**JavaScript no obstructivo** es un [paradigma](https://es.wikipedia.org/wiki/Paradigma) floreciente en el uso del [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [JavaScript](https://es.wikipedia.org/wiki/JavaScript), utilizado en la [Web](https://es.wikipedia.org/wiki/World_Wide_Web). Aunque el término no se define formalmente, sus principios generalmente incluyen:

* [Separación](https://es.wikipedia.org/wiki/Separaci%C3%B3n_de_intereses) de la funcionalidad JavaScript (la "capa del comportamiento") de las capas de estructura/contenido y de presentación de una página.
* Uso de [buenas prácticas](https://es.wikipedia.org/wiki/Buenas_pr%C3%A1cticas) a fin de evitar los problemas de incompatibilidad de la programación tradicional en JavaScript (tales como inconsistencias entre [navegadores](https://es.wikipedia.org/wiki/Navegador_web) y falta de [escalabilidad](https://es.wikipedia.org/wiki/Escalabilidad))

**El valor null y undefined:**

El valor null es un literal de Javascript que representa intencionalmente un valor nulo o "vacío". Es uno de los [valores primitivos](https://developer.mozilla.org/es/docs/Glossary/Primitive) de Javascript.

La propiedad global **undefined** representa el valor primitivo [undefined](https://developer.mozilla.org/es/docs/Glossary/undefined). Es uno de los [valores primitivos](https://developer.mozilla.org/es/docs/Glossary/Primitive) de JavaScript.

**¿En que se diferencian?**

Tanto null como undefined en JavaScript son valores bastante extraños, sobre todo para los principiantes. Ambos sirven para algo muy parecido, que es **indicar la ausencia de valor**. Lo que ocurre es que **undefined** es un valor que denota que no hay valor **porque no se ha definido todavía**, mientras que **null** se usa para indicar que no hay valor **porque así lo hemos querido indicar expresamente**.

La consola su valor, que es undefined, que quiere decir que la variable está declarada pero no se ha definido aún su valor.

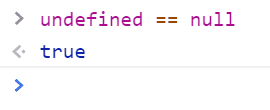
Sin embargo, imagina que ya has utilizado una determinada variable y no quieres utilizarla más, o que tienes asignada una referencia a un objeto pero ya no lo necesitas o que, directamente, quieres tener la variable declarada pero inicializarla con un valor que todavía no sabes qué va a ser exactamente. En todos estos casos lo suyo es asignarle un valor nulo:

var miVariable = null;console.log(miVariable);

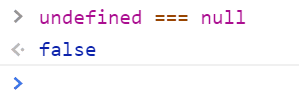
que en este caso mostrará el valor null por consola.

Ambos valores son valores [de tipo "falsoso"](https://www.jasoft.org/Blog/post/Booleanos-pseudo-booleanos-operadores-y-condicionales-en-JavaScript-mucho-mas-complejo-de-lo-que-parece.aspx), es decir, que actúan como false en las comparaciones. También ambos **son tipos básicos de JavaScript**, es decir, tipos primitivos como los números o las cadenas, parte intrínseca del lenguaje.

Una cosa curiosa de su uso es **la comparación entre ambos valores**. Precisamente por las particularidades que tienen, JavaScript a veces tiene que tratar estos dos tipos de manera especial también. Si haces una **comparación no estricta** entre ellos (con ==), como ambos significan la ausencia de valor, la comparación devuelve un true:



O sea, que aparentemente los considera iguales. Sin embargo, cuando haces una **comparación estricta** (con ===), dado que en realidad **no son lo mismo**, te devuelve un false:



**Que cuidados se debe tener al usar null y undefined:**

Al verificar si hay null o undefined, tenga cuidado con las [diferencias entre los operadores de igualdad (==) e identidad (===)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators) , ya que el primero realiza la conversión de tipo.