INTRODUCCION ECMASCRIPT 6

Ecma International es una organización sin ánimo de lucro encargada de regular el funcionamiento de varios estándares en la industria de la computación. Así surge ECMAScript 1 (ES1) en 1997 como la primera versión del estándar de JavaScript. Se usa el término ECMAScript para referirse al estándar y JavaScript para hablar del lenguaje en la práctica. La versión 5 salio el 2009.

Actualmente la versión ES5 es compatible con todos los navegadores. Esta versión usa las características de ES6 mediante un proceso de transpiling y polyfilling que convierte el código en ES5, garantizando así la compatibilidad del código en navegadores viejos.

LET Y CONST

ES6 introduce dos nuevas formas de crear variables con let y const. La principal diferencia entre var y let es el scope. Las sentencias de bloque no crean un scope local.

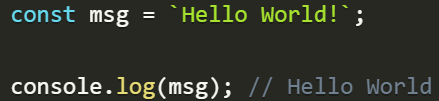
En JavaScript todas las variables son inicializadas con undefined al momento de su creación. Esta es otra diferencia entre let y var. Si se trata de acceder a una variable con let antes de ser declarada obtenemos un ReferenceError en lugar de undefined.

const es muy similar a let, la única diferencia es que una vez asignado un valor a una variable ya no se puede reasignar. Declarar un objeto con const no significa que no podamos cambiar sus propiedades, lo que no podemos hacer es asignarle un nuevo valor.

TEMPLATE STRINGS

Las plantillas de texto o template strings, son cadenas de texto que permiten interpolación mediante expresiones. Hacen mucho más fácil crear textos en los que necesitamos integrar variables o expresiones.

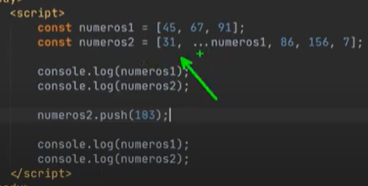
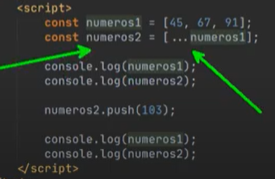
La sintaxis consta de dos partes, la primera es para delimitar la cadena de texto, se usan comillas invertidas. La segunda parte es para agregar placeholders mediante el uso del signo de dólar y llaves.



SPREAD OPERATOR

El operador Spread permite que un iterable se expanda en lugares donde se esperan 0+ argumentos. Se utiliza sobre todo en el array de variables donde se esperan más de 1 valores. Nos permite el privilegio de obtener una lista de parámetros de un array. La sintaxis del operador Spread es la misma que la del parámetro Rest, pero funciona de forma completamente opuesta.

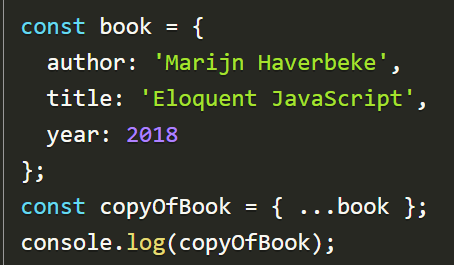
Permite que los arreglos creados con spread a pesar de ser clonados en un principio, no se referencian al mismo arreglo, sino que podemos trabajar de forma aislada con cada uno.



COPIAR ITERABLES

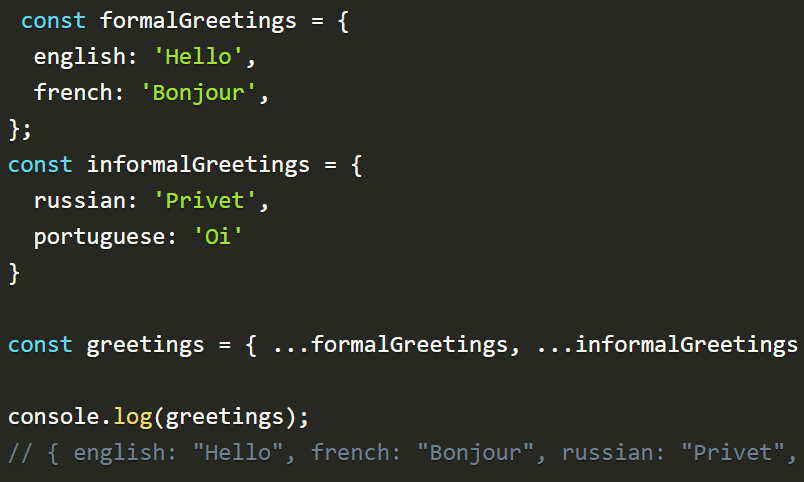
Uno de los usos más comunes de este operador es duplicar arreglos. Después de declarar el nombre de la variable usamos corchetes para asignar un nuevo arreglo y dentro colocamos el spread operator para obtener todos los elementos del arreglo que queremos copiar.

