

Les variables en JS

Table des matières

I. Contexte	3
II. Introduction aux variables et exemples d'utilisation en JS	3
A. Introduction.....	3
B. Définition d'une variable	3
C. Utilité des variables.....	4
D. Les types de variables	4
E. Déclaration de variables en JavaScript	5
F. Comparaison entre 'var', 'let' et 'const'	7
G. Exercice : Quiz.....	8
III. Bonnes pratiques et exercices de compréhension	9
A. Bonnes pratiques pour déclarer et utiliser les variables en JavaScript.....	9
B. Exercice de compréhension et mise en pratique	9
1. Exercice 1 : déclaration et manipulation de variables	9
2. Exercice 2 : la portée des variables.....	10
3. Exercice 3 : application des bonnes pratiques.....	10
C. Exercice : Quiz	11
IV. Essentiel	11
V. Auto-évaluation	12
A. Exercice	12
B. Test.....	12
Solutions des exercices	13

I. Contexte

Durée (en minutes) : 60 minutes

Environnement de travail : PC, connexion internet stable, éditeur de code

Prérequis : connaître le HTML et CSS, avoir des notions en algorithme

Contexte

JavaScript est un langage de programmation permettant d'ajouter des fonctionnalités sur les pages web (comme des cartes interactives ou des animations 2D/3D). Il fait partie des langages standards du web comme le HTML et le CSS. Ce langage est utilisé par la majorité des sites web. Il permet de manipuler des objets (au sens informatique).

En effet, dans un navigateur, un script en JavaScript sera utilisé pour apporter de l'interactivité ou une touche dynamique à un site web.

Le langage JavaScript, comme la plupart des langages de programmation, est porté sur des variables. Celles-ci sont liées à une donnée par affectation. Il existe plusieurs types de variables. C'est ce sur quoi nous allons nous concentrer dans ce cours.

II. Introduction aux variables et exemples d'utilisation en JS

A. Introduction

Visual Studio Code (VScode) est un éditeur de code source léger, mais puissant, qui s'exécute sur votre bureau et qui est disponible pour Windows, macOS et Linux. Il intègre la prise en charge de JavaScript, TypeScript et Node.js. Il dispose d'un riche écosystème d'extensions pour d'autres langages et environnements d'exécution (tels que C++, C#, Java, Python, PHP, Go, .NET).

Complément

Pour le télécharger, suivez ce lien : [Visual Studio Code¹](https://code.visualstudio.com/Download)

Pour consulter la documentation : [Visual Studio Code²](https://code.visualstudio.com/docs)

B. Définition d'une variable

Définition

Qu'est-ce qu'une variable ?

Une variable est un élément fondamental en programmation qui permet de stocker, manipuler et référencer des données dans un programme. Dans le contexte de JavaScript, une variable est un conteneur pour une valeur qui peut être de différents types, tels que des nombres, des chaînes de caractères, des objets, des tableaux, des fonctions, etc.

Une variable est définie par un identifiant (son nom) et une valeur. L'identifiant est utilisé pour accéder à la valeur stockée dans la variable et pour modifier cette valeur si nécessaire. Les variables permettent aux développeurs de gérer facilement les données dans leur programme, de les organiser et de les manipuler en fonction des besoins spécifiques de l'application.

En JavaScript, les variables peuvent être déclarées en utilisant les mots-clés 'var', 'let' et 'const'. Chacun de ces mots-clés possède des propriétés et des comportements spécifiques qui influencent la portée, l'assignation et la déclaration des variables.

¹ <https://code.visualstudio.com/Download>

² <https://code.visualstudio.com/docs>

C. Utilité des variables

Les variables jouent un rôle essentiel dans la programmation, car elles permettent aux développeurs de stocker, organiser et manipuler des données de manière flexible et efficace. Parmi leurs principales utilisations, on trouve le stockage de données, où les variables conservent des informations pour une utilisation ultérieure dans le programme et peuvent contenir divers types de données.

En outre, les variables contribuent à la réutilisation du code, réduisant ainsi la redondance en stockant des valeurs ou des références fréquemment utilisées. Cela permet au développeur d'éviter d'écrire plusieurs fois le même code ou la même valeur. Les variables simplifient également les calculs en facilitant les opérations mathématiques et logiques complexes, en stockant les résultats intermédiaires ou en agissant comme des compteurs.

