<codoacodo/>

Agencia de
Aprendizaje
a lo largo
de la vida

Desarrollo Fullstack

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 11

CSS Avanzado

- Diseño Responsive
- Viewport
- Mobile First
- Media Queries
- Breakpoint

Clase 12

Javascript

- ¿Qué es Javascript?
- Variables
- Tipos de datos
- Prompt, alert y console

Clase 13

Javascript

- Operadores Relacionales
- Operadores Lógicos
- Condicionales
- Bucles

JAVASCRIPT

El dinamismo de la web

¿Qué es Javascript?

Es un lenguaje de programación creado en el año 1995 y basado en el estándar **ECMAScript** quien determina cómo los navegadores deben **interpretar este lenguaje**.

A **Javascript** se le denomina *lenguaje "del lado del cliente"* porque se ejecuta en contexto del navegador (cliente web) a diferencia de otros lenguajes que **corren en el servidor**.

¿Cómo funciona Javascript?

Javascript es un lenguaje orientado a prototipos, multiparadigma e interpretado.

Esto quiere decir que **podemos usar** programación funcional, **orientada a objetos** o imperativa.

Además, al ser un lenguaje interpretado nuestro código será leído y procesado en tiempo de ejecución a diferencia de los **lenguajes compilados** que leen todo el código antes de comenzar a correr el programa.

¿Para qué se usa?

JavaScript en el navegador puede hacer todo lo relacionado con la manipulación de la página web, la interacción con el usuario y el servidor.

- **Cambiar** todo el contenido de una página web (tipo de letra, colores, animaciones, etc.)
- Enviar información a través de la red a servidores remotos, descargar archivos.
- Almacenamiento local en el navegador (recuperar, almacenar información durante la ejecución y visualización de la página web).

¿Para qué NO se usa?

JavaScript **NO** puede acceder a las circuitos integrados de una computadora tales como:

- Disco Duro (Acceso a eliminar información, modificar o leer).
- Acceso a la memoria **RAM, ROM**.
- Acceso a la tarjeta de RED o Procesadores.
- **Trabajar del lado del servidor**.

El objetivo de JavaScript en el navegador solo **se limita** al uso exclusivo de todo lo que una página web te puede brindar.

Vincular nuestro Javascript

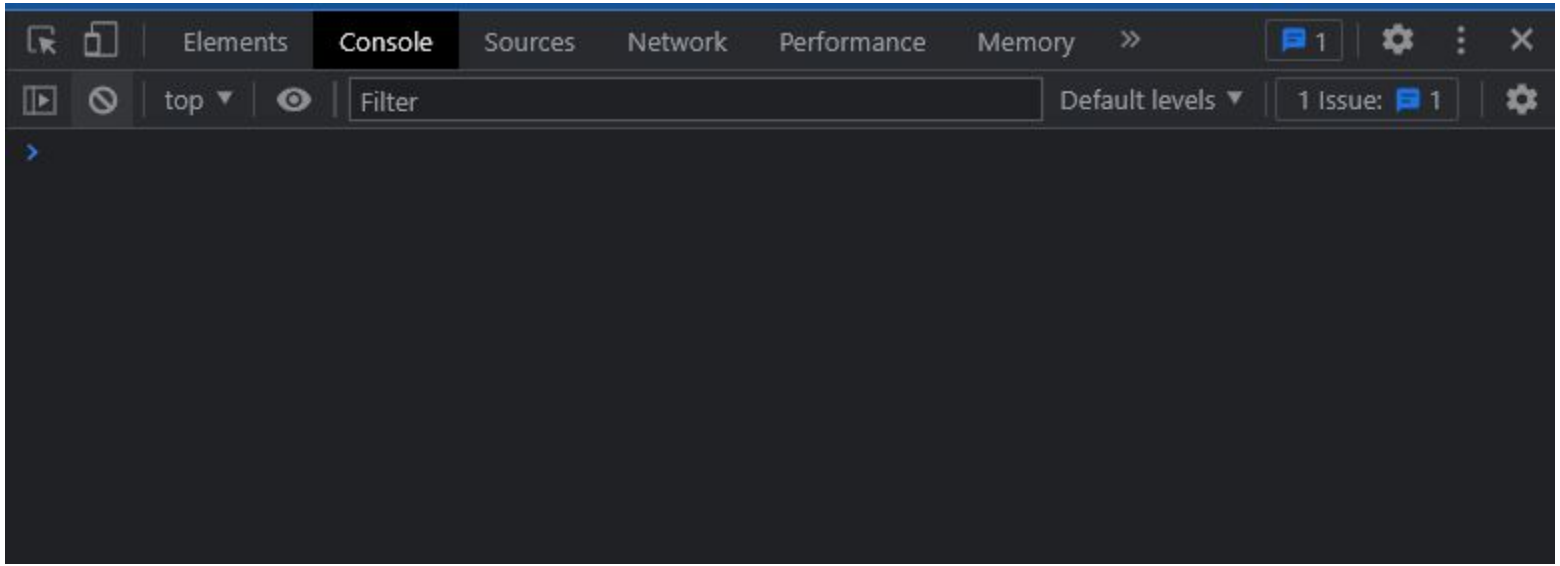
Hay dos formas de incluir nuestro código JavaScript en nuestro documento HTML:

