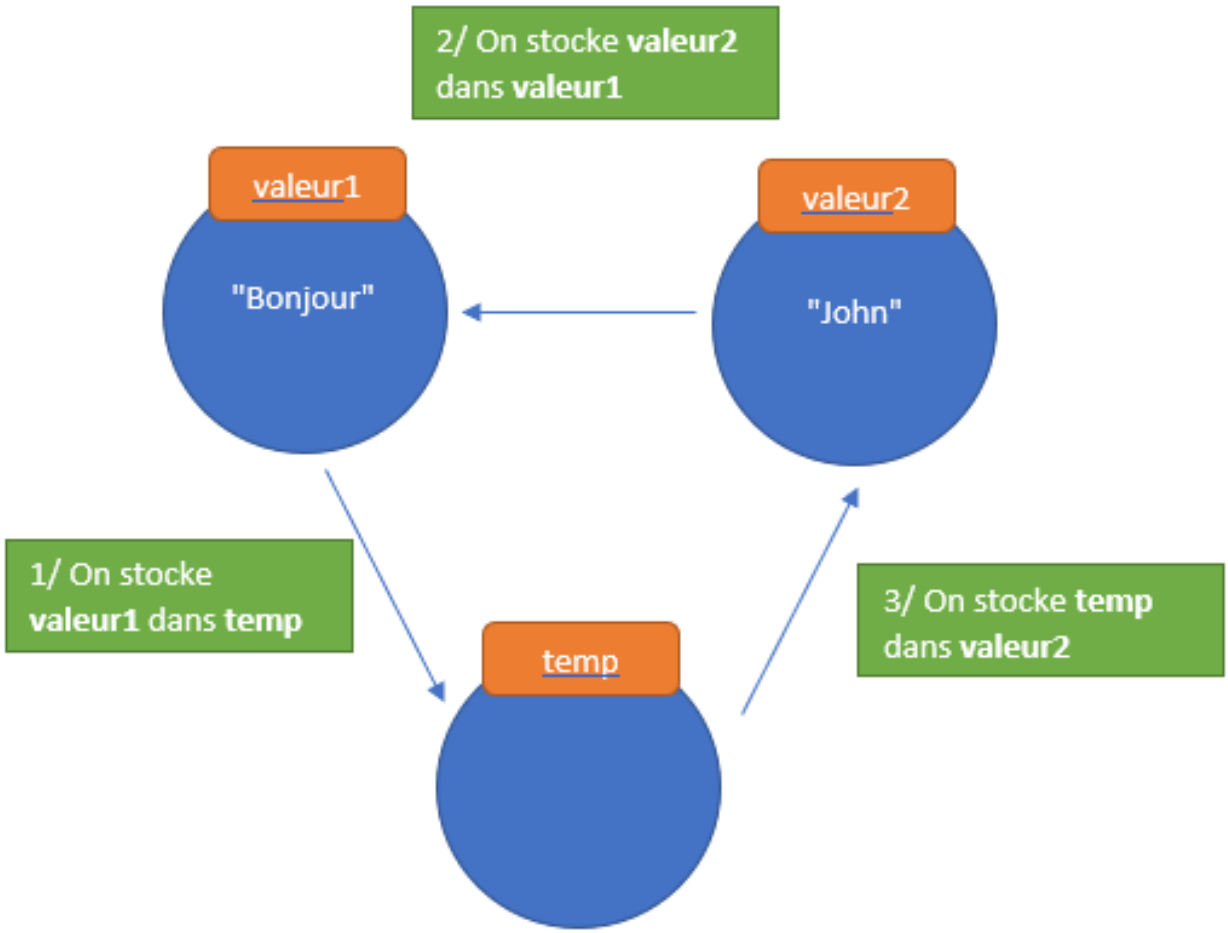
En tant que développeur, nous allons manipuler des valeurs auxquelles nous aurons, au préalable, donné un nom. Cela s'appelle des variables. Ainsi, si nous devons manipuler un nom utilisateur, nous allons créer une variable appelée username qui contiendra cette donnée.

