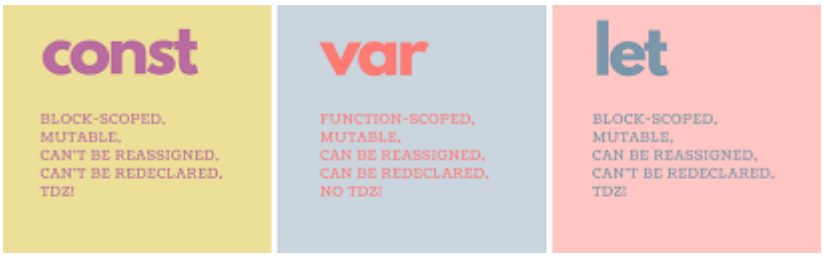
JavaScript ES6: **var** vs **let** vs **const**



Es por esto que estoy a favor de *const* sobre *let* en ES6. En JavaScript, ***const* significa que el identificador no puede ser reasignado.** (No debe confundirse con *valores inmutables*. A diferencia de los tipos de datos inmutables verdaderos como los producidos por Immutable.js y Mori, un objeto *const* puede tener propiedades mutadas).

Uso **let** cuando necesito reasignar una variable. Como **utilizo una variable para representar una cosa,** el caso de uso para *let* tiende a ser para bucles o algoritmos matemáticos.

No uso *var* en ES6. Hay valor en ámbito de bloque de bucles, pero no puedo pensar en una situación en la que yo preferiría *var* sobre *let*.

- **const** es una señal de que **el identificador no será reasignado.**
- **let,** es una señal de que **la variable puede reasignarse**, como un contador en un bucle, o un intercambio de valores en un algoritmo. También indica que la variable se usará **solo en el bloque en el que está definida**, que no siempre es la función que lo contiene.
- **var** ahora es **la señal más débil disponible** cuando defines una variable en JavaScript. La variable puede reasignarse o no, y la variable puede o no ser utilizada para una función completa, o solo para un bloque o bucle.

Warning Con *let* y *const* en ES6, ya no es seguro para comprobar la existencia de un identificador usando **typeof***:*

```
función foo () {
  typeof bar;
  dejar bar = 'baz';
}
foo (); // ReferenceError: no se puede acceder a la declaración léxica
       // `bar 'antes de la inicialización
```

Ver también:
https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/let