

# Les opérateurs

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. Les opérateurs arithmétiques et d'affectation</b>	<b>3</b>
A. Introduction.....	3
B. Les opérateurs arithmétiques.....	3
C. Les opérateurs d'affectation .....	5
D. Opérateurs à utiliser pour les chaînes de caractères.....	6
E. Exercice .....	6
<b>III. Opérateurs de comparaison et opérateurs logiques</b>	<b>7</b>
A. Les opérateurs de comparaison.....	7
B. Les opérateurs logiques .....	8
C. Exercice .....	10
<b>IV. Essentiel</b>	<b>11</b>
A. PROJET DUNE - JEU D'ENQUÊTE ESTIVAL .....	12
<b>V. Auto-évaluation</b>	<b>13</b>
A. Exercice .....	13
B. Test.....	14
<b>Solutions des exercices</b>	<b>15</b>

## I. Contexte

**Durée : 1 h 30**

**Prérequis : avoir étudié le cours jusqu'ici**

### Contexte

Vous connaissez la théorie sur JavaScript, un des langages de programmation côté client le plus populaire au monde. Vous savez aussi déclarer et définir des variables. Mais un point fondamental qui va maintenant nous intéresser est celui des opérateurs. Les opérateurs sont des symboles qui vont nous permettre d'analyser et de manipuler des données de différents types.

Il existe plusieurs types d'opérateurs parmi lesquels les opérateurs arithmétiques, les opérateurs d'affectation, les opérateurs de comparaison, et les opérateurs de logique. Ces différents opérateurs ont des rôles spécifiques, qui vous permettront de modifier et d'analyser les données.

L'objectif est de découvrir les différents opérateurs fondamentaux en JavaScript. Dans un premier temps, nous parlerons des opérateurs arithmétiques et d'affectation. Dans la seconde partie, nous nous intéresserons aux opérateurs de comparaison et de logique. Plusieurs exercices vous permettront d'apprendre à utiliser les opérateurs en JavaScript. Un dernier conseil, n'hésitez pas à consulter, dès que nécessaire, la documentation disponible sur le site [mdn<sup>1</sup>](https://developer.mozilla.org/fr/) (mozilla web docs).

## II. Les opérateurs arithmétiques et d'affectation

### A. Introduction

Nous nous intéresserons à des exemples numériques, puis nous verrons quels opérateurs peuvent nous être utiles lorsqu'on manipule des chaînes de caractères.

### B. Les opérateurs arithmétiques

#### Définition

Les opérateurs arithmétiques vont nous permettre de réaliser des opérations mathématiques simples. Nous utiliserons la commande `console.log` afin d'afficher dans la console les résultats des opérations mathématiques réalisées. Voici des exemples :

#### L'opérateur « + »

L'opérateur `+` permet simplement d'additionner 2 valeurs. Par exemple :

```
1 let a = 2;
2 let b = 567.42;
3 console.log(a + b);
```

Dans ce cas, `console.log(a + b)` affichera dans la console : 569,42.

#### L'opérateur « - »

L'opérateur `-` permet simplement de soustraire une valeur à une autre.

```
1 let a = 567.42;
2 console.log(a - 25);
```

---

<sup>1</sup> <https://developer.mozilla.org/fr/>

Dans ce cas, `console.log(a - 25)` affichera dans la console : 542,42.

### L'opérateur « \* »

L'opérateur `*` permet d'obtenir le produit de 2 nombres (multiplication).

```
1 let a = 2 * 4.5;
2 console.log(a);
```

Dans ce cas, `console.log(a)` affichera dans la console : 9.

### L'opérateur « / »

L'opérateur `/` permet de diviser un nombre par un autre nombre :

```
1 console.log(58 / 2);
```

Dans ce cas, `console.log(58/2)` affichera dans la console : 29.

