Variables en Javascript / Nombres de Variables

Requisitos para nombrar variables:

- Puede contener solo letras, números y los símbolos `$` y `_`
- El primer carácter no puede ser un número
- Los nombres de variables son sensibles a mayúsculas y minúsculas Las variables `miNombre` y `MiNombre` y `MiNOMBRE` son diferentes

Ejemplos válidos de variables:

```
let v10 =                // Nombre de variable válido
let edad = 50;           // Nombre de variable válido
let $ = 1;               // $ es permitido como nombre de variable
let _ = 2;               // _ es permitido como nombre de variable
let 1a = 1;              // Inválido, no puede empezar con número
let mi-nombre = 'Pepito' // Inválido, el guión no es permitido
```

Palabras reservadas que no pueder ser nombre de variable:

break	export	super	else	return	yield
case	extends	switch	do	new	with
catch	finally	this	delete	instanceof	while
class	for	throw	default	in	void
const	function	try	debugger	import	var
continue	if	typeof			

Variables en Javascript / Creación

Para crear variables usamos las palabras clave:

- **var**: Declaración *clásica* usada anteriormente en ES5
- **let**: Declaración *moderna* de variables a partir de ES6
- **const**: Declaración de constantes

Diferencias de alcance entre var, let y const:

- **var**: La define local en una función (sin importar el bloque) o fuera de una función pasa a ser global.
- **let**: Limita el alcance al bloque o expresión donde se declare.
- **const**: Limita el alcance al bloque y su valor no puede ser cambiado.

> Variable Global:
> Las variables que se definen afuera de cualquier función se transforman en propiedades del objeto window y serán globales
> Estarán disponibles en todo el programa, dentro de cualquier bloque o función,

```
let nombre;                    // Creación de a una variable
nombre='Juan';                 // Asignación de valor a una variable

let apellidop='Perez';          // Creación y asignación en una sola línea:
let apellidom='Gomez', edad=18;  // Creación-Asignación variables en una sola línea:

var telefono='867123456';       // Creación-Asignación con var:

const MAXWIDTH=400;             // Creación y asignación de constantes:
```

Variables en Javascript / Ambito (var)

Si declaramos una variable con var dentro de una función, esta será local a la función:

```
function Hola() {
  var saludo = "Hola";    // saludo es variable local en la función
  alert(saludo);          // Imprime Hola
}
Hola();
alert(saludo);            // Error: la variable saludo no está definida
```

Si omitimos la declaración con var dentro de una función, la variable pasará a ser global:

```
function Hola() {
  saludo = "Hola";        // Si omitimos la declaración con var saludo sera global
  alert(saludo);          // Imprime Hola
}
Hola();
alert(saludo);            // Imprime Hola
```

A las variables declaradas con var *no les importan* los bloques:

```
if (true) {
  var saludo = "Hola";  // saludo es visible fuera del bloque if
}
alert(saludo);          // Imprime Hola
```

Variables en Javascript / Ambito (let)

Al contrario la declaración con let tiene alcance de bloque:

```
if (true) {
  let saludo = "Hola"; // saludo sólo es visible dentro del bloque if
  alert(saludo)        // Imprime Hola
}
```

```
if (true) {
  let saludo = "Hola"; // saludo sólo es visible dentro del bloque if
}
alert(saludo);         // Error: la variable saludo no está definida
```

Variables en Javascript / Hoisting (var)

Las declaraciones con var son procesadas al inicio de la función:
En Javascript podemos escribir el siguiente código:

```
function saludar() {
   frase = "Hello";

   alert(frase);

   var frase;
}

saludar();
```

Y Javascript *lo verá* de la siguiente manera:

```
function saludar() {
    var frase;

    frase = "Hello";

    alert(frase);
}

saludar();
```

Javascript elevó (hoist) la declaración al inicio de su ámbito. Este comportamiento se llama Hoisting (Elevación)
Las declaraciones de variables con `var` son elevadas al inicio de la función.

Variables en Javascript / Hoisting (var)

Javascript elevará la declaración de una variable con `var`
Incluso si la declaración se hace dentro de un bloque de código.
En el siguiente ejemplo la declaración de la variable está dentro de un bloque de código if.

```
function saludar() {
   frase = "Hello";

   if (true) {
      var frase;
      alert(flase)
   }
}

saludar();
```

Aun así Javascript elevará la declaración de la variable al inicio de su ámbito es decir al inicio de la función.

```
function saludar() {
    var frase;
    frase = "Hello";

    if (false) {
      alert(frase);
    }
}

saludar();
```

Variables en Javascript / Hoisting (var)

En este conmportamiento (hoisting), las declaraciones son elevadas al inicio de la función.
pero las asignaciones no. Observemos el siguiente código:

```
function saludar() {
  alert(frase);             // Imprime Undefined

  var frase = "Hola";       // Declaración y asigación de valor a la variable
}

saludar();
```

El código anterior, Javascrit *lo verá* así:

```
function saludar() {
    var saludo;             // La declaración es "elevada"

    alert(saludo);          // Imprime Undefined

    saludo = "Hola";        // La asignación "se queda en su lugar"
}

saludar();
```

> Debido a que las variables declaradas con var son *elevadas* al inicio de la función
> podemos referenciarlas en cualquier parte de la función, pera su valor será `Undefined`
> hasta que les asignemos un valor.

Variables en Javascript / Omisión de declaración y Declaracion Repetida)

Omisión de la declaración

Javascript generará un error sí:

- Se trata de leer una variable que no ha sido declarada
- Se trata de asignar valor a una variable no declarada (en modo strict)

> En modo no-strict si se asgina valor a una variable no declarada
> Javascript creará esa variable como propiedad del objeto global.
> Pero esto es una mala (pésima!) práctica y siempre se deben declarar las variables (anteriormente) con `var` y ahora con `let`

Declaraciones repetidas

Con `var` es permitido declarar una variable mas de una vez.
Si la declaración que está repetida tiene una asignación
simplemente cambiará el valor.

Con `let` no se permite una declaración repetida (en un mismo bloque).
Javascript lanzará un error de sintaxis.