

Fundamentos de Programación con JavaScript

03

Variables

JS



Variables. Constantes.

Cuando programamos, podemos crear variables que son pequeños espacios de memoria a los que ponemos un nombre como "nombreVariable", y que nos permite almacenar datos de distintos tipos que posteriormente podremos cambiar y manipular.

Las variables en los lenguajes de programación son similares a las variables en matemáticas. Como hemos dicho, las variables son contenedores de información: texto, números, funciones, programas (podríamos decir que cualquier cosa). Las variables no se destruyen, es necesario asignarles el valor *null*.

De la misma forma que si en Matemáticas no existieran las variables no se podrían definir las ecuaciones y fórmulas, en programación no se podrían hacer programas útiles sin ellas.

Las variables en JavaScript se crean (declaran) mediante la palabra reservada **var**. En realidad hay tres tipos de declaraciones: **var**, **let** y **const**. Estas dos últimas forman parte de ECMAScript 2015 (ES6) standard.

Variables. Constantes.

La palabra reservada **var** se utiliza para declarar una o más variables (separadas por coma) y, opcionalmente, inicializarla/s a un valor.

Si cuando se declara una variable se le asigna también un valor, se dice que la variable ha sido inicializada. En JavaScript no es obligatorio inicializar las variables, ya que se pueden declarar por una parte y asignarles un valor posteriormente.

```
SINTAXIS: var nombre1 [= valor1], [nombre2 [= valor2],...;
nombreN : Nombre de variable. Puede ser cualquier identificador legal.
valorN : Valor inicial de la variable. Puede ser cualquier expresión legal.
```

Si no declaramos una variable, JavaScript lo admite, pero es una muy mala práctica, porque todas las variables creadas así son globales, de manera que se recomienda declarar todas las variables que se vayan a utilizar.

Variables. Constantes.

En JavaScript no se indica el tipo de variable al declararla, por eso decimos que es un lenguaje de tipado dinámico. JavaScript entiende que una variable es de un tipo determinado cuando le asignamos un valor; por ejemplo si le asignamos un número (sea entero o real) JavaScript entiende que es de tipo Number. Si el valor cambia entonces cambia su tipo.

El nombre de una variable también se conoce como identificador y debe cumplir las siguientes normas:

- Sólo puede estar formado por letras, números y los símbolos \$ (dólar) y _ (guión bajo).
- El primer carácter no puede ser un número.
- Debemos usar nombres que expliquen que aporta esa variable a nuestro código.
- No se pueden usar espacios.
- JavaScript sigue la norma camelCase para los nombres (primera letra en minúsculas y comienzo de nueva palabra en mayúsculas).

Variables, conceptos:

- Una **variable** es un espacio en memoria donde **almacenamos temporalmente un dato** que utilizaremos más tarde en las operaciones que nos interese a lo largo de nuestro programa.
- Este dato podrá ser recuperado en cualquier momento, tantas veces como queramos y podremos cambiarlo cuando sea necesario.
- Las variables se pueden: **declarar**, **inicializar** y **modificar**
- **Declarar** una variable es ponerle **nombre**
- Para declarar una variable utilizamos la palabra reservada **let** (no se utiliza **var**)
- Para declarar una constante utilizamos la palabra reservada **const**
- **Ámbito** (scope): zona donde existe, es decir, donde podemos utilizar su valor
- **Elevación** (hoisting): ámbito global, cuando usamos la palabra reservada **var**



Variables: declarar, inicializar y modificar

- **Declarar** una variable es ponerle **nombre**

Let `numero`;

- **Inicializar** una variable es asignarle un valor

`numero = 7;`

- Se puede declarar e inicializar al mismo tiempo

Let `numero = 7;`

- **Modificar** el valor de una variable **existente**

`numero = 25;`

- Las **constantes** deben declararse e inicializarse al mismo tiempo

const `PI = 3,1416;`

Variables en Javascript

Declaración

Inicialización

Las **constantes** deben declararse e inicializarse al mismo tiempo.
Las **constantes** **no** pueden cambiar, aunque dependen del tipo de dato. Los tipos compuestos, como acceden a los valores por referencia, si los podemos modificar aunque sean constantes.

Palabras reservadas

```
var nombre;
```

```
nombre = 'Juan';
```

```
let contador;
```

```
contador = 20;
```

```
var unidad = 'metro';
```

```
let altura = 10;
```

```
const gravedad = 9.8;
```

Variables en Javascript

Ámbito global
(**scope**)
(hoisting)

Ámbito local,
o de bloque

```
var nombre;
```

```
nombre = 'Juan';
```

```
let contador;
```

```
contador = 20;
```

```
var unidad = 'metro';
```

```
let altura = 10;
```

```
const gravedad = 9.8;
```

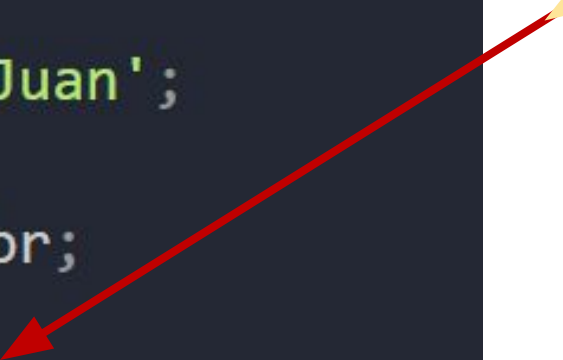
Pueden cambiar

NO puede cambiar

Variables y constantes

```
var nombre;  
nombre = 'Juan';  
  
let contador;  
contador = 20;  
  
var unidad = 'metro';  
  
let altura = 10;  
  
const gravedad = 9.8;
```

Operador de
asignación



Ámbito (scope)

Cuando declaras una variable fuera de una función, esta se llama **variable global**, porque esta disponible para cualquier otro código en el documento actual (esto ocurre también si quitamos la palabra `var` porque la vuelve automáticamente global). Cuando declaras una variable dentro de una función, esta es llamada **variable local**, porque está disponible solo dentro de esa función.

Antes de ECMAScript 6 Javascript no existe el ámbito de sentencias de bloque (`if`, `for`,...); más bien, una variable declarada dentro de un bloque es local para la función en la que reside el bloque. Este comportamiento cambia, cuando usamos la declaración **let** introducida en ECMAScript 2015.

Otra cosa inusual acerca de las variables en JavaScript es que pueden hacer referencia a una variable declarada más tarde, sin dar error (excepción). Sin embargo, las variables que no se han inicializado todavía devolverán un valor *undefined*.

Elevación (hoisting)

En JavaScript, las declaraciones (por ejemplo, de variables o funciones) se mueven al principio de su **scope** o **ámbito**. Este comportamiento se conoce como hoisting y es muy importante tenerlo en cuenta a la hora de programar para prevenir posibles errores.

Teniendo en cuenta cómo funciona el **hoisting**, podemos llamar a una función y definirla más abajo, porque automáticamente JS la “elevará o subirá” a un ámbito superior.

En el caso de las variables, es muy importante tener en cuenta que el hoisting solo se aplica a la declaración, y no a su asignación.

Resumiendo:

- Las funciones siempre se mueven arriba del scope. Por lo tanto, podemos elegir donde declararlas y usarlas.
- La declaración de las variables se mueven arriba del scope, pero no la asignación. Antes de usar una variable, habrá que crearla y asignarla.
- Por buenas prácticas, conviene declarar al principio tanto unas como otras.

Fundamentos de Programación con JavaScript



JS