### L'opérateur « % »

L'opérateur `%` (qu'on appelle en général « *modulo* ») permet d'obtenir le reste d'une division euclidienne :

```
1 let a = 90;
2 let b = 11;
3 console.log(a % b);
```

Dans ce cas, `console.log(a % b)` affichera dans la console : 2.

Pour bien comprendre, si on fait une division euclidienne avec 90 comme dividende et 11 comme diviseur, on obtient :

$$90 = 11 \times 8 + 2$$

On a donc un quotient égal à 8 et un reste égal à 2.

### L'opérateur « \*\* »

L'opérateur `**` permet d'élever un nombre à une puissance. Par exemple, si on veut obtenir l'écriture décimale de  $18^4$ , on fera :

```
1 console.log(18 ** 4);
```

Dans ce cas, `console.log(18 ** 4)` affichera dans la console : 104 976.

$$18^4 = 18 \times 18 \times 18 \times 18 = 104\,976$$

### Les opérateurs « ++ » et « -- »

L'opérateur `++` qu'on appelle « *opérateur d'incrément* » permet d'ajouter l'entier 1 à une valeur (on dit qu'il incrémente une variable).

```
1 let a = 2;
2 a++;
3 console.log(a);
```

Dans ce cas, `console.log(a)` affichera dans la console : 3.

En effet, l'incrément de `a` revient à écrire :

```
1 a = a + 1;
```

L'opérateur `--` est quant à lui appelé « *opérateur de décrémentation* ». Il fonctionne exactement comme l'opérateur d'incrément, mais permet en revanche de retirer 1 à une valeur :

```
1 let a = 3;
2 a --;
3 console.log(a);
```

Dans ce cas, `console.log(a)` affichera dans la console : 2.

## C. Les opérateurs d'affectation

### Définition

Maintenant que nous avons parlé de plusieurs opérateurs arithmétiques en JavaScript, parlons des opérateurs d'affectation. Ils permettent d'affecter (ou d'assigner) une valeur à une autre.

#### L'opérateur « = »

L'opérateur `=` est dit « *opérateur d'assignement simple* ». Il est utilisé pour définir la valeur d'une variable. Voici un exemple simple :

```
1 let a = 2;
2 console.log(a);
```

Ici, on assigne l'entier 2 à la variable `a`, avec l'opérateur d'assignement `=`. La ligne `console.log(a)` affichera donc dans la console : 2.

#### Les opérateurs « += », « -= », « \*= » et « /= »

L'opérateur `+=` peut être appelé « *opérateur d'affectation après addition* ». Il va permettre d'affecter à une variable sa propre valeur à laquelle on ajoute celle d'une seconde valeur :

```
1 let a = 2;
2 a += 8;
3 console.log(a);
```

Ici, `console.log(a)` affiche 10 dans la console. C'est comme si on avait fait :

```
1 a = a + 8;
```

L'opérateur d'affectation après soustraction `-=` fonctionne exactement de la même manière, mais renvoie la différence du premier nombre par le second :

```
1 let a = 18;
2 a -= 4;
3 console.log(a);
```

Ici, `a` sera égal à lui-même moins 4, donc à  $18 - 4$ , c'est-à-dire 14.

Les opérateurs d'affectation après multiplication (`*=`) et après division (`/=`) fonctionnent de la même manière, mais permettent respectivement d'affecter à une variable la valeur de la multiplication d'elle-même par un autre nombre et la valeur de la division d'elle-même par un autre nombre.

Les autres opérateurs d'affectation sont listés à l'adresse : [developer.mozilla.org](https://developer.mozilla.org)<sup>1</sup>

On comprend que les opérateurs vont nous permettre de modifier les valeurs des nombres. Mais ils vont aussi nous permettre de modifier d'autres types de données, comme les chaînes de caractères.

---

1 [https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Expressions\\_and\\_Operators#op%C3%A9rateurs\\_d'affectation](https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Expressions_and_Operators#op%C3%A9rateurs_d'affectation)

## D. Opérateurs à utiliser pour les chaînes de caractères

### Définition

Les chaînes de caractères permettent de représenter des textes. Ce sont, comme leur nom l'indique, des ensembles de caractères, assemblés les uns à la suite des autres. 2 opérateurs vont être très utiles pour modifier des chaînes de caractères : l'opérateur de concaténation et l'opérateur d'affectation après concaténation.