Au niveau de l'organisation du code, les variables aident à structurer et à regrouper des informations connexes, améliorant ainsi la lisibilité et la maintenabilité du code en permettant aux développeurs de donner des noms significatifs aux données. Les variables rendent les programmes plus adaptables, en permettant aux développeurs de modifier facilement les valeurs sans avoir à changer le code lui-même, par exemple, en ne modifiant qu'une seule fois une valeur utilisée à plusieurs endroits dans le programme.

D. Les types de variables

Définition Les nombres à virgule Flottante (Float)

Les nombres à virgule flottante, également connus sous le nom de nombres décimaux, sont utilisés en JavaScript pour représenter des nombres avec des valeurs décimales. Les variables de ce type sont utilisées pour stocker des nombres réels, tels que des nombres décimaux ou des nombres avec une partie décimale. En JavaScript, les nombres à virgule flottante sont représentés par le type de données "number".

```
1 let pi = 3.14159; // Déclaration d'une variable avec un nombre à virgule flottante
2 console.log(pi); // Affiche : 3.14159
```

Définition Les nombres entiers (Integer)

Integer est utilisé en JavaScript pour représenter un nombre entier sans partie décimale. Ce type de variable est utilisé pour stocker des nombres entiers, tels que des nombres de comptages ou des identifiants. En JavaScript, les entiers sont également représentés par le type de données "number".

```
1 let age = 25; // Déclaration d'une variable avec un nombre entier
2 console.log(age); // Affiche : 25
```

Définition Les chaînes de caractères (String)

On retrouve les chaînes de caractères en JavaScript pour représenter du texte. Les variables de ce type sont utilisées pour stocker des séquences de caractères, comme des mots, des phrases ou des adresses électroniques. En JavaScript, les chaînes sont représentées par le type de données "string" et peuvent être déclarées avec des guillemets simples (') ou doubles (").

Exemple

```
1 let nom = 'Alice'; // Déclaration d'une variable avec une chaîne de caractères
2 console.log(nom); // Affiche : Alice
```

Définition Les Booléens (Boolean)

Les valeurs booléennes sont utilisées en JavaScript pour représenter des valeurs de vérité, c'est-à-dire des valeurs qui sont vraies ou fausses. Ce type de variable est utilisé pour stocker des états logiques, tels que des conditions ou des options d'activation/désactivation. En JavaScript, les valeurs booléennes sont représentées par le type de données "Boolean".

```
1 let estActif = true; // Déclaration d'une variable avec une valeur booléenne
2 console.log(estActif); // Affiche : true
```

Définition Les tableaux (Array)

En JavaScript, les tableaux sont utilisés pour stocker plusieurs valeurs dans une seule variable. Ce type de variable est utilisé pour représenter une collection de données, comme une liste d'éléments ou une série de valeurs. En JavaScript, les tableaux sont représentés par le type de données "array" et peuvent contenir des valeurs de différents types.

Exemple

```
1 let nombres = [1, 2, 3, 4, 5];
2 nombres.forEach(function(nombre) {
3   console.log(nombre);
4 });
```

E. Déclaration de variables en JavaScript

Utilisation de 'var'

L'utilisation de 'var' en JavaScript concerne la déclaration de variables qui peuvent être réaffectées et dont la portée est définie par la fonction englobante. Bien que 'var' ait été le mot-clé standard pour déclarer des variables dans les versions antérieures de JavaScript, l'introduction de 'let' et 'const' dans ECMAScript 2015 (ES6) a rendu son utilisation moins courante en raison de certaines limitations.

```
1 // Déclaration et initialisation d'une variable avec 'var'
2 var age = 25;
3
4 // Réaffectation d'une nouvelle valeur à la variable
5 age = 26;
6 console.log(age); // Affiche "26"
```

Dans cet exemple, nous déclarons une variable appelée age avec la valeur 25. Ensuite, nous réaffectons la valeur 26 à la variable et affichons la nouvelle valeur.