On peut voir les variables comme une petite boîte contenant une valeur. Chaque boîte possède une étiquette sur laquelle on met le nom de son contenu, et celles-ci ne peuvent contenir qu'une seule et unique valeur.

Voyons comment cela se traduit en JavaScript, en analysant un algorithme permettant d'échanger le contenu de deux variables.

Méthode **Intervertir deux mots**

Pour échanger le contenu de deux variables, nous allons procéder de la même manière que pour échanger le contenu de deux verres d'eau : nous allons utiliser une variable intermédiaire pour stocker temporairement la valeur de la première variable.



Voici le code JavaScript permettant cette opération :

```
// Le but est d'intervertir les valeurs de ces deux variables
let value1 = "Bonjour";
let value2 = "John";


let temp = value1; // La valeur de value1 est assignée à une nouvelle variable "temp"

value1 = value2; // On assigne "John" à value1
value2 = temp; // On assigne la valeur de temp à value2
```

En JavaScript, le mot -clé `let` permet de créer de nouvelles variables. Ici, on crée tout d'abord les variables `value1` et `value2` qui contiennent respectivement *Bonjour* et *John*.

Définition **La trace d'un algorithme**

Pour mieux comprendre ce qu'il se passe pendant l'exécution d'un algorithme, il est possible de faire une **trace**. Une trace est l'état de toutes les variables à chaque étape de l'algorithme.

Voici la trace de l'algorithme ci-dessus :

Etape	value1	value2	temp
let value1 = "Bonjour";	"Bonjour"		
let value2 = "John";	"Bonjour"	"John"	
let temp = value1;	"Bonjour"	"John"	"Bonjour"
value1 = value2;	"John"	"John"	"Bonjour"
value2 = temp;	"John"	"Bonjour"	"Bonjour"

Remarque

Contrairement aux verres d'eau, une variable n'est pas "vidée" lorsque l'on met son contenu dans une autre variable. À la place, sa valeur est copiée.

Syntaxe **À retenir**

- Les **variables** sont le fondement de la programmation informatique. Elles sont composées d'un **nom** leur permettant d'être manipulées et d'une **valeur**.
- En JavaScript, elles sont déclarées avec le mot-clé `let`.
- Pour suivre l'exécution d'un programme, il est possible de réaliser une **trace**, c'est-à-dire d'écrire le contenu de chaque variable à chaque étape de notre algorithme.

Complément

Trace, dans Wikipédia[1]

VII. Exercice : Appliquez la notion

Question

[solution n°4 p.19]

Nous avons vu comment échanger deux nombres stockés dans deux variables différentes.

Vous disposez de l'algorithme suivant, avec deux variables que l'on souhaite intervertir. Complétez la trace suivante en remplaçant les X.

```
let value1 = 'Caroline';
let value2 = 'Bonsoir';
let temp = value1;
value1 = value2;
value2 = temp;
```

Étape	variable1	variable2	temp
let value1 ='Caroline';	X		
X	Caroline	X	

1 https://fr.wikipedia.org/wiki/Trace_(informatique)

let temp = value1;	X	Bonsoir	X
value1 = X;	Bonsoir	X	X
value2 = X	X	Caroline	Caroline

VIII. Les types de variables

Objectifs

- Connaître les différents types de variables et savoir les utiliser
- Comprendre pourquoi JavaScript est un langage dit à typage faible

Mise en situation

Nous savons maintenant comment déclarer une variable et lui affecter une valeur. Dans ce cours, nous allons voir quels sont les différents types de valeurs que l'on peut affecter à une variable, et leurs particularités.

Les différents types de variables

Les variables peuvent être des types suivants :