### L'opérateur de concaténation « + »

Le symbole `+` n'est pas utilisé uniquement pour additionner des nombres. Il peut servir à la fois d'opérateur d'addition (pour les nombres), mais aussi d'opérateur de concaténation (pour les chaînes de caractères). La concaténation, c'est le fait d'assembler plusieurs chaînes de caractères en une seule. Prenons un exemple simple :

```
1 let nom = "Parker";
2 let prenom = "Peter";
3 let nomComplet = prenom + " " + nom;
4 console.log(nomComplet);
```

Ici, nous avons concaténé 3 chaînes de caractères dans la variable **nomComplet** : la chaîne "Peter" contenue dans la variable **prenom**, la chaîne " " (qui contient un espace) et enfin la chaîne "Parker" contenue dans la variable **nom**. La variable **nomComplet** contient donc la chaîne "Peter Parker".

### L'opérateur d'affectation après concaténation « += »

Le symbole `+=` peut aussi être utilisé pour concaténer des chaînes. Quand il est utilisé avec des chaînes, il ne sert pas d'opérateur d'affectation après addition mais on peut dire « *d'opérateur d'affectation après concaténation* ». Prenons un exemple tout simple :

```
1 let nom = "Peter";
2 nom += " Parker";
3 console.log(nom);
```

La chaîne de caractères qui est placée après l'opérateur est ajoutée à la fin de la chaîne. Donc la variable **nom** contiendra la chaîne "Peter Parker".

Vous connaissez maintenant de nombreux opérateurs arithmétiques et d'affectation.

## E. Exercice

### Question 1

[solution n°1 p.17]

Insérer le bon opérateur pour obtenir la différence de **a** par **b**.

```
1 let a = 1376;
2 let b = 630;
3 console.log(a /*opérateur*/ b);
```

Sortie attendue : 746

### Question 2

[solution n°2 p.17]

Insérer le bon opérateur pour que la console incrémente de 1 la variable **nombre**.

```
1 let nombre = 3;
2 nombre/*opérateur*/;
3 console.log(nombre);
```

Sortie attendue : 4

**Question 3**

[solution n°3 p.17]

La console doit afficher « L'ordinateur HP a 16 GO de ram ».

```
1 let marque = "HP";
2 let ram = 16;
3 console.log(`*code*/marque/*code*/ram/*code*/`);
```

**Sortie attendue :** "L'ordinateur HP a 16 GO de ram"

**Question 4**

[solution n°4 p.17]

Voici un script qui permet d'afficher le produit d'un nombre par 3 et de le stocker dans la même variable. Insérer le bon opérateur pour qu'il fonctionne.

```
1 let nombre = 9;
2 nombre/*opérateur*/3;
3 console.log(nombre);
```

**Sortie attendue :** 27

**Question 5**

[solution n°5 p.17]

Voici un script qui permet d'afficher le périmètre d'un rectangle. Insérer les bons opérateurs pour qu'il soit opérationnel.

```
1 let largeur = 9;
2 let longueur = 10;
3 let perimetre = 2 /*opérateur*/ largeur /*opérateur*/ 2 /*opérateur*/ longueur;
4 console.log(perimetre);
```

**Sortie attendue :** 38

### III. Opérateurs de comparaison et opérateurs logiques

#### A. Les opérateurs de comparaison

**Définition**

Parlons maintenant des opérateurs de comparaison. Vous avez déjà entendu parler des fameux booléens, les variables dont la valeur est égale soit à **true**, soit à **false**. Les opérateurs de comparaison vont nous permettre de comparer des valeurs et de vérifier en quelques sortes des « hypothèses », en renvoyant un booléen (**true** si l'hypothèse est validée et **false** si elle ne l'est pas). Les opérateurs de comparaison seront donc indispensables lorsque nous utiliserons des systèmes de conditions. Voyons différents opérateurs de comparaison et comment ils fonctionnent :