Cependant, il est important de noter que l'utilisation de 'var' présente certaines limitations. Par exemple, les variables déclarées avec 'var' sont accessibles en dehors du bloc dans lequel elles sont définies, tant qu'elles restent dans la fonction englobante. Cela peut entraîner des problèmes de lisibilité et des erreurs involontaires.

Voici d'ailleurs une erreur type de l'utilisation de "var" :

```
1 function testFunction() {
2   if (false) {
3     var testVariable = "Hello world";
4   }
5   console.log(testVariable);
6 }
7
8 testFunction(); // Résultat : "undefined"
```

Dans cet exemple, la variable testVariable est déclarée dans une condition qui ne sera jamais satisfaite, car la condition if (false) ne sera jamais vraie. Cependant, la variable est toujours accessible en dehors de cette condition, car elle est déclarée avec le mot-clé "var".

Lorsque la fonction testFunction() est appelée, elle affiche la valeur de la variable test avec console.log(). Comme la variable est déclarée avec "var" dans la fonction, elle est accessible tout au long de la fonction, même si elle n'a jamais été définie.

Cela peut conduire à des erreurs involontaires, car si la variable n'est pas définie dans toutes les conditions dans lesquelles elle est utilisée, elle peut causer des problèmes dans d'autres parties du code. C'est pourquoi les variables sont déclarées dans des blocs de code avec `"let"` ou `"const"` afin de limiter leur portée et d'éviter les erreurs involontaires.

```
1 if (true) {
2     var prenom = "Alice";
3 }
4 console.log(prenom); // Affiche "Alice" même si la variable est déclarée dans le bloc if
```

De plus, les variables déclarées avec `'var'` ont une particularité concernant leur portée : elles sont traitées comme si elles étaient déclarées au début de la fonction englobante, même si elles sont définies plus loin dans le code. Ce phénomène peut provoquer des comportements inattendus.

```
1 console.log(monNom); // Affiche "undefined" au lieu de générer une erreur
2 var monNom = "Alice";
```

Malgré ces limitations, `'var'` est toujours pris en charge pour des raisons de compatibilité avec les anciens codes. Cependant, pour les débutants, il est recommandé d'apprendre et d'utiliser `'let'` et `'const'` pour déclarer des variables, car ils offrent une meilleure gestion de la portée et préviennent les erreurs liées au hoisting.

Définition

Le hoisting est un comportement en JavaScript où les déclarations de variables et de fonctions sont déplacées au début de leur portée respective, c'est-à-dire que la variable ou la fonction peut être utilisée avant d'être déclarée. Cela signifie que même si une variable est déclarée plus tard dans le code, elle peut être utilisée avant sa déclaration en raison du hoisting.

Utilisation de 'let'

L'utilisation de `'let'` en JavaScript concerne la déclaration de variables qui peuvent être réaffectées et dont la portée est définie par le bloc englobant. `'let'` a été introduit dans ECMAScript 2015 (ES6) et est devenu un choix préféré pour déclarer des variables, en particulier pour les débutants, en raison de sa gestion améliorée de la portée et de l'absence de hoisting.

Exemple

```
1 // Déclaration et initialisation d'une variable avec 'let'
2 let age = 25;
3
4 // Réaffectation d'une nouvelle valeur à la variable
5 age = 26;
6 console.log(age); // Affiche "26"
```

Dans cet exemple, nous déclarons une variable appelée `age` avec la valeur 25. Ensuite, nous réaffectons la valeur 26 à la variable et affichons la nouvelle valeur.

Contrairement à `'var'`, les variables déclarées avec `'let'` ont une portée de bloc, ce qui signifie qu'elles ne sont accessibles qu'à l'intérieur du bloc dans lequel elles sont définies.

```
1 if (true) {
2     let prenom = "Alice";
3 }
4 console.log(prenom); // Erreur : prenom n'est pas défini à l'extérieur du bloc if

1 console.log(monNom); // Erreur : monNom n'est pas défini
2 let monNom = "Alice";
```

Utilisation de 'const'

L'utilisation de 'const' en JavaScript concerne la déclaration de variables dont la valeur ne peut pas être modifiée une fois qu'elles ont été initialisées. 'const' a été introduit dans ECMAScript 2015 (ES6) et est devenu un choix préféré pour déclarer des variables immuables, en raison de sa gestion améliorée de la portée et de l'absence de hoisting.

