

Bootcamp Java Developer

Fase 2 - Java Web Developer
Módulo 17



Variables

Introducción

Las variables en los lenguajes de programación en general, no solamente en JS, tienen la misma lógica que en una operación matemática.

Una variable es un elemento que se emplea para almacenar, guardar la información en un cajón y utilizarla nuevamente (por eso el nombre de *variable*).

Sin variables cualquier programa es inútil y sin sentido, por eso es necesario conocerlas.

Las variables se crean o definen mediante la **palabra reservada var**.



Reglas de nomenclatura

Las **reglas para nombrar una variable** son las siguientes:

- Los nombres de variables pueden contener **letras, números, _ (underscore) y signo de dólar (\$)**.
- **No se puede comenzar con un número.**
- Los nombres son **case sensitive** (esto significa que no es lo mismo A que a).

- Hay **palabras reservadas** que se pueden usar para nombres de variables, a medida que vayamos avanzado aprenderás cuáles son.

A screenshot of a code editor with two tabs: 'index.html' and 'codigo.js'. The 'codigo.js' tab is active and shows the code 'var primerVariable;' in a dark theme with syntax highlighting. The word 'var' is in light blue, 'primer' is in light blue, and 'Variable;' is in light blue. The editor has a light blue border on the right side.

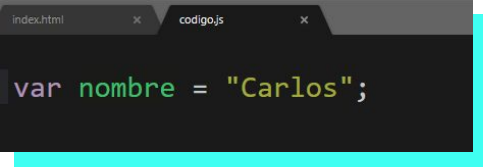
```
index.html x codigo.js x
var primerVariable;
```

Variables tipo String

String son fundamentalmente **cadenas de texto**.

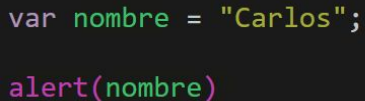
Se escriben entre comillas dobles o comillas simples. Vamos a probarlo en nuestro **codigo.js**, por ejemplo, en la imagen de arriba a la derecha:

Por supuesto que nada de esto tiene sentido si yo no puedo mostrar la información, por lo tanto, vamos a mostrar el nombre del empleado a través de una ventana de alerta (segunda imagen). En la línea de la alerta, no es necesario, como en el primer ejemplo, poner **nombre** entre comillas ya que la variable es un objeto.

A screenshot of a code editor with two tabs: 'index.html' and 'codigo.js'. The 'codigo.js' tab is active and shows the code:

```
var nombre = "Carlos";
```

```
var nombre = "Carlos";
```

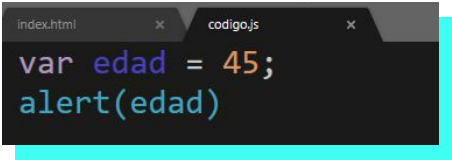
A screenshot of a code editor showing the same code as the previous image, but with an additional line:

```
alert(nombre)
```

```
var nombre = "Carlos";  
alert(nombre)
```

Variables tipo numéricas

Comprenden cualquier expresión numérica que puede ser **float (decimal)** o **entera**. En nuestro archivo, trabajaremos de la siguiente manera:

A screenshot of a code editor with two tabs: 'index.html' and 'codigo.js'. The 'codigo.js' tab is active and contains the following code:

```
var edad = 45;  
alert(edad)
```

`var edad = 45;
alert(edad)`

Si queremos **distinguir decimales de enteros** lo haremos de la siguiente forma:

A screenshot of a code editor with two tabs: 'index.html' and 'codigo.js'. The 'codigo.js' tab is active and contains the following code:

```
var iva = 21; // variable tipo entero  
var total = 9234.65; // variable tipo decimal
```

`var iva = 21; // variable tipo entero
var total = 9234.65; // variable tipo decimal`

También existen otras variables que más adelante profundizaremos tales como:

- **Boolean:** expresiones booleanas **TRUE** o **FALSE**.
- **Undefined:** toda variable declarada sin valor o cualquier propiedad interna no existente de un **objeto**.
- **Object:** vector asociativo en n dimensiones. El mismo se inicializa de manera literal con **{ }** (**llaves**) y contiene en su interior pares de índices asociados a valores por el operador **:** (**dos puntos**) separados por **,** (**coma**).
- **Array :** objeto especializado en poder tener, además de su comportamiento habitual, la habilidad de guardar datos de manera secuencial, es decir, bajo **índices numéricos** auto incrementales.

Ej.: `{nombre:"Educacion IT"}`



Output de JS: document.write()

Hay variadas formas de **mostrar nuestra información en JS**, por ejemplo, vimos cómo se puede hacer a través de una ventana de alerta.

Sin embargo, existen otras maneras de interactuar con nuestro **HTML**, por ejemplo con **document.write()**, que permite escribir directamente en el documento.

