



Mayo 1995: Brendan Eich crea un lenguaje Ilamado "Mocha"



Mayo 1995: Brendan Eich crea un lenguaje Ilamado "Mocha"



Sept. 1995: Se renombra como LiveScript



Mayo 1995: Brendan Eich crea un lenguaje Ilamado "Mocha"

Dic. 1995: Se renombra como JavaScript

Sept. 1995: Se renombra como LiveScript



Mayo 1995: Brendan Eich crea un lenguaje Ilamado "Mocha"

Dic. 1995: Se renombra como JavaScript

Sept. 1995: Se renombra como LiveScript

Nov. 1996: Netscape propuso la estandirazación de JavaScript a Ecma International



Mayo 1995: Brendan Eich crea un lenguaje Ilamado "Mocha"

Dic. 1995: Se renombra como JavaScript

Jun. 1997: Esto resultó en un nuevo lenguaje, conocido como EcmaScript

Sept. 1995: Se renombra como LiveScript

Nov. 1996: Netscape propuso la estandirazación de JavaScript a Ecma International



Mayo 1995: Brendan Eich crea un lenguaje Ilamado "Mocha"

Dic. 1995: Se renombra como JavaScript

Jun. 1997: Esto resultó en un nuevo lenguaje, conocido como EcmaScript

Sept. 1995: Se renombra como LiveScript

Nov. 1996: Netscape propuso la estandirazación de JavaScript a Ecma International

ECMAScript es el estándar y JavaScript es la implementación más popular



# ES\_?

**ES** es una abreviación de **ECMAScript**. Siempre que vemos **ES** seguido de un número, está hablando de una versión de ECMAScript

• ES1: 1997

• ES2: 1998

• ES3: 1999

• ES4: No se publicó

• ES5: 2009

• ES6 / ES2015: 2015

• ES7 / ES2016: 2016

• ES8 / ES2017: 2017

• ES9 / ES2018: 2018



#### **SCOPE DE VARIABLES**

El scope de una variable hace referencia al lugar donde va a vivir esta, o podrá ser accesible.





# **SCOPE DE VARIABLES**

var tiene un scope de función

```
if (true) {
 var foo = 'bar';
console.log( foo );
// bar
```

```
function variable () {
var foo = 'bar';
// ejecuto la función
variable();
console.log( foo );
// Uncaught ReferenceError: foo is not
defined
```



# **LET Y CONST**

let y const son una nueva forma de crear variables.

En un principio, funcionan igual que var, pero con una principal diferencia: **let** y **const** tienen **scope de bloque** 

```
var a = 1;
let b = 2;
const c = 3;
console.log(a); // 1
console.log(b); // 2
console.log(c); // 3
```



# LET

let funciona igual que var, con la única diferencia que tiene scope de bloque

```
if (true) {
var foo = 'bar';
console.log( foo );
// bar
```

```
console.log( foo );
// Uncaught ReferenceError: foo is not
defined
```



# **CONST**

const funciona como let, pero una variable creada como const no puede ser REASIGNADA

```
if (true) {
 const foo = 'bar';
console.log( foo );
// Uncaught
ReferenceError: foo
is not defined
```

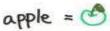
```
const foo = 'bar';
foo = 'barrr';
// Uncaught TypeError: Assignment to constant
variable.
```







a thing in a box hamed "apple"

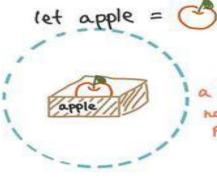




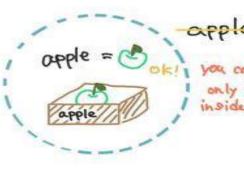
you can swap item later

# let

var



a thing in a box normed "apple" w/ protection shield



apple = Cong you can swap item only if you ask inside of the shield

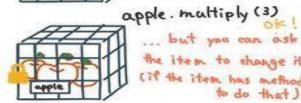
# const



const apple = 0



apple = 6 you can't swap item later.



... but you can ask the item to change itself ( if the item has method

to do that)



# LET, CONST... ¿CUANDO USO CADA UNA?

La única diferente entre **const** y **let**, es que cuando usamos **const** hace que la variable no pueda ser reasignada.