En el caso de objetos es casi lo mismo, la única diferencia es el uso de llaves en lugar de corchetes.



UNIR ITERABLES

También podemos usar el spread operator para concatenar arreglos. Hay que tomar en cuenta que la posición donde se coloque el spread afecta el orden de los elementos en el arreglo. Es muy similar en el caso de objetos. En el caso de tener propiedades duplicadas se sobrescriben.

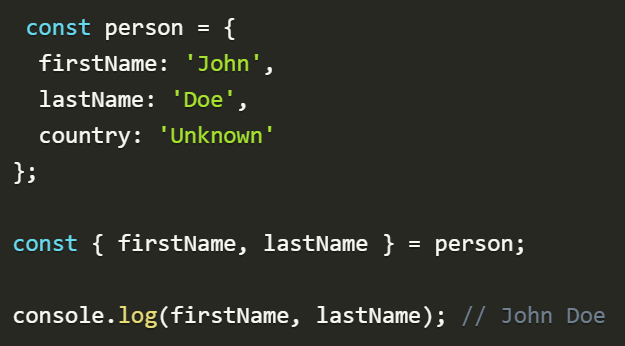


DESTRUCTURING

Destructuring es extraer valores o propiedades de un arreglo u objeto.

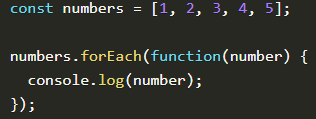
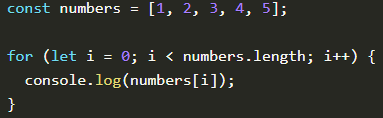
En ocasiones queremos crear variables a partir de propiedades de un objeto. Podemos lograr lo mismo en una sola línea.

Las llaves del lado izquierdo del = no son un objeto. Esta es la sintaxis de object destructuring. Estamos creando dos nuevas variables firstName y lastName, después estamos extrayendo dos propiedades de person con el mismo nombre de las variables, el valor de esas propiedades es el que se asigna a las variables creadas.

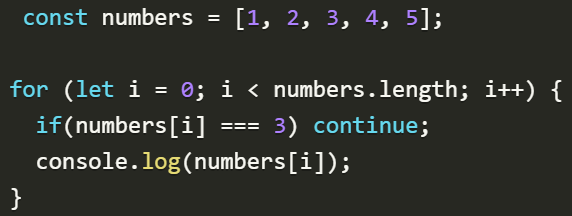


ARREGLOS

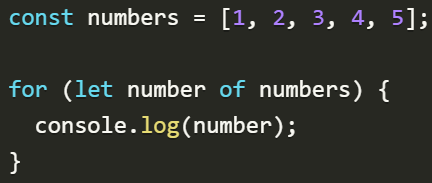
Cuando trabajamos con arreglos, solemos realizar el recorrido por medio de un bucle for. Una alternativa es usar el método forEach, el cual ejecuta una función por cada elemento del arreglo.



Una desventaja de usar forEach es que no podemos implementar break o continue para saltarnos una iteración o salir del bucle. Si quieres salir del arreglo debes usar la palabra continue.



ES6 introduce la sentencia for...of, esta es una combinación entre for y forEach ya que nos permite ejecutar un bloque de código para cada elemento de un objeto iterable. Es decir, funciona con arreglos, strings y otros objetos similares a un arreglo como NodeList. Similar a forEach, en cada iteración se crea una variable que corresponde al elemento del arreglo por lo que no necesitamos hacer algo como numbers[i].



También podemos manipular las iteraciones con break o continue.

