educación IT
CODE YOUR FUTURE

JavaScript desde cero

Módulo 1

Variables

Introducción

Qué es una variable

Una *variable* es un elemento que permite almacenar o guardar información para ser utilizada nuevamente. Cuando se incorporan variables en una aplicación web, y esta se ejecuta, reserva un espacio en la memoria de la computadora o dispositivo, donde deja disponible la variable con el valor que se le haya asignado.

Tanto en JavaScript como en cualquier otro lenguaje de programación, **las variables son una parte importante de nuestras aplicaciones web, dado que nos permitirán almacenar, recuperar y compartir información que se deba utilizar en diferentes partes de nuestro programa**.

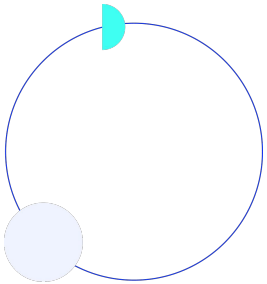
```
let numero = 21
```

Variación del valor almacenado

Una variable, tal como su nombre lo indica, **puede "variar" el valor almacenado en ella, en cualquier momento** que se decida o que la aplicación web lo requiera, a través de su lógica.

JavaScript, en particular, es un lenguaje de programación de tipado débil, lo cual indica que una variable puede contener un dato determinado, y luego cambiarlo por otro tipo de dato y valor completamente diferente. Aún así, esto último **no es algo que suceda a menudo**.

```
let numero = 21
let empresa = "EducaciónIT"
let usuarioIdentificado = true
```



Declaración de las variables

Las variables se declaran **anteponiendo la palabra reservada `let`, seguido del nombre "identificativo"** que le daremos a la variable. Este nombre debe **identificar su valor almacenado**.

Para asignar un **valor**, en el mismo momento en que la declaramos, se agrega el **operador de asignación igual `=` y luego el valor deseado**.

```
let nombreCompleto            //variable declarada

let empresa = "EducaciónIT".  //variable declarada con un valor asignado
```

En JavaScript, **se pueden declarar variables con las palabras clave var** y **`let`**, según el alcance que se desea definir para esa variable.

A partir de la versión ***ES6* (año 2015)**, se insiste en priorizar el uso de **`let`** para declarar variables, ya que ofrece un alcance de bloque más claro y menos propenso a errores.

Si tu propósito futuro es trabajar en el universo del desarrollo web *frontend* o *backend* con el lenguaje JavaScript, **enfócate en declarar las variables, siempre, mediante `let`**.

> Si deseas conocer qué es *Ecmascript (ES)*, puedes visitar este artículo donde se explica bien su significado.

```
var nombreCompleto = "Joe McMillian"   ❌
var empresa = "EducaciónIT"            ⚠️
let totalDelCarrito = 2599.22
let usuarioLogueado = TRUE
```

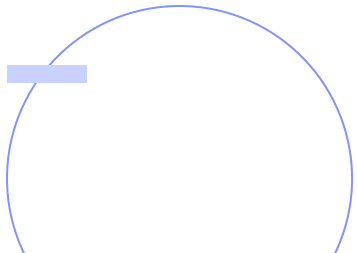
Si declaramos una variable con un valor inicial, y luego se necesita **cambiar el valor por otro**, simplemente se debe **definir el nombre de la variable seguida del signo = y el nuevo valor** que deseamos almacenar en ella.

Con esto, la variable cambió su valor inicial, y se podrá utilizar su nuevo valor, en cualquier parte del programa donde se requiera.

```
let nombreCompleto = "Joe McMillian"

// otras tantas líneas de código de tu programa


 // aquí cambiamos el valor de la variable
nombreCompleto = "Donna Clark"
```



Reglas de nomenclatura

Reglas para nombrar una variable en JavaScript:

- Los nombres de variables pueden contener **letras**, **números**, **_** (***underscore***) y **signo de dólar** (**$**).
- **No** pueden comenzar con un número.
- Deben **comenzar en minúsculas**.
- Los **nombres** son **case sensitive** (no es lo mismo `A` que `a`).