Para lograr tal fin, generamos las siguientes líneas de código en el **archivo.js**:

```
document.write('Hola estoy dentro de mi documento')
```

El resultado será el siguiente:

Hola estoy dentro de mi documento

Para generar un mejor formato, se **implementan etiquetas de HTML**. Por ejemplo, la intención es que este **texto sea un título**, lo haremos de la siguiente forma:

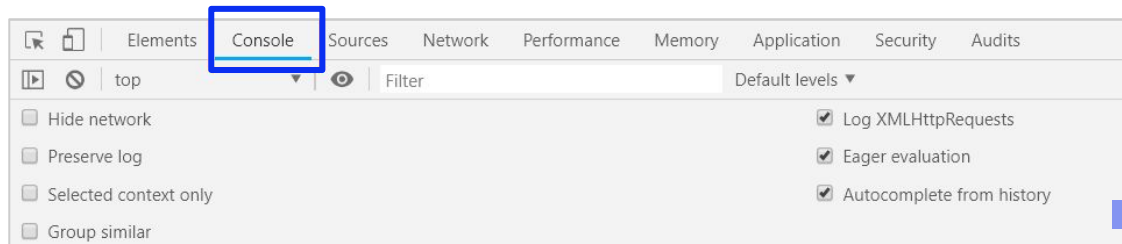
```
document.write('<h1>Hola estoy dentro de mi documento</h1>')
```

Podemos también implementar varias líneas dentro de nuestro archivo con diferentes etiquetas válidas de HTML:

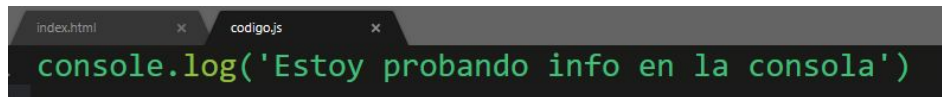
```
document.write('<h1>Hola estoy dentro de mi documento</h1>')  
document.write('<p>Este es un párrafo con<strong>negrita</strong></p>')
```

Output de JS: console.log()

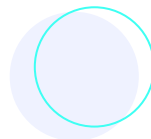
Esta forma de **generar contenido** se hace a través **de la consola**. Se accede presionando la **tecla f12** desde el navegador:



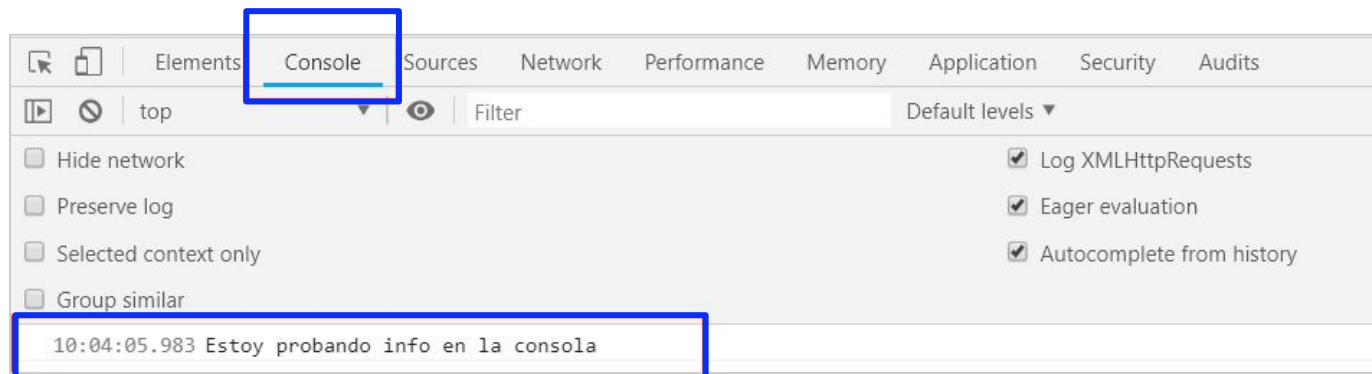
La **consola** permite conocer errores, datos y demás cuestiones importantes para el **trabajo con JS**, pero puntualmente en este caso la utilizaremos para generar información de la siguiente manera:

A screenshot of a code editor interface. At the top, there are two tabs: 'index.html' and 'codigo.js'. The 'codigo.js' tab is active. Below the tabs, a line of JavaScript code is displayed: `console.log('Estoy probando info en la consola')`. The code is written in a green monospace font on a dark background. The entire editor area is highlighted with a cyan border.

```
index.html x codigo.js x
console.log('Estoy probando info en la consola')
```



El resultado del slide anterior será el siguiente:



Revisión

- Repasar los conceptos básicos de un **lenguaje de programación**.
- Trabajar con variables en sus **diferentes tipos**.
- Implementar una **ventana de alerta de forma externa a través de un string**.
- Mostrar datos a través de **document.write()**
- Trabajar con **console.log()**
- Aplicar todas las propiedades en el **proyecto integrador**.



**¡Sigamos
trabajando!**