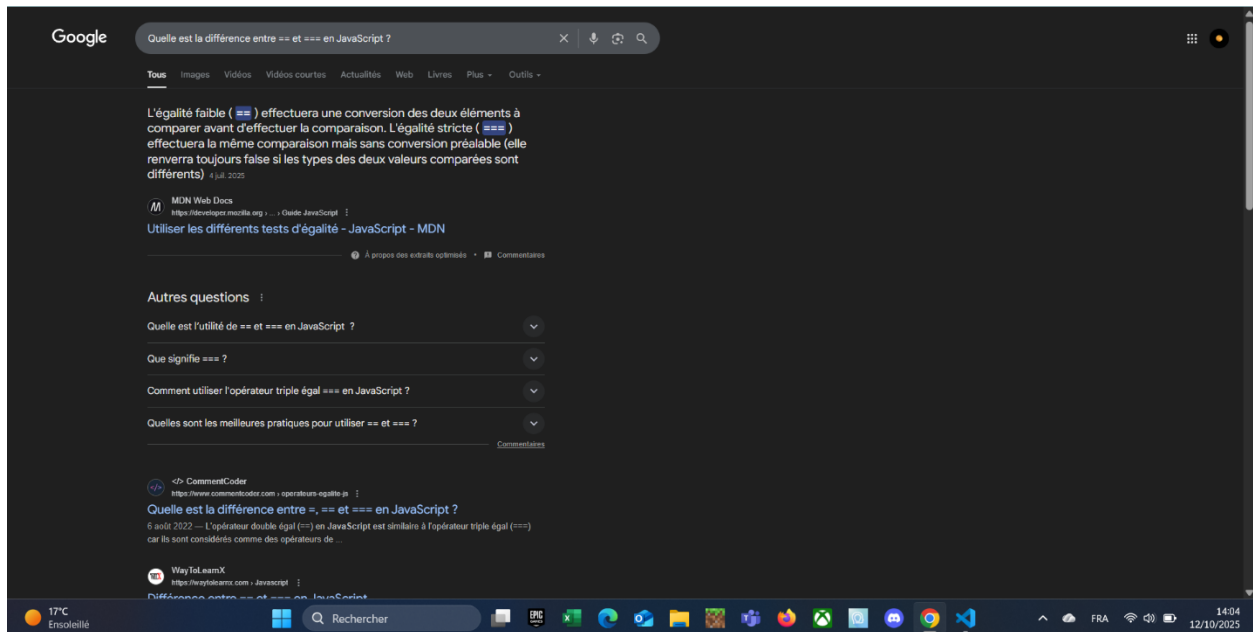


Résultats Google :



Google search results for the query "Quelle est la différence entre == et === en JavaScript ?". The top result is from MDN Web Docs, titled "Utiliser les différents tests d'égalité - JavaScript - MDN". Below it, there are "Autres questions" (Other questions) related to the topic. The bottom result is from "Way To Learn X" titled "Quelle est la différence entre ==, == et === en JavaScript ?".

Quelle est la différence entre == et === en JavaScript ?

L'égalité faible (==) effectuera une conversion des deux éléments à comparer avant d'effectuer la comparaison. L'égalité stricte (===) effectuera la même comparaison mais sans conversion préalable (elle renverra toujours false si les types des deux valeurs comparées sont différents) 4 juil. 2025

MDN Web Docs
https://developer.mozilla.org / ... / Guide JavaScript |
Utiliser les différents tests d'égalité - JavaScript - MDN

Autres questions :

Quelle est l'utilité de == et === en JavaScript ?

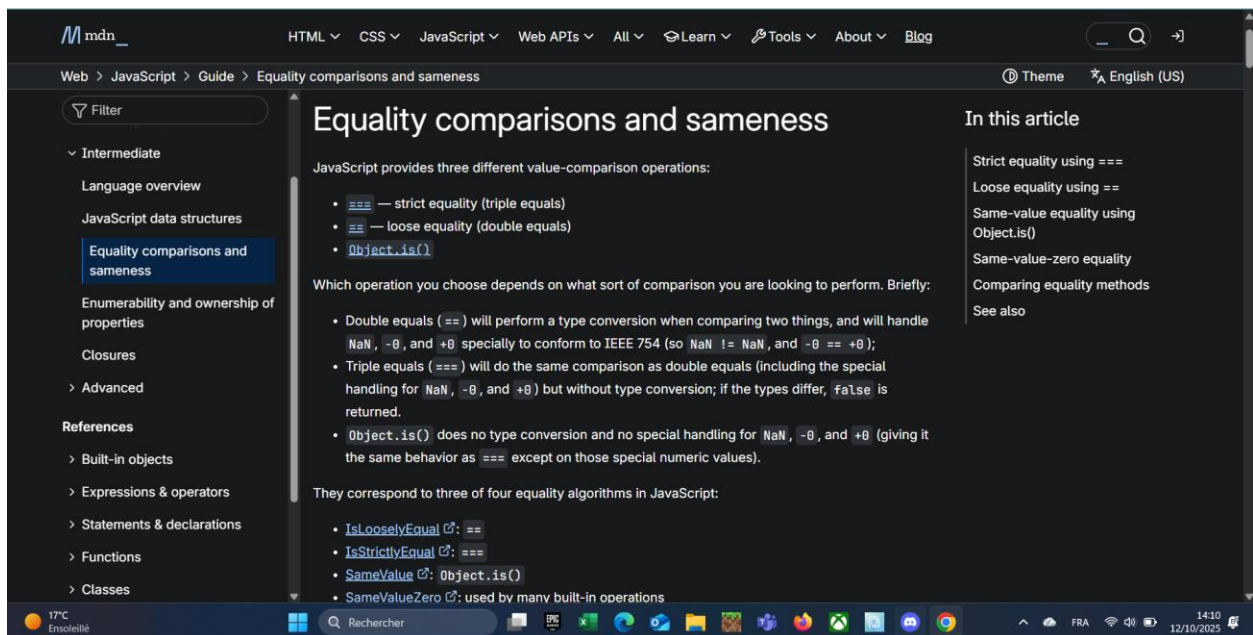
Que signifie === ?