Exemple Exemple simple de l'utilisation de 'const'

```
1 // Déclaration et initialisation d'une variable avec 'const'
2 const age = 25;
3
4 // Tentative de réaffectation d'une nouvelle valeur à la variable
5 age = 26; // Erreur : tentative de réaffecter une valeur à une constante
```

Dans cet exemple, nous déclarons une variable appelée age avec la valeur 25 en utilisant 'const'. Si nous essayons de réaffecter la valeur 26 à la variable, cela générera une erreur, car la valeur d'une constante ne peut pas être modifiée une fois qu'elle a été initialisée.

Comme pour 'let', les variables déclarées avec 'const' ont une portée de bloc, ce qui signifie qu'elles ne sont accessibles qu'à l'intérieur du bloc dans lequel elles sont définies.

```
1 if (true) {
2   const prenom = "Alice";
3 }
4 console.log(prenom); // Erreur : prenom n'est pas défini à l'extérieur du bloc if
```

De plus, les variables déclarées avec 'const' ne sont pas hoistées, ce qui empêche les comportements inattendus et améliore la lisibilité du code.

```
1 console.log(monNom); // Erreur : monNom n'est pas défini
2 const monNom = "Alice";
```

Il est recommandé d'utiliser 'const' pour déclarer des variables immuables et de se familiariser avec 'let' pour les variables dont la valeur doit être modifiée.

F. Comparaison entre 'var', 'let' et 'const'

Caractéristique	var	let	const
Portée	Portée de fonction	Portée de bloc	Portée de bloc
Réaffectation	Oui	Oui	Non
Hoisting	Oui (déclaration)	Oui (déclaration)	Non
Initialisation	Facultative	Facultative	Obligatoire

En ce qui concerne la portée, il convient de noter que 'var' a une portée de fonction, c'est-à-dire qu'elle peut être appelée n'importe où dans la fonction où elle est déclarée. En revanche, 'let' et 'const' ont une portée de bloc, ce qui signifie qu'elles sont uniquement accessibles dans le bloc de code où elles sont déclarées, généralement un couple de {}.

En ce qui concerne la réaffectation, il est important de noter que 'var' et 'let' peuvent être réaffectées à tout moment, ce qui signifie que leur valeur peut être modifiée après leur déclaration. En revanche, une fois que 'const' a été déclarée et initialisée, elle ne peut pas être réaffectée.

En ce qui concerne le hoisting, cela signifie que la déclaration de 'var' et 'let' sont élevées en haut de leur scope, c'est-à-dire qu'elles peuvent être appelées avant leur déclaration effective. Toutefois, seule la déclaration de 'var' est "hoisted", pas son affectation.

Si l'on essaie d'appeler une variable déclarée à l'aide de 'let' avant son affectation, cela nous renverra un undefined.

En revanche, 'const' n'est pas soumise au hoisting.

Les variables déclarées avec "let" et "const" sont hoisting sans initialisation par défaut.

Les variables déclarées avec "var" sont hoisting avec comme initialisation par défaut "undefined".

Le "scope" se réfère à la portée ou à l'étendue des variables déclarées en JavaScript. En d'autres termes, le scope détermine où dans le code une variable peut être utilisée ou référencée.

G. Exercice : Quiz

[solution n°1 p.15]

Question 1

Une variable en JavaScript permet de stocker, manipuler et référencer des données dans un programme.

- ☐ Vrai
- ☐ Faux

Question 2

Les variables ne peuvent contenir que des nombres.

- ☐ Vrai
- ☐ Faux

Question 3

Les variables déclarées avec 'const' doivent être utilisées pour les valeurs constantes, tandis que 'let' devrait être utilisé pour les variables dont la valeur peut changer dans leur bloc tout au long de l'exécution du programme.

- ☐ Vrai
- ☐ Faux

Question 4

Les variables aident à structurer et à regrouper des informations connexes, améliorant ainsi la lisibilité et la maintenabilité du code.

- ☐ Vrai
- ☐ Faux

Question 5

Les variables déclarées avec 'var' ne peuvent pas être réaffectées.