**Les opérateurs « == » et « != »**

L'opérateur d'égalité **==** vérifie si 2 valeurs sont égales ou non. Si elles le sont, l'expression renverra **true**, sinon **false** :

```
1 let a = "Ordinateur";
2 let b = "Telephone";
3 let c = "Ordinateur";
4 console.log(a == b);
5 console.log (a == c);
```

On peut voir que **console.log(a == b)** affiche **false** car les 2 chaînes ne sont pas égales tandis que **console.log (a == c)** affiche **true** car les deux chaînes sont égales.

L'opérateur d'inégalité `!=` vérifie quant à lui si 2 valeurs ne sont pas égales :

```
1 let a = 17.2;
2 let b = 17.5;
3 let c = 17.2;
4 console.log(a != b);
5 console.log (a != c);
```

Dans ce cas, `console.log(a != b)` affiche dans la console **true** puisque **a** et **b** ne sont pas égaux. Cependant, `console.log (a != c)` affiche **false** car **a** et **c** sont égaux.

#### Complément Les opérateurs « === » et « !== »

Les opérateurs `===` (d'égalité stricte) et `!==` (d'inégalité stricte) vérifient respectivement si une valeur est égale ou inégale à une autre, mais prennent aussi en compte le type de la valeur (entier, booléen, chaîne de caractères, etc.). L'opérateur `===` vérifie si une valeur est égale à une autre et si leur type sont les mêmes. Dans ce cas, il renvoie **true**. L'opérateur `!==` renvoie **true** dans le cas où il y a au moins une différence de type ou de valeur entre les 2 valeurs.

```
1 let a = 1;
2 let b = "1";
3
4 console.log(a == b); //egalite : true
5 console.log(a === b); //egalite stricte : false
6
7 console.log(a != b); //inegalite : false
8 console.log (a !== b); ///inegalite stricte : true
```

#### Les opérateurs « < », « > », « <= » et « >= »

Pour bien comprendre, listons et expliquons chacun de ces opérateurs :

- `<` : stricte infériorité. Vérifie si une valeur est strictement inférieure à une autre,
- `>` : stricte supériorité. Vérifie si une valeur est strictement supérieure à une autre,
- `<=` : infériorité. Vérifie si une valeur est inférieure ou égale à une autre,
- `>=` : supériorité. Vérifie si une valeur est supérieure ou égale à une autre.

Prenons un exemple simple :

```
1 let a = 17.2;
2 let b = 17.5;
3 let c = 17.2;
4 console.log(a < b); //true
5 console.log (a <= c); //true
6 console.log (a > b); //false
7 console.log (a >= c); //true
```

Pour chaque ligne `console.log`, un commentaire spécifie dans le code le booléen affiché. On voit que c'est cohérent.

Abordons à présent les opérateurs logiques et comment ils fonctionnent avec des exemples concrets.

## B. Les opérateurs logiques

Nous pouvons citer 3 opérateurs logiques fondamentaux en JavaScript : `&&`, `||` et `!`. Dans cette partie, nous utiliserons uniquement des exemples où les expressions renvoient un booléen.



### L'opérateur « && »

L'opérateur **&&** signifie littéralement « et ». Dans le cas où on utilise un **&&** au milieu de deux expressions renvoyant un booléen, l'opérateur renverra **true** si les deux expressions renvoient **true**, sinon, il renverra **false**. Donc, quand on utilise l'opérateur **&&**, on vérifie si deux expressions sont « vraies ». Si une seule l'est, alors, l'opérateur renverra **false**. Voyons un exemple concret :

```
1 let a = 17.5;
2 let b = 17.5;
3 let c = 17.2;
4 console.log(a > c && a < b); //false
5 console.log(a == b && a > c); //true
```

L'instruction `console.log(a > c && a < b)` affiche dans la console **false** car les deux expressions ne sont pas vraies. Même si l'expression `a > c` est vérifiée, l'expression `a < b` est fausse.