- **`object`** : le type `object` regroupe plusieurs types d'objets fréquemment utilisés, principalement `Object`, `Array`, `Date`.
  - Un objet `Object` est composé d'un ensemble de couples clé/valeur. Les clés sont des chaînes de caractères, et les valeurs peuvent être de n'importe quel type.
  - Un objet `Array` est un tableau de valeurs dont les données sont indexables à partir de 0 (le premier élément correspondra à l'élément 0 du tableau).
  - Un objet `Date` retourne une date.

```
let exempleObjetPersonne = {
  "nom": "Paul",
  "age": 20,
  "informations": ["information1", "information2"]
}

// On accède aux différents éléments avec nomDeLaVariable.nomDeLaClé
console.log(exempleObjetPersonne.nom, " a ", exempleObjetPersonne.age, " ans");
// Affichera "Paul a 20 ans"
```

```
let exempleArray = ["valeur1", "valeur2", 3];

// On accède à l'élément "valeur1" comme ceci : exempleArray[0] car l'élément "valeur1" est à l'index 0 de l'array.

console.log("2ème élément : ", exempleArray[1]);
// Affichera "2ème élément : valeur2"
```

```
let date = new Date();
// date sera égale à la date du jour
```

- **`string`** : pour les chaînes de caractères, c'est une séquence de caractères qui représente une valeur textuelle. Tout ensemble de caractères entouré de guillemets (") ou d'apostrophes (')est considéré comme une chaîne de caractères, y compris les chiffres .

  Si la chaîne de caractère contient une apostrophe ou un guillemet alors que celui-ci est aussi le délimiteur choisi, il faudra échapper ce caractère en le précédant par un antislash (\) ou changer de délimiteur.

```
let maChaineDeCaractereGuillemet = "Je m'appelle Paul."
let maChaineDeCaractereApostropheEchappe = 'Je m\'appelle Paul.' // Echappement sur le '
let maChaineDeCaractereGuillemetEchappe = "Je m'appelle \"Paul\"." // Echappement sur le "
```

- **`number`** : comprend tous les nombres réels et décimaux. Il faut cependant faire attention : le typage simple des nombres en JavaScript est une exception. La plupart des autres langages, avec un typage plus strict, différencient les nombres réels (`int`) et les décimaux (`float, long, double`).
- **`symbol`** : pour les symboles, est apparu avec ECMAScript 2015 (ES6). Ce type représente des données immuables et uniques, permettant d'éviter les problèmes de doublons dans certains cas (clés d'un objet, par exemple).
- **`null`** : un mot-clé spécial pour indiquer une valeur nulle au sens informatique. Étant donné que JavaScript est sensible à la casse, seul `null` est une valeur correcte. Toute autre manière de l'écrire (*Null, NULL*...) serait incorrecte.
- **`undefined`** : pour les valeurs non définies.
- **`boolean`** : ce type ne contient que deux valeurs : *true* (vrai) et *false* (faux). Pour que cette valeur soit bien stockée comme un booléen, il ne faut pas l'entourer de guillemets ou autre. C'est un type de données à part entière, et non une chaîne de caractères.

```
var exempleBoolean = 1000 > 5;
console.log(exempleBoolean);
// Affichera true
```

JavaScript, un langage à typage faible et dynamique

JavaScript est un langage dit à typage faible et dynamique : cela signifie qu'il n'est pas nécessaire de préciser le type des variables qu'on déclare. Le langage va de lui-même détecter le type de la valeur renseignée, et nous pourrons effectuer les traitements propres à son type grâce à cela. Cela signifie qu'on pourra stocker différents types de valeurs dans une même variable au fil du temps. Il est également possible de modifier le type d'une variable déjà définie lors de l'exécution d'un script.

```
let x = 10;
console.log(typeof x);
// Affichera "number"
x = "10";
console.log(typeof x);
// Affichera "string"
// Le type de la variable a été modifié dynamiquement.
```

Tout opérateur renvoie un résultat, quitte à transformer le type de l'un des éléments.

Si l'assignation d'une variable mélange chaînes de caractères et valeurs numériques ainsi que l'opérateur "+", JavaScript convertira les nombres en chaînes de caractères.

```
let x = "La réponse est "
x = x + 25;
console.log(x);
// Affichera "La réponse est 25" comme une chaine de caractères
```

Ce n'est pas le cas pour les autres opérateurs, pour lesquels on obtiendra ce type de résultats :

```
let x = "37" - 7;
console.log(x);
// Affichera 30 en tant que nombre : convertira la chaîne de caractère en nombres pour effectuer l'opération
x = "37" + 7;
console.log(x);
// Affichera 377 en tant que chaîne de caractères : incluera le 7 à la suite de la chaîne de caractères
```

Rappel

La concaténation consiste à chaîner des variables ou des propriétés les unes à la suite des autres, afin de créer un `string` unique. Cette notion est expliquée plus précisément dans le cours sur les opérateurs logiques.

