Variables

La mayoría del tiempo, una aplicación de JavaScript necesita trabajar con información. Aquí hay 2 ejemplos:

1. Una tienda en línea – La información puede incluir los bienes a la venta y un "carrito de compras".
2. Una aplicación de chat – La información puede incluir los usuarios, mensajes, y mucho más.

Utilizamos las variables para almacenar esta información.

Una variable

Una variable es un "almacén con un nombre" para guardar datos. Podemos usar variables para almacenar golosinas, visitantes, y otros datos.

Para generar una variable en JavaScript, se usa la palabra clave `let` .

La siguiente declaración genera (en otras palabras: *declara* o *define*) una variable con el nombre "message":

```
let message;
```

Ahora podemos introducir datos en ella al utilizar el operador de asignación = :

```
let message;

message = 'Hola'; // almacenar la cadena 'Hola' en la variable llamada message
```

La cadena ahora está almacenada en el área de la memoria asociada con la variable. La podemos acceder utilizando el nombre de la variable:

```
let message;
message = 'Hola!';

alert(message); // muestra el contenido de la variable
```

Para ser concisos, podemos combinar la declaración de la variable y su asignación en una sola línea:

```
let message = 'Hola!'; // define la variable y asigna un valor

alert(message); // Hola!
```

También podemos declarar variables múltiples en una sola línea:

```
let user = 'John', age = 25, message = 'Hola';
```

Esto puede parecer más corto, pero no lo recomendamos. Por el bien de la legibilidad, por favor utiliza una línea por variable.

La versión de líneas múltiples es un poco más larga, pero se lee más fácil:

```
let user = 'John';
let age = 25;
let message = 'Hola';
```

Algunas personas también definen variables múltiples en estilo multilínea:

```
let user = 'John',
  age = 25,
  message = 'Hola';
```

...Incluso en este estilo "coma primero":

```
let user = 'John'
  , age = 25
  , message = 'Hola';
```

Técnicamente, todas estas variantes hacen lo mismo. Así que, es cuestión de gusto personal y preferencia estética.

`var` en vez de `let`

En scripts más viejos, a veces se encuentra otra palabra clave: `var` en lugar de `let`:

```
var mensaje = 'Hola';
```

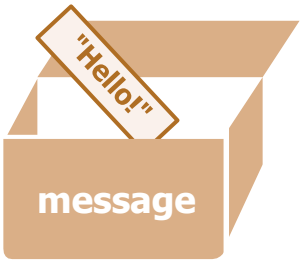
La palabra clave `var` es *casi* lo mismo que `let`. También hace la declaración de una variable, aunque de un modo ligeramente distinto, y más antiguo.

Existen sutiles diferencias entre `let` y `var`, pero no nos interesan en este momento. Cubriremos el tema a detalle en el capítulo La vieja "var".

Una analogía de la vida real

Podemos comprender fácilmente el concepto de una "variable" si nos la imaginamos como una "caja" con una etiqueta de nombre único pegada en ella.

Por ejemplo, podemos imaginar la variable `message` como una caja etiquetada `"message"` con el valor `"Hola!"` adentro:



Podemos introducir cualquier valor a la caja.

También la podemos cambiar cuantas veces queramos:

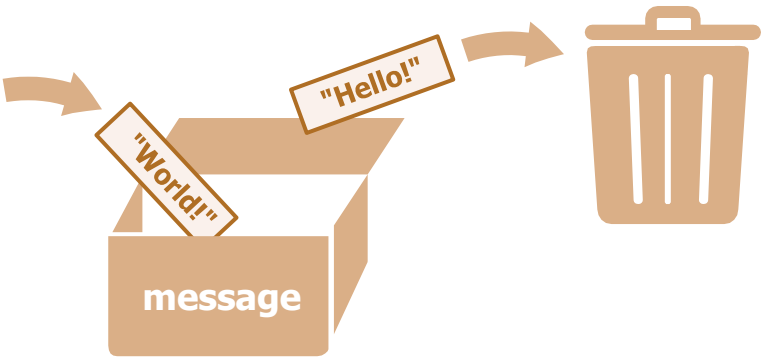
```
let message;

message = 'Hola!';

message = 'Mundo!'; // valor alterado

alert(message);
```

Cuando el valor ha sido alterado, los datos antiguos serán removidos de la variable:



También podemos declarar dos variables y copiar datos de una a la otra.

```
let hello = 'Hola mundo!';

let message;

// copia 'Hola mundo' de hello a message
message = hello;

```

```
// Ahora, ambas variables contienen los mismos datos
alert(hello); // Hola mundo!
alert(message); // Hola mundo!
```

Declarar dos veces lanza un error

Una variable debe ser declarada solamente una vez.