Si bien existe flexibilidad, en la actualidad, se recomienda crear variables con el **formato *camelCase***. Este formato representa una estructura fiel a la que adopta JavaScript como lenguaje de programación. Veamos un ejemplo en el siguiente slide.

```
//variable de palabra única
let numero

//variable con palabras compuestas, utilizando la nomenclatura snake_case
let nombre_empresa

//variable con palabras compuestas, utilizando la nomenclatura camelCase
let usuarioIdentificado = true
```

En JavaScript se estila **declarar cada variable o constante utilizando una palabra, en minúsculas**, que describa el tipo de datos que almacena.

Es posible **usar dos o más palabras**, en aquellos casos donde una sola no alcance para describir el valor que se almacenará.

Cuando se combinen palabras usando el formato ***camelCase*** se debe contemplar que, **cada palabra adicional tenga su primera inicial en mayúsculas**.

```
let nombre = "EducaciónIT"
let empresa = "EDUCACIONIT"

let nroA = 2005
let nroB = 1996
```

Evitar la declaración de variables que incluyan:

- **Verbos** en su nombre.
- Uso de letras con **tilde**, **diéresis**, o **ñuflos** (letra *ñ*).
- **Primera inicial en mayúsculas** (se reserva para la creación de objetos).

Constantes

Definición

Qué son las constantes

Las *constantes*, son una **especie de variable** que también ocupan un espacio en memoria, y permiten **almacenar un valor específico para ser utilizado dentro de nuestra aplicación web.**

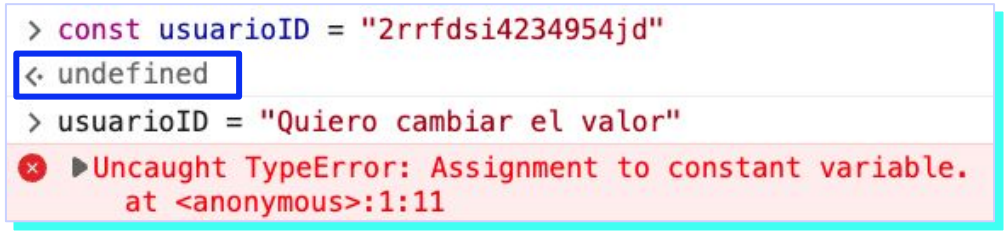
Se declaran al anteponer la palabra reservada **`const`**.

```
const usuarioID = "2rrfdsi4234954jd"

const empresa = "EducaciónIT"

const metodoDePago = "Tarjeta de débito"
```

A diferencia de una variable declarada con `let`, **las constantes no permiten cambiar el valor que se le haya asignado**. Tal como su nombre lo indica, el **valor inicial** que le asignemos será *"un **valor constante**"*, durante todo el ciclo de vida de nuestra aplicación. Si intentamos cambiar su valor predefinido, la constante arrojará un error y no realizará el cambio forzado.



¿Cuándo utilizarlas?

Las constantes se deben utilizar **en escenarios donde sea necesario guardar un valor y este no deba cambiar**.

> **Por ejemplo:** el ***ID*** asignado a un usuario es generalmente unívoco, o el ***Email*** del usuario identificado. También su ***nombre de usuario***.
>
> En estos escenarios y en otros que veremos más adelante, las constantes deben implementarse de forma efectiva. A medida que se vayan incorporando más conocimientos del lenguaje se irá comprendiendo mejor en qué escenarios deben entrar en acción.

Tipos de datos

Tipos de datos

Se define como ***tipado de datos***, a la **forma de poder crear variables que respeten una estructura específica de la información que guarden**.

JavaScript posee lo que se denomina un tipado débil de datos, por lo tanto no es necesario definir qué tipo de datos posee una variable y **podrá cambiar durante el ciclo de vida de la aplicación web**.

