Diferencia entre == y ===. En java es igual que en javascript?

La diferencia principal entre `==` y `===` en JavaScript es que `==` compara solo el valor, mientras que `===` compara el valor y el tipo de dato. En Java, la comparación funciona de manera diferente y no existe el triple igual (`===`). Java usa `==` para comparar valores primitivos y referencias de objetos, pero para comparar el contenido de objetos hay que usar el método `.equals()` [1] [2] [3] [4].

JavaScript: `==` vs `===`

- `==` (doble igual): Compara los valores y realiza conversión de tipos si es necesario ("comparación débil"). Por ejemplo, `null == undefined` devuelve `true` porque ambos representan ausencia de valor, aunque sean de tipo distinto [5] [6].
- `===` (triple igual): Compara tanto el valor como el tipo ("comparación estricta"). `null === undefined` devuelve `false`; `5 === "5"` también es `false` porque los tipos son distintos [5] [7].
- Se recomienda usar `===` para evitar resultados inesperados debido a conversiones automáticas de tipo ("coerción") [1] [5] [8].

Java: Comparaciones

- Para tipos primitivos (`int`, `double`, `char`, etc.), `==` compara el valor directamente [9] [4].
- Para objetos, `==` compara si las referencias son iguales (si apuntan al mismo objeto en memoria). Por ejemplo, dos cadenas creadas con `new String()` tendrán el mismo contenido pero referencias distintas, por lo que `str1 == str2` será `false` [3] [9].
- Para comparar el contenido de dos objetos, se usa el método `.equals()`, ejemplo: `str1.equals(str2)` [3] [9] [10].

Tabla Comparativa

Operador	JavaScript	Java
`==`	Compara valor y hace conversión de tipo (débil) [1]	Compara valor en primitivos, referencia en objetos [4]
`===`	Compara valor y tipo (estricto) [7]	No existe
`.equals()`	No existe	Compara contenido de objetos [3] [10]

Resumen

En JavaScript, `==` y `===` tienen comportamientos distintos por la tipificación débil del lenguaje, mientras que en Java toda comparación entre objetos debe tener en cuenta si se trata de valor o referencia, usando `==` solo para comparación de referencias y `.equals()` para contenido[3] [11] [9]. Por tanto, `===` **en JavaScript no existe como tal en Java**; la comparación segura de contenido en Java se realiza mediante `.equals()` y no con operadores de igualdad como en JavaScript[3] [10] [12].

⁂

1. https://es.stackoverflow.com/questions/331/diferencias-entre-y-comparaciones-en-javascript
2. https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Equality
3. https://academiasanroque.com/comparaciones-en-java-equals-equalsignorecase-y-mas/
4. https://codersfree.com/posts/operadores-relacionales-en-java
5. https://frontendleap.com/fundamentales/comparaciones-javascript
6. https://www.uv.es/jac/guia/jscript/javascr04.htm
7. https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Strict_equality
8. https://rubentejera.com/diferencias-entre-doble-igual-y-triple-igual-en-javascript/
9. https://www.arquitecturajava.com/comparando-java-vs-equals/
10. https://openwebinars.net/blog/introduccion-a-java-operadores/
11. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/java-vs-javascript/
12. https://www.campusmvp.es/recursos/post/Comparando-valores-y-referencias-en-varios-lenguajes-de-programacion.aspx
13. image-1.jpg
14. https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_operators
15. https://www.hackaboss.com/blog/diferencias-javascript-java
16. https://www.manualweb.net/java/operadores-igualdad-relacionales-java/
17. https://www.startechup.com/es/blog/java-vs-javascript/
18. https://www.youtube.com/watch?v=ucBDe9ct6fU
19. https://www.eniun.com/javascript-java-diferencias-estructuras-control/
20. https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Equality_comparisons_and_sameness
21. https://es.javascript.info/comparison