Una declaración repetida de la misma variable es un error:

```
let message = "This";

// 'let' repetidos lleva a un error
let message = "That"; // SyntaxError: 'message' ya fue declarado
```

Debemos declarar una variable una sola vez y desde entonces referirnos a ella sin `let`.

Lenguajes funcionales

Es interesante notar el hecho que lenguajes de programación funcional, como Scala o Erlang prohíben cambiar el valor de variables.

En tales lenguajes, una vez la variable ha sido almacenada "en la caja", permanece allí por siempre. Si necesitamos almacenar algo más, el lenguaje nos obliga a crear una nueva caja (generar una nueva variable). No podemos reusar la antigua.

Aunque puede parecer un poco extraño a primera vista, estos lenguajes son muy capaces de desarrollo serio. Más aún, existen áreas como computación en paralelo en las cuales esta limitación otorga ciertos beneficios. Estudiar tales lenguajes (incluso sin la intención de usarlo en el futuro cercano) es recomendable para ampliar la mente.

Nombramiento de variables

Existen dos limitaciones de nombre de variables en JavaScript:

1. El nombre únicamente puede incluir letras, dígitos, o los símbolos `$` y `_`.
2. El primer carácter no puede ser un dígito.

Ejemplos de nombres válidos:

```
let userName;
let test123;
```

Cuando el nombre contiene varias palabras, comúnmente se utiliza camelCase. Es decir: palabras van una detrás de otra, con cada palabra iniciando con letra mayúscula: `miNombreMuyLargo`.

Es interesante notar – el símbolo del dólar `'$'` y el guión bajo `'_'` también se utilizan en nombres. Son símbolos comunes, tal como las letras, sin ningún significado especial.

Los siguientes nombres son válidos:

```
let $ = 1; // Declara una variable con el nombre "$"
let _ = 2; // y ahora una variable con el nombre "_"

alert($ + _); // 3
```

Ejemplos de nombres incorrectos:

```
let 1a; // no puede iniciar con un dígito

let my-name; // los guiones '-' no son permitidos en nombres
```

La Capitalización es Importante

Dos variables con nombres `manzana` y `MANZANA` son variables distintas.

Letras que no son del alfabeto inglés están permitidas, pero no se recomiendan

Es posible utilizar letras de cualquier alfabeto, incluyendo el cirílico e incluso jeroglíficos, por ejemplo:

```
let имя = '...';
let 我 = '...';
```

Técnicamente, no existe ningún error aquí. Tales nombres están permitidos, pero existe una tradición internacional de utilizar inglés en el nombramiento de variables. Incluso si estamos escribiendo un script pequeño, este puede tener una larga vida por delante. Puede ser necesario que gente de otros países deba leerlo en algún momento.

Nombres reservados

Hay una lista de palabras reservadas, las cuales no pueden ser utilizadas como nombre de variable porque el lenguaje en sí las utiliza.

Por ejemplo: `let`, `class`, `return`, y `function` están reservadas.

El siguiente código nos da un error de sintaxis:

```
let let = 5; // no se puede le nombrar "let" a una variable  ¡Error!
let return = 5; // tampoco se le puede nombrar "return", ¡Error!
```

Una asignación sin utilizar `use strict`

Normalmente, debemos definir una variable antes de utilizarla. Pero, en los viejos tiempos, era técnicamente posible crear una variable simplemente asignando un valor sin utilizar 'let'. Esto aún funciona si no ponemos 'use strict' en nuestros scripts para mantener la compatibilidad con scripts antiguos.

```
// nota: no se utiliza "use strict" en este ejemplo

num = 5; // se crea la variable "num" si no existe antes

alert(num); // 5
```

Esto es una mala práctica que causaría errores en 'strict mode':

```
"use strict";

num = 5; // error: num no está definida
```

Constantes

Para declarar una variable constante (inmutable) use `const` en vez de `let`:

```
const myBirthday = '18.04.1982';
```

Las variables declaradas utilizando `const` se llaman "constantes". No pueden ser alteradas. Al intentarlo causaría un error:

```
const myBirthday = '18.04.1982';

myBirthday = '01.01.2001'; // ¡error, no se puede reasignar la constante!
```

Cuando un programador está seguro de que una variable nunca cambiará, puede declarar la variable con `const` para garantizar y comunicar claramente este hecho a todos.

Constantes mayúsculas

Existe una práctica utilizada ampliamente de utilizar constantes como aliases de valores difíciles-de-recordar y que se conocen previo a la ejecución.

Tales constantes se nombran utilizando letras mayúsculas y guiones bajos.