Syntaxe **À retenir**

- JavaScript est un langage à typage faible et dynamique, qui détecte de lui-même le type de données stockées dans une variable. Il est possible, grâce à des fonctions natives, de caster des variables, c'est-à-dire les passer d'un type à un autre type.

Complément

Types de variables[1]

IX. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[2]

Question

[solution n°5 p.20]

Nous devons calculer le total du panier de l'utilisateur. Nous avons dans la variable `productsTotalValue` la valeur totale des produits, et `specialOffer` représente la valeur de la réduction à appliquer.

```
let productsTotalValue = "154.38";
let specialOffer  = "10";

let cartTotalValue = ?;
```

X. Les constantes en JavaScript

Objectifs

- Maîtriser l'utilisation des constantes

Mise en situation

Nous allons maintenant voir à quoi servent les constantes, en quoi elles sont différentes des variables classiques et comment les utiliser, puis ce qu'est l'immutabilité et en quoi elle est utile en programmation fonctionnelle.

L'utilisation des constantes

Une constante est une variable qui ne peut être définie qu'une seule fois, et il est impossible de modifier sa valeur une fois définie. Elle a forcément une valeur et ne peut donc pas être non définie.

Elle est définie avec le mot-clé `const`.

1 Types de variables (MDN Web Docs)
2 https://repl.it/

Le nommage de l'identifiant répond aux mêmes critères que pour une variable classique, et il est impossible de déclarer une constante avec le même nom qu'une autre variable ou fonction dans la même portée. Même si les noms de constantes en minuscules fonctionnent parfaitement, celles-ci sont par convention écrites en majuscules.

Sa portée est limitée au bloc courant : si elle doit être accessible de manière globale, il faut la déclarer hors de toute fonction.

Une constante ne peut pas changer de valeur grâce à une affectation ni être déclarée à nouveau pendant l'exécution du script.

```
const MACONSTANTE;
// Retourne une erreur car aucune valeur n'est attribuée
const MACONSTANTE = "ma constante";
// Pas d'erreur car une valeur est bien attribuée
```

Syntaxe **À retenir**

- Les constantes sont une forme de variables dont la valeur est toujours définie et n'est pas modifiable, existant dans la portée du bloc dans lequel elles sont définies, tout comme le mot-clé `let`.

Complément

Constantes

XI. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

1

Question 1

[solution n°6 p.20]

Lors de la récupération des informations d'un utilisateur, nous stockons son nom et son statut (en ligne, hors ligne, en train de faire une activité, etc.) dans deux variables. Écrivez ces deux variables.

Indice :

Le nom de l'utilisateur ne changera jamais, son statut peut évoluer au fil du temps.

Question 2

[solution n°7 p.20]

Le code suivant est-il correct ?

```
const populationFrance ;
populationFrance = 67390000 ;
```

Question 3

[solution n°8 p.20]

Dans l'exemple ci-dessous, les nouvelles déclarations de la constante x sont-elles autorisées ?

```
const x = 2 ;

{
  const x = 3;
}

```

1 https://repl.it/

```
{
  const x = 4;
}
```

XII. Essentiel

XIII. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°9 p.20]

Exercice

Une variable sert à stocker des informations.

- ○ Vrai
- ○ Faux

Exercice

Une variable peut se définir par les mots-clés...

- ☐ var
- ☐ this
- ☐ const
- ☐ let

Exercice

Quels sont les types que la console affichera ?

```
let myVariable = '10';
console.log(typeof myVariable);
myVariable = Number(10)
console.log(typeof myVariable);
myVariable = new Date()
console.log(typeof myVariable);
myVariable = new Array()
console.log(typeof myVariable);
```

- ☐ string
- ☐ boolean
- ☐ number
- ☐ array
- ☐ object
- ☐ date

Exercice

Qu'affichera la console ? Répondre

```
const var1 = 'le résultat est '
const var2 = 3
const var3 = 4
console.log(var1 + (var2 + var3)+ ' ou ' + var2 + var3)
```

Exercice

L'utilisation du mot-clé `var` est recommandé en JavaScript.