En revanche, l'instruction `console.log(a == b && a > c)` affiche **true** car les expressions `a == b` et `a > c` sont toutes les deux vraies.

### L'opérateur « || »

L'opérateur **||** signifie littéralement « ou ». Dans le cas où on utilise un **||** au milieu de deux expressions renvoyant un booléen, l'opérateur renverra **true** si au moins une des deux expressions est juste, sinon, il renverra **false** :

```
1 let a = 17.5;
2 let b = 17.5;
3 let c = 17.2;
4 console.log(a == b || a < c); //true
5 console.log(a < c || a == c); //false
```

L'instruction `console.log(a == b || a > c)` affiche dans la console **true** car une des deux expressions renvoie **true** (l'expression `a == b` est vérifiée). Si les deux expressions étaient vraies, l'instruction aurait aussi affiché **true**.

En revanche, l'instruction `console.log(a < c || a == c)` affiche **false** car aucune des deux expressions n'est vraie.

### L'opérateur « ! »

L'opérateur **!** signifie littéralement « non ». On l'utilise pour renvoyer **true** si une expression renvoie **false** et inversement :

```
1 let a = 17.5;
2 let b = 17.5;
3 let c = 17.2;
4 let d = true;
5 let e = false;
6
7 console.log(!(a == b)); //false
8
9 console.log(!(a > b)); //true
10
11 console.log(! d); //false
12
13 console.log(! e); //true
```

Dans les deux premiers `console.log`, on place les expressions `a == b` et `a > b` entre parenthèses pour indiquer que si l'expression en entier renvoie **true**, alors, c'est **false** qui sera affiché et inversement. Les parenthèses permettent d'inverser le booléen renvoyé par toute l'expression et pas seulement la variable qui vient tout de suite après l'opérateur : **!**.

## Utilisation de plusieurs opérateurs dans un même exemple

Les opérateurs de comparaison et les opérateurs logiques peuvent être utilisés pour créer des structures plus complexes et vérifier si un ensemble d'expressions sont vérifiées. Mais le principe de base reste le même.

Les opérateurs en JavaScript n'ont plus beaucoup de secrets pour vous. La documentation de MDN fournit des listes assez complètes des opérateurs JS, en les agrémentant d'exemples. On les trouve sur la page citée plus haut : [mdn web docs](https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Expressions_and_Operators#op%C3%A9rateurs_d'affectation)<sup>1</sup>.

Cette page détaille tous les points que nous avons vus et traite d'opérateurs que nous n'avons pas abordés. N'hésitez donc pas à la consulter.

### C. Exercice

#### Question 1

[solution n°6 p.17]

Insérer le bon opérateur pour savoir si **a** est égal à **b**.

```
1 let a = "Hello World";
2 let b = "Hello World";
3 console.log(a /*opérateur*/ b);
```

**Sortie attendue :** true

#### Question 2

[solution n°7 p.17]

Vous réalisez un script permettant de savoir si un prix d'ordinateur est dans votre budget. Il doit être inférieur ou égal à 1 500 et strictement supérieur à 800. Complétez le code et lancez-le :

```
1 let prix = 1238;
2
3 console.log(/*code*/);
```

**Sortie attendue :** true

#### Question 3

[solution n°8 p.18]

Vous souhaitez réaliser un script vous permettant de déterminer si un vélo correspond ou non à vos critères d'achat. Le vélo doit être de la marque « *Rodkrider* » ou « *Btwin* » et doit avoir un prix strictement inférieur à 500.

```
1 let marque = "Btwin";
2
3 let prix = 800;
4
5 console.log(/*code*/);
```

**Sortie attendue :** false

#### Question 4

[solution n°9 p.18]