# Mathias Bynens - V8 Engineer @ Google:

- Usar **const** por defecto.
- Solo usar **let** si necesitamos que esa variable pueda ser reasignada.
- var no debería ser usado cuando utilizamos ES6



#### **EJERCICIOS**

1. Cambiar el siguiente código para que todas las variables se creen usando let o const.

```
function mostrarVariasVeces(str) {
 var oracion = str + " es lo mejor!";
 for (var i = 0; i < str.length; i += 2) {</pre>
   console.log(oracion);
mostrarVariasVeces("ES6");
```



# **EJERCICIOS**

2. El siguiente código, busca el número más grande dentro de un array. ¿Cómo cambiarías el código para utilizar sólo **let** y/o **const**?

```
var array = [4, 2, 1, 7, 5];
var max = array[0];
for (var i = 0; i < array.length;</pre>
i++) {
var el = array[i];
 if (el > max) {
   max = array[i];
console.log( max );
```



Los template literals es una nueva forma de crear strings.

Son strings que permiten expresiones y soportan simple/múltiple línea.

Se crean con las comillas invertidas ``.

```
// ES5
var mensaje = 'Hola mundo!';

// ES6
const mensaje = `Hola mundo!`;
```



Nos permiten escribir texto de varias líneas de una forma más sencilla.

```
// ES5
var mensaje =
'<div>\n' +
'  Hola mundo!  \n' +
'</div>';
console.log(mensaje);
// <div>
// Hola mundo!
// </div>
```

```
// ES6
const mensaje = `
<div>
 Hola mundo!
</div>
console.log(mensaje);
// <div>
// Hola mundo!
// </div>
```



Nos permiten escribir expresiones, o lo que se conoce como interpolación de strings.

Las variables (o valores interpolados) se indican con el signo pesos y llaves \${}

```
// de la vieja forma
var nombre = 'Ada';
var saludo = '¡Hola, ' + nombre + '!';
console.log( saludo );
// ¡Hola, Ada!
```

```
// de la nueva forma
const nombre = 'Ada';
const saludo = `;Hola, ${nombre}!`;
console.log( saludo );
// ¡Hola, Ada!
```



En las interpolaciones, también podemos utilizar expresiones de JS.

```
const IVA = 0.21;
const precio = 19.99;

const total = `Precio final: ${precio + (precio * IVA)}`;

console.log(total);

// Precio final: 24.1879
```



1. Utilizá template literals para obtener el output esperado a partir de las variables: "Ada es programadora."

```
const nombre = "Ada";
const profesion = "programadora";
console.log() // completá el código acá
```



2. Tenemos dos números. El primero representa el precio original de un producto. El segundo número representa el precio después de haber aplicado un descuento.

Necesitamos calcular el porcentaje que una persona ahorra si compra con descuento.

```
const precioOriginal = 25;
const precioConDescuento = 10;

console.log(); // escribir acá la solución

// "Ahorraste un 60% con este descuento."
```



3. Utilizá **template literals** para actualizar y mejorar el siguiente código. Además, cambia var por **let** o **const** siguiendo los criterios vistos antes.

```
var personaje = 'Batman';
var identidad = 'Bruce Wayne';
var oracion = 'La identidad secreta de ' + personaje + ' es ' +
identidad.toUpperCase() + '.';
console.log( oracion );
// "La identidad secreta de Batman es BRUCE WAYNE."
```



4. Utilizá **template literals** para actualizar y mejorar el siguiente código. Además, cambia var por **let** o **const** siguiendo los criterios vistos antes.

```
var nombreUsuario = 'Grace';
var idUsuario = 2;
var espacioUsado = 10;
var espacioTotal = 11;
var url = 'https://drive.google.com/' + idUsuario + '/buy';
var email = ';Hola, ' + nombreUsuario + '!\n' +
'Estás utilizando el ' + (espacioUsado / espacioTotal * 100) + '% de tu espacio.\n' +
'Solo te quedan ' + (espacioTotal - espacioUsado) + ' GB. \n' +
'\n' +
'Podés comprar más espacio ingresando al a siquiente dirección: ' + url + '\n' +
'\n' +
'Saludos,\n' +
'Google';
```



# ¡Gracias!