Comment utiliser l'opérateur triple égal === en JavaScript ?

Quelles sont les meilleures pratiques pour utiliser == et === ?

CommentCoder
https://www.comment-coder.com / opérateurs-equalite-js |
Quelle est la différence entre ==, == et === en JavaScript ?
6 août 2022 - L'opérateur double égal (==) en JavaScript est similaire à l'opérateur triple égal (===) car ils sont considérés comme des opérateurs de ...

Way To Learn X
https://waytolearnx.com / Javascript |
Différence entre == et === en JavaScript



MDN Web Docs article titled "Equality comparisons and sameness". The article explains that JavaScript provides three different value-comparison operations: strict equality (===), loose equality (==), and Object.is(). It details how these operations handle type conversion and special values like NaN, -0, and +0. The article also mentions that these operations correspond to three of four equality algorithms in JavaScript: IsLooselyEqual, IsStrictlyEqual, and SameValueZero.

Web > JavaScript > Guide > Equality comparisons and sameness

Equality comparisons and sameness

JavaScript provides three different value-comparison operations:

- === — strict equality (triple equals)
- == — loose equality (double equals)
- Object.is()

Which operation you choose depends on what sort of comparison you are looking to perform. Briefly:

- Double equals (==) will perform a type conversion when comparing two things, and will handle NaN, -0, and +0 specially to conform to IEEE 754 (so NaN != NaN, and -0 == +0);
- Triple equals (===) will do the same comparison as double equals (including the special handling for NaN, -0, and +0) but without type conversion; if the types differ, false is returned.
- Object.is() does no type conversion and no special handling for NaN, -0, and +0 (giving it the same behavior as === except on those special numeric values).

They correspond to three of four equality algorithms in JavaScript:

- IsLooselyEqual: ==
- IsStrictlyEqual: ===
- SameValue: Object.is()
- SameValueZero: used by many built-in operations

In this article

- Strict equality using ===
- Loose equality using ==
- Same-value equality using Object.is()
- Same-value-zero equality
- Comparing equality methods
- See also

Résultat Chat GPT : lien réponse : <https://chatgpt.com/share/68eb992d-f0bc-8004-9f70-83dd493cf35b>

Quelle est la différence entre `==` et `===` en JavaScript ?

En JavaScript, la différence entre `==` et `===` est le type de comparaison effectuée :

- **`==` (égalité lâche)**
 - Compare les valeurs après avoir fait une conversion de type automatique (type coercion).
 - Cela signifie que JavaScript essaie de convertir les deux opérandes dans un type commun avant de les comparer.

Exemples :

```
js
5 == "5" // true → "5" est converti en nombre
0 == false // true → false est converti en 0
null == undefined // true → considérés égaux dans ce cas particulier
```

`Copier le code`

- **`===` (égalité stricte)**
 - Compare la valeur et le type.
 - Aucune conversion de type n'est effectuée : si les types sont différents, le résultat est toujours `false`.

Exemples :

```
js
5 === "5" // false → types différents (number vs string)
0 === false // false → types différents (number vs boolean)
null === undefined // false → types différents
5 === 5 // true → même type et même valeur
```

`Copier le code`

✓ En pratique :

- Il est fortement recommandé d'utiliser `===` (égalité stricte) dans la majorité des cas.
- Cela rend le code plus prévisible et évite les conversions implicites parfois trompeuses.

Résumé rapide :

Opérateur	Compare la valeur ?	Compare le type ?	Conversion automatique ?
<code>==</code>	✓ Oui	✗ Non	✓ Oui
<code>===</code>	✓ Oui	✓ Oui	✗ Non

Souhaites-tu que je te montre des cas concrets où `==` donne des résultats surprenants ?

🗨️ 📄 📌 🔄 ⋮

Pour moi, la réponse la plus claire la plus rapide à exploiter est celle de l'IAG, bien que plus courte, elle réponds directement à la question posée. Google nous liste des ressources pouvant contenir la réponse à notre question mais ils nous reste à trier les informations pertinentes dans celles-ci et à les interpréter par nous même. L'IAG quant à elle, fait le recoupement d'informations pour nous et nous retranscrit ça de manière plus concise et visuellement plus simple à comprendre et transposable à notre projet.