Vous souhaitez que la console affiche **true** si la marque n'est pas « *Btwin* », et uniquement dans ce cas. L'objectif est d'insérer le bon opérateur :

```
1 let marque = "Btwin";
2
3 console.log(marque /*opérateur*/ "Btwin");
```

**Sortie attendue :** false

<sup>1</sup> [https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Expressions\\_and\\_Operators#op%C3%A9rateurs\\_d'affectation](https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Expressions_and_Operators#op%C3%A9rateurs_d'affectation)

### Question 5

[solution n°10 p.18]

Vous souhaitez à nouveau que votre script affiche **true** si la marque n'est pas « *Btwin* », et uniquement dans ce cas. Cependant, la configuration n'est pas la même. Insérer le bon opérateur :

```
1 let marque = "Btwin";
2
3 console.log(/*opérateur*/(marque == "Btwin"));
```

**Sortie attendue :** false

## IV. Essentiel

Nous avons abordé des opérateurs en JavaScript. Les opérateurs vont permettre de manipuler et d'analyser les données. Nous avons vu à l'aide d'exemples 4 principaux types d'opérateurs : les opérateurs arithmétiques, les opérateurs d'affectation, les opérateurs de comparaison et les opérateurs logiques.

Les opérateurs d'arithmétique permettent de réaliser des opérations mathématiques. Les opérateurs d'affectation permettent d'affecter une valeur à une variable. Nous avons aussi vu que certains opérateurs peuvent permettre de manipuler des données comme des chaînes de caractères, et de concaténer du texte. Les opérateurs de comparaison permettent quant à eux de comparer des valeurs et de renvoyer un booléen en fonction du résultat de la comparaison. Les opérateurs de logique permettent notamment de combiner de manière logique des expressions. Les opérateurs de logique et de comparaison seront donc essentiels dans l'utilisation des conditions.

Les opérateurs sont donc indispensables lorsqu'on fait du développement web, et plus généralement de la programmation. Ce tableau récapitule les différents opérateurs que nous avons abordés :

Opérateur	#	Exemple	
Opérateurs arithmétiques			
Addition	+	let a = 10 + 14.5;	//a = 24.5
Soustraction	-	let a = 12 - 10.5;	//a = 1.5
Multiplication	*	let a = 12 * 10.5;	//a = 126
Division	/	let a = 12 / 4;	//a = 3
Modulo	%	let a = 13 % 4;	//a = 1
Puissance	**	let a = 10 ** 4;	//a = 10000
Incrémentation	++	let a = 14; a++;	//a = 15
Décrémentation	--	let a = 14; a--;	//a 13
Opérateurs d'affectation			
Assignement simple	=	let a = "texte";	//a = "texte"
Affectation après addition	+=	let a = 20; a += 10;	//a = 30
Affectation après soustraction	-=	let a = 20; a -= 10;	//a = 10
Affectation après multiplication	*=	let a = 20; a *= 10;	//a = 200
Affectation après division	/=	let a = 20; a /= 10;	//a = 2
Opérateurs utiles pour les chaînes de caractères			

Opérateur	#	Exemple	
Opérateurs arithmétiques			
Concaténation	+	let a = "Hello" + "World";	//a = "Hello World"
Affectation après concaténation	+=	let a = "Hello"; let a += " World";	//a = "Hello World"
Opérateurs de comparaison			
Égalité	==	let a = "bla"; console.log(a == "bla");	//true
Inégalité	!=	let a = "bla"; console.log(a != "bla");	//false
Égalité stricte	===	let a = "1"; console.log(a === 1);	//false
Inégalité stricte	!==	let a = "1"; console.log(a !== 1);	//true
Stricte infériorité	<	console.log(5 < 9);	//true
Stricte supériorité	>	console.log(5 > 9);	//false
Infériorité	<=	console.log(5 <= 5);	//true
Supériorité	>=	console.log(5 >= 4);	//true
Opérateurs de logique			
Et	&&	console.log(5 == 5 && 4 < 2);	//false
Ou		console.log(5 == 5    4 < 2);	//true
Non	!	console.log(! true);	//false

## A. PROJET DUNE - JEU D'ENQUÊTE ESTIVAL



Pour participer, rien de plus simple !