Por ejemplo, creemos constantes para los colores en el formato "web" (hexadecimal):

```
const COLOR_RED = "#F00";
const COLOR_GREEN = "#0F0";
const COLOR_BLUE = "#00F";
const COLOR_ORANGE = "#FF7F00";

// ...cuando debemos elegir un color
let color = COLOR_ORANGE;
alert(color); // #FF7F00
```

Ventajas:

- `COLOR_ORANGE` es mucho más fácil de recordar que `"#FF7F00"`.
- Es mucho más fácil escribir mal `"#FF7F00"` que `COLOR_ORANGE`.
- Al leer el código, `COLOR_ORANGE` tiene mucho más significado que `#FF7F00`.

¿Cuándo se deben utilizar letras mayúsculas para una constante, y cuando se debe nombrarla de manera normal? Dejémoslo claro.

Ser una "constante" solo significa que el valor de la variable nunca cambia. Pero hay constantes que son conocidas previo a la ejecución (como el valor hexadecimal del color rojo) y hay constantes que son *calculadas* en el tiempo de ejecución, pero no cambian después de su asignación inicial.

Por ejemplo:

```
const pageLoadTime = /* el tiempo que tardó la página web para cargar */;
```

El valor de `pageLoadTime` no se conoce antes de cargar la página, así que la nombramos normalmente. No obstante, es una constante porque no cambia después de su asignación inicial.

En otras palabras, las constantes con nombres en mayúscula son utilizadas solamente como alias para valores invariables y preestablecidos ("hard-coded").

Nombrar cosas correctamente

Estando en el tema de las variables, existe una cosa de mucha importancia.

Una variable debe tener un nombre claro, de significado evidente, que describa el dato que almacena.

Nombrar variables es una de las habilidades más importantes y complejas en la programación. Un vistazo rápido a el nombre de las variables nos revela cuál código fue escrito por un principiante o por un desarrollador experimentado.

En un proyecto real, la mayor parte de el tiempo se pasa modificando y extendiendo una base de código en vez de empezar a escribir algo desde cero. Cuando regresamos a algún código después de hacer algo distinto por un rato, es mucho más fácil encontrar información que está bien etiquetada. O, en otras palabras, cuando las variables tienen nombres adecuados.

Por favor pasa tiempo pensando en el nombre adecuado para una variable antes de declararla. Hacer esto te da un retorno muy sustancial.

Algunas reglas buenas para seguir:

- Use términos legibles para humanos como `userName` p `shoppingCart` .
- Evite abreviaciones o nombres cortos `a` , `b` , `c` , al menos que en serio sepa lo que está haciendo.
- Cree nombres que describen al máximo lo que son y sean concisos. Ejemplos que no son adecuados son `data` y `value` . Estos nombres no nos dicen nada. Estos solo está bien usarlos en el contexto de un código que deje excepcionalmente obvio cuál valor o cuales datos está referenciando la variable.
- Acuerda en tu propia mente y con tu equipo cuáles términos se utilizarán. Si a un visitante se le llamara "user", debemos llamar las variables relacionadas `currentUser` o `newUser` en vez de `currentVisitor` o `newManInTown` .

¿Suena simple? De hecho lo es, pero no es tan fácil crear nombres de variables descriptivos y concisos a la hora de practicar. Inténtelo.

> **¿Reusar o crear?**
>
> Una última nota. Existen programadores haraganes que, en vez de declarar una variable nueva, tienden a reusar las existentes.
>
> El resultado de esto es que sus variables son como cajas en las cuales la gente introduce cosas distintas sin cambiar sus etiquetas. ¿Que existe dentro de la caja? ¿Quién sabe? Necesitamos acercarnos y revisar.
>
> Dichos programadores se ahorran un poco durante la declaración de la variable, pero pierden diez veces más a la hora de depuración.
>
> Una variable extra es algo bueno, no algo malvado.
>
> Los minificadores de JavaScript moderno, y los navegadores optimizan el código suficientemente bien para no generar cuestiones de rendimiento. Utilizar diferentes variables para distintos valores incluso puede ayudar a optimizar su código

Resumen

Podemos declarar variables para almacenar datos al utilizar las palabra clave `var` , `let` , o `const` .

- `let` – es la forma moderna de declaración de una variable.
- `var` – es la declaración de variable de vieja escuela. Normalmente no lo utilizamos en absoluto. Cubriremos sus sutiles diferencias con `let` en el capítulo La vieja "var", por si lo necesitaras.
- `const` – es como `let` , pero el valor de la variable no puede ser alterado.

Las variables deben ser nombradas de tal manera que entendamos fácilmente lo que está en su interior.