FUNCIÓN eval() javascript — Alternativas

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/eval

No utilice `eval` innecesariamente

`eval()` es una **función peligrosa**, que ejecuta el código el cual es pasado con los privilegios de quien llama.

Si ejecuta `eval()` con una cadena de caracteres que podría ser afectada por un elemento malicioso, **podría terminar ejecutando código malicioso dentro de la computadora del usuario con los permisos de su página o extensión web**.

Más importante aún, una parte del código de terceros podría acceder al ámbito en el cual `eval()` fue invocada, lo que puede permitir la entrada a posibles ataques de formas a las que el constructor `Function` (art. en inglés) el cual es similar, no es susceptible.

`eval()` **es generalmente también más lenta que otras alternativas** desde que se invoca en el intérprte de JS, mientras que otros muchos constructores son optimizados por los motores JS modernos.

Existen alternativas más seguras (y rápidas) que `eval()` para casos de uso común.

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Function

Function

Cada función de JavaScript en realidad es un objeto `Function`. Esto se puede ver con el código `(function() {}).constructor === Function`, que devuelve `true`.

Constructor

Function()

> Crea un nuevo objeto `Function`. Llamar al constructor directamente puede crear funciones dinámicamente, pero tiene problemas de seguridad y de rendimiento similares (pero mucho menos importantes) para `eval`. Sin embargo, a diferencia de `eval`, el constructor `Function` crea funciones que solo se ejecutan en el ámbito global.

https://www.educative.io/edpresso/eval-vs-function-in-javascrip

An alternative to eval is Function()
eval() vs. Function() in JavaScript

Function() takes some expression as a string for execution, except, rather than outputting the result directly, it returns an anonymous function to you that you can call.

Function () toma alguna expresión como una cadena para su ejecución, excepto que, en lugar de generar el resultado directamente, le devuelve una función anónima a la que puede llamar.

Ej.:
let userInput = "2+4";
let result = Function("return " + userInput)(); // which is same as "return 2+4" console.log(result)
// resultado 6 en consola

t