Curso HTML5, CSS y JavaScript

Módulo 5: Introducción a JavaScript

- emilianoagustingallo@gmail.com
- https://www.linkedin.com/in/emiliano-gallo/

Capacitarte

Variables en JS

Son contenedores para almacenar datos o valores

En el siguiente ejemplo: x, y y z son variables

```
var x = 5;
var y = 6;
var z = x + y;
```

Usando let y var

La principal diferencia son las reglas de alcance.

Las variables declaradas por la palabra clave **var** tienen como alcance el cuerpo de la función inmediata. Una variable definida mediante una instrucción **var** se conoce en toda la función en la que se define, desde el inicio de la función.

Una variable definida mediante una sentencia **let** solo se conoce en el bloque en el que está definida, desde el momento en que se define en adelante. Tienen el alcance del bloque envolvente inmediato denotado por {}.

Tipos de datos

Las variables de JS pueden contener muchos tipos de datos: **números**, **texto**, **objetos**, etc.

En programación, los tipos de datos son un concepto importante.

```
var length = 16;
var lastName = "Johnson";
var x = {firstName:"John", lastName:"Doe"};
```

Operando con data types

Para poder operar con variables, es importante saber respecto al **tipo**.

En los 2 siguientes ejemplos, la variable **x** va a tener el valor: “16Volvo”

```
var x = 16 + "Volvo";
```

```
var x = "16" + "Volvo";
```

Esto se debe a que al agregar un **string** y un **número**, JS lo va a tratar todo como un **string**

Operando con data types

JS evalúa expresiones de izquierda a derecha. Diferentes secuencias pueden producir diferentes resultados:

En el siguiente ejemplo, la variable **x** va a tener el valor: “20Volvo”

```
var x = 16 + 4 + "Volvo";
```

En el siguiente ejemplo, la variable **x** va a tener el valor: “Volvo164”

```
var x = "Volvo" + 16 + 4;
```

Tipado dinámico

JS tiene tipos dinámicos. Esto significa que la misma variable se puede utilizar para contener diferentes tipos de datos:

```
var x;           // Ahora x es undefined

x = 5;           // Ahora x es un número

x = "John";      // Ahora x es un texto
```

Cadenas de texto en JS

Una cadena (o una cadena de texto) es una serie de caracteres como "**Esto es un texto**".

Las cadenas se escriben con comillas. Puede utilizar comillas **simples** o **dobles**:

```
// Texto dentro de comillas dobles
var text1 = "Hola! Bienvenidos al curso de JS";

// Texto con comillas simples dentro de comillas dobles
var text2 = "Hola! Bienvenidos al curso de 'JS'";

// Texto con comillas dobles dentro de comillas simples
var text3 = 'Hola! Bienvenidos al curso de "JS"';
```

Números en JS

JS solamente tiene un tipo de dato del tipo numérico. Y este puede ser representado con o sin parte decimal.

```
var x1 = 34.00;     // Con parte decimal

var x2 = 34;        // Sin parte decimal
```

Booleano en JS

Un variable del tipo **boolean** solamente puede tener 2 valores posibles: **true** o **false**

```
var x = 5;
var y = 5;
var z = 6;
(x == y)       // Devuelve true
(x == z)       // Devuelve false
```

Matrices en JS

Las matrices de **JS** se escriben entre corchetes.

Los elementos de la matriz están separados por comas.

El siguiente código declara una matriz llamada **cars**, que contiene tres elementos:

```
var cars = ["Fiat", "Ford", "BMW"];
```

Los índices de matriz están basados en **cero**, lo que significa que el primer elemento es **[0]**, el segundo es **[1]** y así sucesivamente.

Objetos, clases o JSON en JS

Los objetos de **JS** se escriben con llaves **{}**.

Las propiedades de los objetos se escriben como pares **nombre**: **valor**, separados por **comas**.

En el siguiente ejemplo, el objeto **person** 4 propiedades: **firstName**, **lastName**, **age** y **eyeColor**.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Null

En JS, **null** es "nada". Se supone que es algo que no existe.

En JS, el tipo de datos null es un objeto.

Puede considerar que es un error en JavaScript que typeof null sea un objeto. Debería ser nulo.

Puede vaciar un objeto configurándolo en null:

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
person = null;    // Ahora el valor es null, pero sigue siendo un object
```

Algunos ejemplos