- ☐ Vrai
- ☐ Faux

III. Bonnes pratiques et exercices de compréhension

A. Bonnes pratiques pour déclarer et utiliser les variables en JavaScript

JavaScript est l'un des langages de programmation les plus populaires utilisés pour le développement web. Il permet de créer des applications interactives et dynamiques sur le Web, des jeux, des applications mobiles et bien plus encore. Dans ce contexte, les variables en JavaScript sont des conteneurs qui stockent des valeurs pouvant être utilisées et modifiées tout au long de l'exécution d'un programme. Les variables sont déclarées à l'aide de mots-clés tels que 'var', 'let' ou 'const'. Ainsi, pour optimiser l'utilisation des variables en JavaScript, il est recommandé de suivre certaines bonnes pratiques.

La première bonne pratique consiste à nommer les variables de manière significative. En effet, le nom d'une variable doit être descriptif et significatif pour permettre aux développeurs de comprendre facilement ce que la variable représente. Évitez les noms génériques tels que 'a', 'b' ou 'x' qui peuvent entraîner des erreurs et rendre le code difficile à comprendre. Ensuite, il est essentiel d'utiliser la portée de variable appropriée. En JavaScript, la portée d'une variable définit où elle peut être utilisée dans un programme. Les variables 'var' ont une portée de fonction, tandis que les variables 'let' et 'const' ont une portée de bloc. Il est donc important de bien choisir la portée de la variable pour éviter les collisions de noms et les erreurs.

Il est également recommandé d'éviter les variables globales autant que possible. En effet, les variables globales sont accessibles depuis n'importe où dans le programme, ce qui peut rendre le code difficile à comprendre et à maintenir. Il est préférable d'utiliser des variables locales pour limiter leur portée. Toutes les variables doivent être déclarées avant leur utilisation dans un programme. L'utilisation de variables non déclarées peut entraîner des erreurs et des comportements inattendus. Il est donc important de déclarer toutes les variables. Il est conseillé d'utiliser 'const' pour les valeurs constantes telles que les nombres pi ou les adresses URL qui ne changent pas tout au long du programme. Les variables déclarées avec 'const' sont des constantes et ne peuvent pas être réaffectées après leur déclaration. Pour les variables qui changent de valeur tout au long du programme, il est recommandé d'utiliser 'let'.

Il est important d'éviter les variables réaffectées autant que possible, car cela peut rendre le code difficile à comprendre et à déboguer. Il est donc préférable d'utiliser des variables immuables pour faciliter la lecture et la maintenance du code.

Les variables peuvent être initialisées lors de leur déclaration en leur attribuant une valeur. Cela peut améliorer la lisibilité du code et faciliter sa maintenance. Enfin, évitez de déclarer des variables inutiles et utilisez plutôt des valeurs littérales directement dans le code, pour éviter de rendre le code difficile à comprendre et à déboguer.

B. Exercice de compréhension et mise en pratique

1. Exercice 1 : déclaration et manipulation de variables

Méthode

Créez un programme qui calcule l'aire d'un triangle à partir de la longueur de sa base et de sa hauteur. Déclarez et initialisez les variables pour la base et la hauteur, puis calculez l'aire du triangle en utilisant la formule "aire = (base * hauteur) / 2". Affichez ensuite l'aire du triangle à l'aide de la méthode console.log().

Solution :

```
1 // Déclaration et initialisation des variables pour la base et la hauteur
2 let base = 7;
3 let hauteur = 4;
4
5 // Calcul de l'aire du triangle
6 const aire = (base * hauteur) / 2;
7
8 // Affichage de l'aire du triangle
9 console.log("L'aire du triangle est de " + aire + " unités carrées.");
10 // Le programme affichera "L'aire du triangle est de 14 unités carrées."
```

Dans ce code JavaScript, nous effectuons le calcul de l'aire d'un triangle. Tout d'abord, nous déclarons et initialisons les variables de la base et de la hauteur en utilisant la syntaxe `let`. Ensuite, nous utilisons ces variables pour calculer l'aire du triangle en multipliant la base par la hauteur et en divisant le résultat par 2. Finalement, nous affichons le résultat en utilisant la méthode `console.log()` avec un message qui inclut l'aire calculée.