- Vrai
- Faux

Exercice

Ce code est correct.

```
let myVariable = 25;
myVariable = 'test';
myVariable = false;
```

- Vrai
- Faux

Exercice

À combien est égale la variable `counter` ?

```
let counter = 5;
for (let i = 0; i < 10; i++){
    let counter = i;
    // counter est égal à 9 lorsque la boucle se termine
}
console.log(counter )
```

Exercice

Quelle nomenclature correspond à une bonne pratique ?

- const mySupervariable ;
- const mysupervariable ;
- const MYSUPERVARIABLE ;
- const mySuperVariable ;

Exercice

Dans le code suivant, `name` est une variable.

```
const personne = {
   name: 'Paul',
   age: 12
};
```

- Vrai
- Faux

Exercice

Ce code est juste.

```
1 const var1 = var2 = var3 = 4;
```

- ○ Vrai
- ○ Faux

B. Exercice : Défi

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

1

Question

[solution n°10 p.22]

Pour simplifier sa comptabilité, un marchand de légumes souhaite créer un petit script lui permettant de rentrer le nombre de pommes et poires vendues dans la journée, et de récupérer le chiffre d'affaires par fruit, ainsi que le chiffre d'affaires total en euros.

Une pomme coûte 0,32 €.

Une poire coûte 0,44 €.

Vous devrez stocker dans un premier temps le prix d'une pomme et d'une poire dans des variables, puis permettre à l'utilisateur de rentrer le nombre de poires et de pommes vendues dans la journée.

Enfin, une fenêtre d'alerte devra s'afficher indiquant le nombre de pommes et de poires vendues dans la journée, ainsi que le chiffre d'affaires par produit et le chiffre d'affaires total.

Indice :

Pour indiquer les quantités, vous pouvez utiliser la méthode `prompt()`.

Pour afficher une alerte, vous pouvez utiliser la méthode `alert()`.

Attention à l'affichage du chiffre d'affaires : on parle en euros, l'affichage devra donc être arrondi à deux chiffres après la virgule. Vous pouvez utiliser la méthode `Math.round()` pour réaliser cette opération.

Afin de clarifier l'affichage des informations dans l'alerte, on peut aller à la ligne en utilisant '`\n`'.

Solutions des exercices

1 https://repl.it/

p. 4 **Solution n°1**

```
// En français
let nom
let prenom
let age
let adresseEmail
let numeroDeTelephone

// En anglais
let surName
let firstName
let age
let emailAdress
let telephoneNumber
```

p. 8 **Solution n°2**

```
let postalAddress = '1, rue de la paix, 75000 Paris';
let birthCountry = 'France';
let children = 2;

console.log(postalAddress);
console.log(birthCountry);
console.log(children);
```

p. 8 **Solution n°3**

```
let postalAddress = '1, rue de la paix, 75000 Paris';
let children = 2;

children = 3;
postalAddress = '32, rue de Vaugirard, 75000 Paris';

console.log(postalAddress);
console.log(children);
```

p. 10 **Solution n°4**

Étape	variable1	variable2	temp
let value1 ='Caroline';	Caroline		
let value2 = 'Bonsoir',	Caroline	Bonsoir	
let temp = value1;	Caroline	Bonsoir	Caroline
value1 = value2 ;	Bonsoir	Bonsoir	Caroline
value2 = temp	Bonsoir	Caroline	Caroline

p. 13 Solution n°5

```
let cartTotalValue = parseFloat(productsTotalValue) - parseInt(specialOffer);
console.log(cartTotalValue);
```

p. 14 Solution n°6

```
const USERNAME = "John Doe";
let status;
```

p. 14 Solution n°7

Non. Ce code est incorrect car une valeur doit être assignée à une constante lors de la déclaration de celle-ci.

p. 14 Solution n°8

Oui. Les nouvelles déclarations sont autorisées car elles se situent chacune dans un scope différent.

Exercice p. 15 Solution n°9

Exercice

Une variable sert à stocker des informations.