Los tipos de datos comúnmente manejados por JavaScript, son los que se muestran en la tabla de la próxima pantalla.

```
let empresa = "EducaciónIT"

let anioNacimiento = 2003

let usuarioLogueado = TRUE
```

Tipo de dato	Descripción
`string`	Almacena datos del tipo cadena de caracteres. Puede ser una cadena alfabética o alfanumérica, incluyendo caracteres extendidos.
`number`	Almacena un dato numérico. El mismo puede ser entero o de punto flotante (poseer decimales). Para este último punto, JavaScript maneja los decimales utilizando el punto (`.`) como separador decimal.
`boolean`	Un tipo de dato estándar para definir valores verdaderos (**`TRUE`**), o valores falsos (**`FALSE`**).
`object`	Un tipo de dato propio de los objetos en JS. Se utiliza tanto para los objetos literales como también para objetos del sistema, instanciados a partir de una clase, y también para un array de objetos.
`array`	Un tipo de dato propio de los *array* o colecciones de elementos o de objetos. En JS es muy utilizado y es la base fundamental de conceptos avanzados en este lenguaje de programación.
`NaN`	La propiedad global **`NaN`** es un valor que representa ***Not-A-Number***. Por ejemplo, si declaramos una variable sin valor, y la multiplicamos por un número, su resultado será **`NaN`**.
`undefined`	La propiedad global **`undefined`** representa el valor primitivo undefined. Una variable a la que no se le ha asignado valor es de tipo **`undefined`**.

string

Las variables del tipo **`string`**:

- Permiten almacenar valores también denominados como ***cadenas de texto***.
- Pueden ser ***alfanuméricas***, y contener cualquier tipo de dato.
- Al declararlas, se debe encerrar el **valor asignado entre comillas** (simples `''` o comillas dobles `""`).

Variables

```
let nombre = "Joe"

let apellido = "McMillian"

let empresa = "Alphabet"

let nombreCompleto = "Cameron Howe"
```

number

Las variables del tipo **number**:

- Permiten almacenar **valores numéricos**. Estos pueden ser del tipo **entero**, o con **decimales**.
- En la definición de un valor numérico con decimales, se debe utilizar el punto "**.**" como separador.
- **No se utiliza separador de miles, y se declaran valores sin encerrarlos entre comillas**.

Variables

```
let importe = 1250.21

let precioFinal = 1550.42

let anioActual = 2023
```

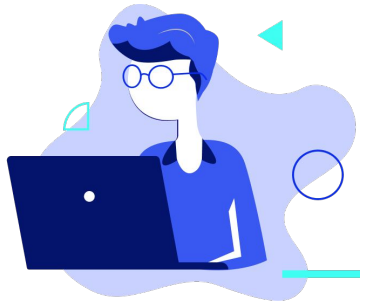
boolean

Las variables del tipo **`boolean`**:

- Permiten almacenar valores *booleanos* (**`TRUE`** o **`FALSE`**). Son los únicos dos valores posibles en un entorno booleano.
- **Estos valores representan un dato, cuando este es verdadero, o es falso**.

```
let usuarioLogueado = true

let passwordAlmacenado = false
```



array

- Un *array* es un tipo de variable que **permite almacenar múltiples valores** en lugar de uno solo. Este tipo de dato facilita la **agrupación de valores que tienen algo en común**.

- De esta forma, se pueden concentrar todos **en un único lugar**. Esto hará que sean mucho más fácil de acceder. Los *arrays* tienen un poder y un valor importante en la construcción cotidiana de aplicaciones JavaScript.

Variables

```
const frutas = ['Pera', 'Banana', 'Manzana', 'Tomate']
```

NaN

Las variables del tipo **NaN**:

- Definen un valor que significa ***Not-A-Number***.

  Esto ocurre, por ejemplo, cuando tenemos una variable declarada pero no inicializada (no se le asignó un valor), e intentamos multiplicarla por un valor numérico.

Variables

```
let nombreDelCurso

let valorNominal

valorNominal * 122 // retornará NaN
```

Propiedades y métodos en variables

Propiedades y métodos en variables

Cuando se crea una variable, esta recibe, de forma predeterminada, una serie de ***métodos***.

Estos métodos, **permiten trabajar sobre el valor de la variable** de acuerdo a la necesidad que debemos aplicar en el programa.

```
let numero = 21
let empresa = "EducaciónIT"
let usuarioIdentificado = true
```

Los métodos disponibles para una variable del tipo **`String`**, son propios de este tipo de dato. **Existen muchísimos métodos**. **Ayudan a interactuar con los datos de la variable**, de forma precisa; **leen, convierten o transforman el valor almacenado** según la necesidad.