2. Exercice 2 : la portée des variables

Quelle est la sortie de ce code ?

```
1 let x = 5;
2
3 function foo() {
4   let x = 10;
5   console.log(x);
6 }
7
8 foo();
9 console.log(x);
```

Solution :

La sortie de ce code sera :

10

5

Au début du code, la variable `x` est déclarée et initialisée à 5. Ensuite, une fonction nommée `foo()` est définie. À l'intérieur de cette fonction, une variable locale `x` est déclarée et initialisée à 10. Lorsque la fonction `foo()` est appelée, la valeur de la variable locale `x` (qui est 10) est affichée. Enfin, en dehors de la fonction `foo()`, la valeur de la variable globale `x` est affichée (qui est 5). Cela montre que la portée de la variable locale `x` à l'intérieur de la fonction `foo()` est limitée à cette fonction, et que la portée de la variable globale `x` s'étend à l'ensemble du code.

3. Exercice 3 : application des bonnes pratiques

Considérez le code suivant :

```
1 let x = 5;
2 function calculate(num1, num2) {
3   let result = num1 + num2 + x;
4   return result;
5 }
6 console.log(calculate(2, 3));
```

Identifiez les bonnes pratiques qui ont été suivies dans ce code et proposez une amélioration pour respecter une autre bonne pratique.

Solution :

Dans ce code, les bonnes pratiques suivantes ont été appliquées :

- Suppression de la variable `x` qui n'est pas utilisé et définir un troisième paramètre dans la fonction :

```
1 function calculate(num1, num2, num3) {
2
3   let result = num1 + num2 + num3
4   return result
5 }
```

- L'utilisation de fonctions pour regrouper les blocs de code réutilisables,
- L'utilisation de `console.log()` pour afficher les résultats.

La suppression de la variable `result` qui n'est utilisée nulle part, il est possible de retourner directement le résultat de l'opération.

C. Exercice : Quiz

[solution n°2 p.15]

Question 1

La première bonne pratique consiste à nommer les variables de manière significative et surtout d'éviter les noms génériques tels que 'a', 'b' ou 'x' qui peuvent entraîner des erreurs et rendre le code difficile à comprendre.

- ☐ Vrai
- ☐ Faux

Question 2

Il est recommandé d'utiliser des variables globales autant que possible pour faciliter l'accès à ces variables à partir de n'importe où dans le programme.

- ☐ Vrai
- ☐ Faux

Question 3

Les variables déclarées avec 'const' doivent être utilisées pour les valeurs constantes, et les variables déclarées avec 'let' pour les variables qui changent de valeur tout au long du programme.

- ☐ Vrai
- ☐ Faux

Question 4

Il est important d'éviter les variables réaffectées autant que possible, car cela peut rendre le code difficile à comprendre et à déboguer.

- ☐ Vrai
- ☐ Faux

Question 5

En JavaScript, toutes les variables ont une portée de fonction.

- ☐ Vrai
- ☐ Faux

IV. Essentiel

Les variables sont des éléments essentiels de la programmation qui permettent de stocker des données et de les réutiliser dans le code. En JavaScript, il existe trois façons de déclarer des variables : `var`, `let` et `const`.

La déclaration de variables avec `var` a été utilisée dans les anciennes versions de JavaScript, mais son utilisation est maintenant déconseillée. La raison en est que `var` a une portée globale et peut être redéclaré plusieurs fois dans le même bloc de code, ce qui peut conduire à des erreurs difficiles à déboguer.

Les variables déclarées avec `let` et `const` ont une portée de bloc, ce qui signifie qu'elles ne sont accessibles que dans le bloc dans lequel elles ont été déclarées. La différence entre `let` et `const` est que les variables déclarées avec `let` peuvent être réaffectées ultérieurement, tandis que les variables déclarées avec `const` ne peuvent pas être réaffectées.

Pour utiliser les variables de manière efficace, il est important de suivre certaines bonnes pratiques. La première est de toujours déclarer les variables avec `let` ou `const`, et d'éviter d'utiliser `var`. La deuxième est de déclarer les variables au début de leur portée, pour éviter les problèmes de portée. Enfin, il est important d'utiliser des noms de variables significatifs et de commenter le code pour faciliter la compréhension et la maintenance du code.