- (●) Vrai
- ( ) Faux

Exercice

Une variable peut se définir par les mots-clés...

- [x] var
- [ ] this
- [x] const
- [x] let

Exercice

Quels sont les types que la console affichera ?

```
let myVariable = '10';
console.log(typeof myVariable);
myVariable = Number(10)
console.log(typeof myVariable);
myVariable = new Date()
console.log(typeof myVariable);
myVariable = new Array()
console.log(typeof myVariable);
```

- [x] string
- [ ] boolean
  Un booléen est un type primitif retournant les valeurs true ou false.
- [x] number
- [ ] array
  En JavaScript, `Array` est un sous-objet du type `Object`. La console ne reconnaît pas ce type directement.
- [x] object
- [ ] date
  En JavaScript, `Date()` est un sous-objet du type `Object`. La console ne reconnaît pas ce type directement.

Exercice

Qu'affichera la console ? Répondre

```
1 const var1 = 'le résultat est '
2 const var2 = 3
3 const var3 = 4
4 console.log(var1 + (var2 + var3)+ ' ou ' + var2 + var3)
```

le résultat est 7 ou 34

On utilise la concaténation de variables de types différents. `var1` étant un `string`, l'utilisation du signe + va ajouter du texte à la suite.

`var2` et `var3` sont de type `number` : elles sont donc additionnables. La parenthèse les englobant permet cette addition.

Exercice

L'utilisation du mot-clé `var` est recommandé en JavaScript.

- ○ Vrai
  Il est préférable d'utiliser les mots-clés `let` et `const`.
- ◉ Faux

Exercice

Ce code est correct.

```
1 let myVariable = 25;
2 myVariable = 'test';
3 myVariable = false;
```

- ◉ Vrai
- ○ Faux
  L'utilisation de `let` implique que la variable peut être modifiable. Le faible typage de JavaScript permet d'affecter des valeurs de types différents au type d'origine de la variable.

Exercice

À combien est égale la variable `counter` ?

```
let counter = 5;
for (let i = 0; i < 10; i++){
    let counter = i;
    // counter est égal à 9 lorsque la boucle se termine
}
console.log(counter )
```

5

La variable `counter` étant dans la boucle, elle n'appartient qu'au scope de celle-ci : elle n'a pas de lien avec la variable `counter` précédant celle-ci.

Exercice

Quelle nomenclature correspond à une bonne pratique ?

- ○ const mySupervariable ;
- ○ const mysupervariable ;
- ◉ const MYSUPERVARIABLE ;
- ○ const mySuperVariable ;

La bonne pratique veut que l'on écrive les constantes en majuscules.

Exercice

Dans le code suivant, `name` est une variable.

```
const personne = {
   name: 'Paul',
   age: 12
};
```

- ○ Vrai

  `name` est une propriété appartenant à un `Object`.

- ◉ Faux

Exercice

Ce code est juste.

```
const var1 = var2 = var3 = 4;
```

- ◉ Vrai
- ○ Faux

  On a le droit de définir plusieurs variable sur une seule ligne.

p. 17 Solution n°10

```
// On stocke les prix
const PRICEAPPLE = 0.32;
const PRICEPEAR = 0.44;

// On stocke les quantités
const QuantityAppleSold = prompt('Saisissez un nombre de pommes');
const QuantityPearSold = prompt('Saisissez un nombre de poires');

```

```
// Recette réalisée pour chaque fruit
const priceTotalApple = PRICEAPPLE * QuantityAppleSold;
const priceTotalPear= PRICEPEAR * QuantityPearSold ;

// Chiffre d'affaires
const turnover = priceTotalApple + priceTotalPear;

// On crée l'alerte qui affichera les informations à l'utilisateur
alert('pommes => quantité : ' + QuantityAppleSold + ', chiffre d\'affaire: ' +
Math.round(priceTotalApple * 100) / 100 + '€ \n' +
'poires => quantité : ' + QuantityPearSold + ', chiffre d\'affaire: ' +
Math.round(priceTotalPear * 100) / 100 + '€ \n' +
'chiffre d\'affaire => ' + Math.round(turnover * 100) / 100 + '€');
```