Débutez l'aventure en consultant la première partie du Live dédié à cette épopée :

👉 **PROJET DUNE - JEU D'ENQUÊTE ESTIVAL - FILIERES CODE et IT - PARTIE 1/3<sup>1</sup>**

### ÉNIGME NUMÉRO 11 - MAISON ATREÏDES

Développement Web - Analyse de Code HTML

Vous êtes un développeur web et vous devez corriger un bug dans le code HTML d'une page web.

Voici un extrait de code HTML :

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

**QUESTION :** *Quel est le titre de la page HTML ? La première lettre est le onzième élément à trouver pour le code pour les Atréïdes*

## V. Auto-évaluation

### A. Exercice

Vous cherchez à réaliser un système vérifiant qu'un ordinateur répond à différents critères. Cet ordinateur a plusieurs caractéristiques, voici le script de base :

```
1 let marque = "HP";
2 let modele = "Pavillon";
3 let stockage = 512;
4 let ram = 16;
5 let processeur = "intel core i7";
6
7 console.log(/*code Question 1*/);
8
9 /*code Question 2*/
```

La console doit afficher **true** si et seulement si l'ordinateur est de marque « HP » ou « Acer », qu'il a un stockage supérieur ou égal à 256 GO et une ram supérieure ou égale à 8 GO.

#### Question 1

[solution n°11 p.18]

Dans le `console.log` (à l'emplacement `/*code Question 1*/`) écrivez le code permettant de réaliser ce test.

---

1 <https://replay.studi.fr/2.3/24415176f7c4e9190ba008654ee85c906a583c22-1719329603985?p=data12/2.3>

## Question 2

[solution n°12 p.18]

Dans la suite du code (à l'emplacement `/*code Question 2*/`), écrivez des instructions permettant d'assembler dans une chaîne la marque, le nom du modèle, la ram et le processeur de l'ordinateur. La chaîne pourra ressembler à « *HP Pavillon, 512 GO de stockage, 16 GO de ram, processeur intel core i7* ». Bien évidemment, les données ne seront pas les mêmes quand vous changerez la valeur des variables.

Ensuite, définissez une variable **prix**. Celle-ci contiendra le prix de l'ordinateur qui sera calculé en additionnant le **stockage**, et la **ram** multipliée par 20. Enfin, rajouter le prix dans la chaîne concaténée pour que la chaîne ressemble à : « *HP Pavillon, 512 GO de stockage, 16 GO de ram, processeur intel core i7, prix : {prix} euros* ». Afficher la chaîne dans la console.

## B. Test

### Exercice 1 : Quiz

[solution n°13 p.19]

#### Question 1

Quel opérateur permet de vérifier l'égalité de 2 valeurs ?

- ☐ ==
- ☐ !=
- ☐ =

#### Question 2

Quel opérateur permet d'obtenir le reste d'une division euclidienne ?

- ☐ /
- ☐ \*
- ☐ %

#### Question 3

Que permet l'opérateur `||` ?

- ☐ De vérifier si 2 expressions renvoient **true**
- ☐ De vérifier si, pour 2 expressions, au moins une renvoie **true**
- ☐ De vérifier si 2 expressions sont fausses

#### Question 4

Que permet l'opérateur `&&` ?

- ☐ De vérifier si 2 expressions retournent **true**
- ☐ De changer la valeur d'une variable
- ☐ De retourner **true** dans le cas où une expression retourne **false**

#### Question 5

Quel opérateur permet de concaténer une chaîne ?

- ☐ +
- ☐ ++
- ☐ \*/

## **Solutions des exercices**





**p. 6 Solution n°1**

```
1 let a = 1376;  
2 let b = 630;  
3 console.log(a - b);
```

[cf.]