V. Auto-évaluation

A. Exercice

Vous êtes développeur web dans une agence de communication digitale et travaillez sur un projet de site e-commerce pour un client.

Question 1

[solution n°3 p.16]

Vous devez rédiger en JavaScript, un morceau de code capable d'afficher les données suivantes dans la console :

1. Une variable pour stocker le nom d'un produit vendu,
2. Une variable pour stocker le nombre d'unités vendues d'un produit,
3. Une variable pour stocker le prix unitaire d'un produit,
4. Une variable pour stocker l'identifiant unique d'un client qui a effectué une transaction.

Question 2

[solution n°4 p.17]

Ce n'est pas vous, mais un collègue également développeur qui avait commencé à rédiger les variables JavaScript représentant un acheteur et celui-ci est en vacances et injoignable. Toutefois, il semblerait que le code contienne des erreurs. Votre chef d'équipe vous demande donc de corriger celles-ci.

Voici le code en question :

```
1 // Variables pour les informations de l'acheteur
2 let Nom = "Alice"; // Nom de l'acheteur (chaîne de caractères)
3 let prenom = "Dupont"; // Prénom de l'acheteur (chaîne de caractères)
4 let adresseLivraison = 123 Rue des Fleurs; // Adresse de livraison de l'acheteur (chaîne de caractères)
5 let adresseFacturation = "456 Avenue des Étoiles"; // Adresse de facturation de l'acheteur (chaîne de caractères)
6 let e-mail = "alice.dupont@example.com; // Adresse e-mail de l'acheteur (chaîne de caractères)
7 let téléphone = 0123456789; // Numéro de téléphone de l'acheteur (chaîne de caractères)
```

B. Test

Exercice 1 : Quiz

[solution n°5 p.17]

Question 1

L'utilisation de `'let'` en JavaScript concerne la déclaration de variables qui ne peuvent pas être réaffectées.

- ☐ Vrai
- ☐ Faux

Question 2

Les variables déclarées avec `'var'` ont une portée limitée à la fonction dans laquelle elles ont été déclarées.

- ☐ Vrai
- ☐ Faux

Question 3

Les variables déclarées avec 'let' ont une portée définie par le bloc dans lequel elles sont déclarées.

- ☐ Vrai
- ☐ Faux

Question 4

Les variables sont définies par un identifiant (leur nom) et une valeur.

- ☐ Vrai
- ☐ Faux

Question 5

Il est possible d'initialiser une variable sans lui affecter une valeur en JavaScript.

- ☐ Vrai
- ☐ Faux


Solutions des exercices

Exercice p. 8 Solution n°1**Question 1**

Une variable en JavaScript permet de stocker, manipuler et référencer des données dans un programme.

☒ Vrai

☐ Faux


 Les variables en JavaScript sont utilisées pour stocker et manipuler des données.

Question 2

Les variables ne peuvent contenir que des nombres.

☐ Vrai

☒ Faux


 Les variables en JavaScript peuvent contenir différents types de données, tels que des chaînes de caractères, des objets, des tableaux, des fonctions, etc.

Question 3

Les variables déclarées avec 'const' doivent être utilisées pour les valeurs constantes, tandis que 'let' devrait être utilisé pour les variables dont la valeur peut changer dans leur bloc tout au long de l'exécution du programme.

☒ Vrai

☐ Faux


 Les variables déclarées avec 'const' doivent être utilisées pour les valeurs constantes, et les variables déclarées avec 'let' pour les variables qui changent de valeur tout au long du programme.

Question 4

Les variables aident à structurer et à regrouper des informations connexes, améliorant ainsi la lisibilité et la maintenabilité du code.

☒ Vrai

☐ Faux


 Les variables peuvent être utilisées pour organiser les données et améliorer la lisibilité du code.

Question 5

Les variables déclarées avec 'var' ne peuvent pas être réaffectées.

☐ Vrai

☒ Faux

 Les variables déclarées avec 'var' peuvent être réaffectées.


Exercice p. 11 Solution n°2

Question 1

La première bonne pratique consiste à nommer les variables de manière significative et surtout d'éviter les noms génériques tels que 'a', 'b' ou 'x' qui peuvent entraîner des erreurs et rendre le code difficile à comprendre.

