Mémo JS

En HTML, le code JavaScript est inséré entre les balises <script>et .</script>

Pour utiliser un script externe, mettez le nom du fichier de script dans l' src attribut (source) d'une <script>balise :

<script src="myScript.js"></script>

JavaScript peut "afficher" les données de différentes manières :

Écrire dans un élément HTML, en utilisant innerHTML.

Écrire dans la sortie HTML à l'aide de document.write().

Écrire dans une boîte d'alerte, en utilisant window.alert().

Écrire dans la console du navigateur, en utilisant console.log().

document.getElementById("demo").innerHTML = 5 + 6;

document.write(5 + 6); L'utilisation de document.write() après le chargement d'un document HTML supprimera tout le code HTML existant : La méthode document.write() ne doit être utilisée qu'à des fins de test.

window.alert("toto")

À des fins de débogage, vous pouvez appeler la console.log()méthode dans le navigateur pour afficher les données.

Console.log("toto")

4 façons de déclarer une variable JavaScript :

Utilisant var → ne plus faire, c’était avant 2015

Utilisant let → variable structuré

Utilisant const → variable structurée

→ obligatoirement valorisée à la création

→ ne peut pas être réaffectée (x=x+2 interdit)

→ ne peut pas être réaffectée dans le même block

Ne rien utiliser

Toute variable peut être vidée, en définissant la valeur sur undefined. Le type sera également undefined.

car = undefined; // Value is undefined, type is undefined

Opérateurs arithmétiques JavaScript

Les opérateurs arithmétiques effectuent des opérations arithmétiques sur des nombres (littéraux ou variables).

Operator , Description

+ , Addition

- , Subtraction

\* , Multiplication

\*\* , Exponentiation (ES2016)

/ , Division

% , Modulus (Remainder)

++ , Increment

-- , Decrement

Opérateurs de comparaison JavaScript

Operator , Description

== , equal to

=== , equal value and equal type

!= , not equal

!== , not equal value or not equal type

> , greater than

< , less than

>= , greater than or equal to

<= , less than or equal to

? , ternary operator

Opérateurs logiques JavaScript

Operator , Description

&& , logical and

|| , logical or

! , logical not

Opérateurs de type JavaScript

Operator , Description

typeof , Returns the type of a variable

instanceof , Returns true if an object is an instance of an object type

Opérateurs bit à bit JavaScript

Les opérateurs de bits fonctionnent sur des nombres de 32 bits.

Tout opérande numérique dans l'opération est converti en un nombre de 32 bits. Le résultat est reconverti en nombre JavaScript.

Operator , Description , Example , Same as , Result , Decimal

& , AND , 5 & 1 , 0101 & 0001 , 0001 , 1

| , OR , 5 | 1 , 0101 | 0001 , 0101 , 5

~ , NOT , ~ 5 , ~0101 , 1010 , 10

^ , XOR , 5 ^ 1 , 0101 ^ 0001 , 0100 , 4

<< , left shift , 5 << 1 , 0101 << 1 , 1010 , 10

>> , right shift , 5 >> 1 , 0101 >> 1 , 0010 , 2

>>> , unsigned right shift , 5 >>> 1 , 0101 >>> 1 , 0010 , 2

Lors de l'ajout d'un nombre et d'une chaîne, JavaScript traitera le nombre comme une chaîne.

let x = 16 + 4 + "Volvo" → 20Volvo

let x = "Volvo" + 16 + 4; → Volvo164

let answer1 = "It's alright" → It's alright

let answer3 = 'He is called "Johnny"'-→ He is called "Johnny"

Vous pouvez utiliser l' typeofopérateur JavaScript pour trouver le type d'une variable JavaScript. L' typeofopérateur renvoie le type d'une variable ou d'une expression :

Une fonction JavaScript est définie avec le mot- functionclé, suivi d'un nom , suivi de parenthèses () . Les parenthèses peuvent inclure des noms de paramètres séparés par des virgules :( paramètre1, paramètre2, ... )

Le code à exécuter, par la fonction, est placé entre accolades : {}

function name(parameter1, parameter2, parameter3) {  
 // code to be executed  
 }

Lorsque JavaScript atteint une return instruction, la fonction s'arrête.

Si la fonction a été appelée à partir d'une instruction, JavaScript "reviendra" pour exécuter le code après l'instruction d'appel.

let x = myFunction(4, 3);   
 function myFunction(a, b) {  
 return a \* b; // Function returns the product of a and b  
 }

L'accès à une fonction sans () renverra l'objet de la fonction au lieu du résultat de la fonction :

document.getElementById("demo").innerHTML = myFunction → « function myFunction(a, b) { return a \* b; } »