Variables

```
let nombre = 'EducaciónIT';

nombre.length //devuelve el total de caracteres
nombre.at(2)  //devuelve el caracter en dicha posición
nombre.trim() //elimina espacios agregados al inicio o final del
nombre.toUpperCase() //convierte el texto a mayúsculas
nombre.toLowerCase() //convierte el texto a minúsculas
```

Los métodos para variables que almacenan valores del tipo **number** ayudan a **trabajar mejor el formato de cada número**, de acuerdo a la necesidad **en el momento de almacenarlos, o de representarlos en pantalla**.

Variables

```
let numero = 2103;

numero.toPrecision(6);    //convierte el número a 6 dígitos, acorta o agrega
numero.toFixed(2);        //fuerza un redondeo decimal a la cantidad numérica
numero.toLocaleString();  //formatea el número a la configuración regional del S.O.
numero.toString();        //convierte el número en texto
```

educación IT
CODE YOUR FUTURE

Visualizar variables en la consola JS

Visualizar variables en la consola JS

La **consola JS** integrada en *Developer Tools* es una herramienta netamente **pensada para depurar aplicaciones**.

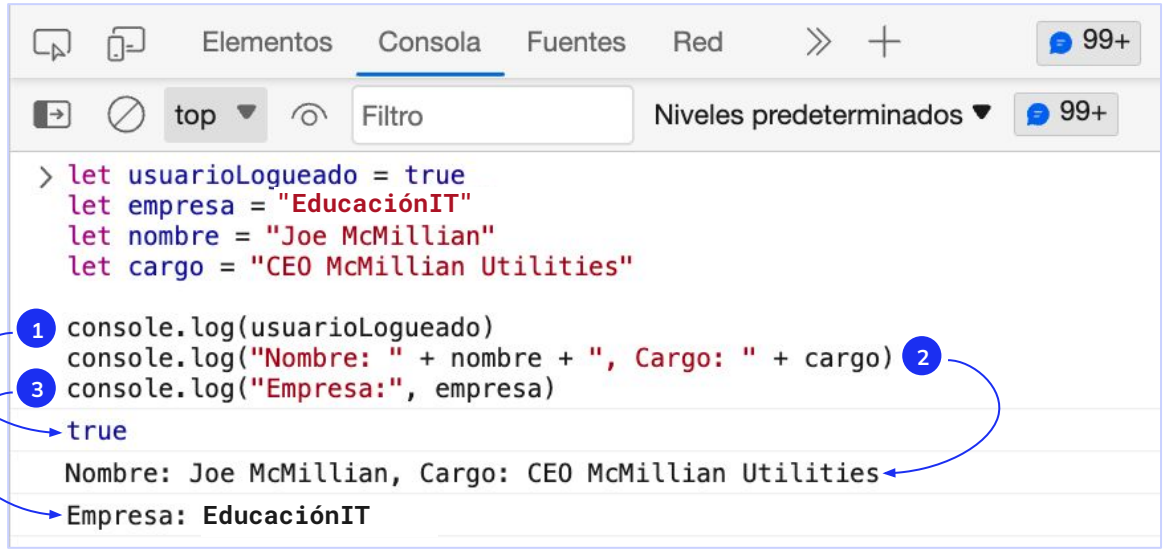
Entre todas las funcionalidades que tiene, permite visualizar el valor de variables y constantes. Para ello, se utiliza el objeto **`console`**, integrado a JS, y su método **`.log()`**.

Dentro de los paréntesis de este método, se referencia la variable y así se puede ver, en la consola JS, qué valor posee almacenado.

```
let usuarioLogueado = true
let empresa = "EducaciónIT"
let nombre = "Joe McMillian"
let cargo = "CEO McMillian Utilities"

console.log(usuarioLogueado)
console.log("Nombre: " + nombre + ", Cargo: " + cargo)
console.log("Empresa:", empresa)
```

Como se observa en el ejemplo, es posible visualizar, en la consola JS, el valor de una variable (***1***), el valor de una variable concatenado (**+**) con texto estático (***2***), o también el valor de una variable concatenada con texto estático utilizando una coma en lugar del signo más (***3***).



Revisión

- Repasar los conceptos básicos de un **lenguaje**
- **de programación**.
- Trabajar con variables en sus **diferentes tipos**.
- Trabajar con la ***consola JS***.
- Aplicar todas las propiedades en el ***Proyecto integrador***.

¡Sigamos trabajando!