**p. 6 Solution n°2**

```
1 let nombre = 3;  
2 nombre++;  
3 console.log(nombre);
```

[cf.]

**p. 7 Solution n°3**

```
1 let marque = "HP";  
2 let ram = 16;  
3 console.log("L'ordinateur " + marque + " a " + ram + " GO de ram");
```

[cf.]

**p. 7 Solution n°4**

```
1 let nombre = 9;  
2 nombre *= 3;  
3 console.log(nombre);
```

[cf.]

**p. 7 Solution n°5**

```
1 let largeur = 9;  
2 let longueur = 10;  
3 let perimetre = 2 * largeur + 2 * longueur;  
4 console.log(perimetre);
```

[cf.]

**p. 10 Solution n°6**

```
1 let a = "Hello World";  
2 let b = "Hello World";  
3 console.log(a == b);
```

**p. 10 Solution n°7**

```
1 let prix = 1238;  
2  
3 console.log(prix > 800 && prix <= 1500);
```

**p. 10 Solution n°8**

```
1 let marque = "Btwin";
2
3 let prix = 800;
4
5 console.log((marque == "Btwin" || marque == "Rockrider") && prix < 500);
```

**p. 10 Solution n°9**

```
1 let marque = "Btwin";
2
3 console.log(marque != "Btwin");
```

**p. 11 Solution n°10**

```
1 let marque = "Btwin";
2
3 console.log(!(marque == "Btwin"));
```

**p. 13 Solution n°11**

Voici un code simple qui fonctionne :

```
1 let marque = "HP";
2 let modele = "Pavillon";
3 let stockage = 512;
4 let ram = 16;
5 let processeur = "intel core i7";
6
7 console.log((marque == "HP" || marque == "Acer") && stockage >= 256 && ram >= 8);
8
9 /*code Question 2*/
```

**p. 14 Solution n°12**

Voici un code qui fonctionne :

```
1 let marque = "HP";
2 let modele = "Pavillon";
3 let stockage = 512;
4 let ram = 16;
5 let processeur = "intel core i7";
6
7 console.log((marque == "HP" || marque == "Acer") && stockage >= 256 && ram >= 8);
8
9 let prix = stockage + (ram * 20) ;
10
11 let fiche = marque + " " + modele + ", " + stockage + " GO de stockage, " + ram + " GO de
12 ram, processeur " + processeur + ", prix : " + prix + " euros";
13 console.log(fiche);
```

## Exercice p. 14 Solution n°13

## Question 1

Quel opérateur permet de vérifier l'égalité de 2 valeurs ?

☒ ==

☐ !=

☐ =

 L'opérateur d'égalité == permet de vérifier si 2 valeurs sont égales.


## Question 2

Quel opérateur permet d'obtenir le reste d'une division euclidienne ?

☐ /

☐ \*

☒ %

 L'opérateur %, parfois appelé « modulo », permet d'obtenir le reste d'une division euclidienne.


## Question 3

Que permet l'opérateur || ?

☐ De vérifier si 2 expressions renvoient **true**

☒ De vérifier si, pour 2 expressions, au moins une renvoie **true**

☐ De vérifier si 2 expressions sont fausses

 L'opérateur || (OU) permet de vérifier si pour 2 expressions, au moins une des 2 renvoie **true**.


## Question 4

Que permet l'opérateur && ?

☒ De vérifier si 2 expressions retournent **true**

☐ De changer la valeur d'une variable

☐ De retourner **true** dans le cas où une expression retourne **false**

 L'opérateur && (ET) permet de vérifier si 2 expressions renvoient le booléen **true**.

## Question 5

Quel opérateur permet de concaténer une chaîne ?

☒ +

☐ ++

☐ \*/

Q L'opérateur `+` permet, lorsqu'on l'utilise avec des chaînes de caractères, de concaténer des chaînes. C'est alors un opérateur de concaténation. On peut aussi utiliser l'opérateur `+=` pour rajouter une chaîne à la fin d'une première chaîne de caractères.