☒ Vrai

☐ Faux


 Il est important de nommer les variables de manière significative pour permettre aux autres développeurs de comprendre facilement ce que la variable représente.

Question 2

Il est recommandé d'utiliser des variables globales autant que possible pour faciliter l'accès à ces variables à partir de n'importe où dans le programme.

☐ Vrai

☒ Faux


 Il est préférable d'utiliser des variables locales pour limiter leur portée et éviter les collisions de noms.

Question 3

Les variables déclarées avec 'const' doivent être utilisées pour les valeurs constantes, et les variables déclarées avec 'let' pour les variables qui changent de valeur tout au long du programme.

☒ Vrai

☐ Faux


 Les variables déclarées avec 'const' doivent être utilisées pour les valeurs constantes, et les variables déclarées avec 'let' pour les variables qui changent de valeur tout au long du programme.

Question 4

Il est important d'éviter les variables réaffectées autant que possible, car cela peut rendre le code difficile à comprendre et à déboguer.

☒ Vrai

☐ Faux


 Il est important d'utiliser des variables immuables autant que possible pour faciliter la lecture et la maintenance du code.

Question 5

En JavaScript, toutes les variables ont une portée de fonction.

☐ Vrai

☒ Faux

 En JavaScript, les variables 'var' ont une portée de fonction, tandis que les variables 'let' et 'const' ont une portée de bloc.

Voici une réponse qui aurait pu convenir à ce cas de figure :

```
1 // Variables pour stocker les données de vente
2 let nom_produit = "iPhone X";
3 let quantite_vendue = 10;
4 let prix_unitaire = 999.99;
5 let client_id = 12345;
6 // Affichage des variables pour vérification
7 console.log("Nom du produit : ", nom_produit);
8 console.log("Quantité vendue : ", quantite_vendue);
9 console.log("Prix unitaire : ", prix_unitaire);
10 console.log("ID du client : ", client_id);
```

p. 12 Solution n°4


Voici la correction :

```
1 // Variables pour les informations de l'acheteur
2 let nom = "Alice"; // Nom de l'acheteur (chaîne de caractères)
3 let prenom = "Dupont"; // Prénom de l'acheteur (chaîne de caractères)
4 let adresseLivraison = "123 Rue des Fleurs"; // Adresse de livraison de l'acheteur (chaîne de caractères)
5 let adresseFacturation = "456 Avenue des Étoiles"; // Adresse de facturation de l'acheteur (chaîne de caractères)
6 let email = "alice.dupont@example.com"; // Adresse e-mail de l'acheteur (chaîne de caractères)
7 let telephone = "0123456789"; // Numéro de téléphone de l'acheteur (chaîne de caractères)
```

Exercice p. 12 Solution n°5


Question 1

L'utilisation de 'let' en JavaScript concerne la déclaration de variables qui ne peuvent pas être réaffectées.

- ☐ Vrai
- ☒ Faux
-  Les variables déclarées avec 'let' peuvent être réaffectées.

Question 2

Les variables déclarées avec 'var' ont une portée limitée à la fonction dans laquelle elles ont été déclarées.

- ☒ Vrai
- ☐ Faux
-  Les variables déclarées avec 'var' ont une portée limitée à la fonction englobante dans laquelle elles ont été déclarées.

Question 3

Les variables déclarées avec 'let' ont une portée définie par le bloc dans lequel elles sont déclarées.

- ☒ Vrai
- ☐ Faux

Q Les variables déclarées avec 'let' ont une portée de bloc.

Question 4

Les variables sont définies par un identifiant (leur nom) et une valeur.

- ☒ Vrai
- ☐ Faux

Q Les variables ont un nom et une valeur.

Question 5

Il est possible d'initialiser une variable sans lui affecter une valeur en JavaScript.

- ☒ Vrai
- ☐ Faux

Q Il est possible de déclarer une variable sans lui affecter une valeur initiale en JavaScript. Dans ce cas, la valeur de la variable sera 'undefined' jusqu'à ce qu'une valeur lui soit affectée. Il est également possible d'initialiser une variable avec une valeur 'null', qui indique que la variable ne possède pas de valeur valide.