Interna: dentro de una etiqueta **`<script></script>`** antes del cierre del **`</body>`** en nuestro html.

Externa: en un archivo con extensión .js vinculado a través de una etiqueta **`<script src="index.js"></script>`** usando el **atributo source** para indicar la ruta al archivo.

¿Cómo vemos los resultados de nuestro código?

Para esto usaremos la consola del navegador a la cual podemos acceder presionando **f12**, ctrl + Mayús + J o simplemente haciendo click derecho y presionando la opción “inspeccionar”.



Entrada y Salida de datos

Estas **funciones** las utilizamos para tomar datos de entrada del usuario o para representar datos de salida sin entrometer directamente el HTML.

Hacen uso de mensajes desplegables y la consola del navegador.

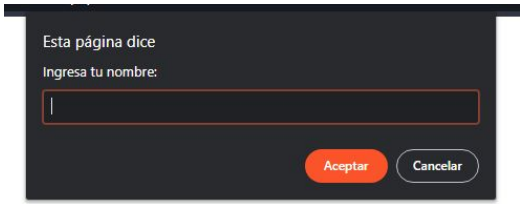
Entrada y salida de información.

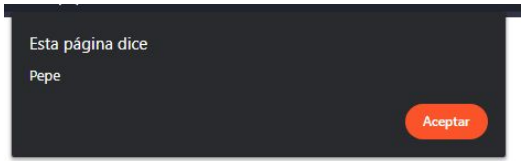
prompt(): despliega un mensaje en la ventana del navegador **con una casilla para ingresar un valor**. El **valor ingresado** será tomado como un string.

alert(): despliega un mensaje en la ventana del navegador con el texto que reciba por parámetro.

console.log(): envía lo que recibe por parámetro a la consola del navegador.

```
1  let entrada = prompt('Ingresa tu nombre: ');
2
3  alert(entrada);
4
5  console.log(entrada);
```





Variables

Una variable es un espacio en memoria reservado para **alojar** información de nuestro programa **durante la ejecución del mismo**.

Necesitan ser declaradas al momento de escribir nuestro programa y **asignadas a un nombre único** que pueda identificarlas.

Variables

Para **declarar una variable** debemos utilizar las palabras reservadas var o let seguidas por el **nombre** que deseamos asignarle.

Existen ciertas restricciones sobre los nombres o caracteres a utilizar:

- El nombre debe contener solo letras, dígitos o los símbolos **$** o **_**
- El **primer carácter** no debe ser un número.
- **No** debe ser una palabra reservada del lenguaje.

```
var miVariable = 'Hola Mundo';

let otrVariable = 50;
```

Nuestras variables son mutables ya que pueden cambiar con el tiempo a diferencia de las **constantes** que no se les puede reasignar un valor.

```
const PI = 3.14;
```

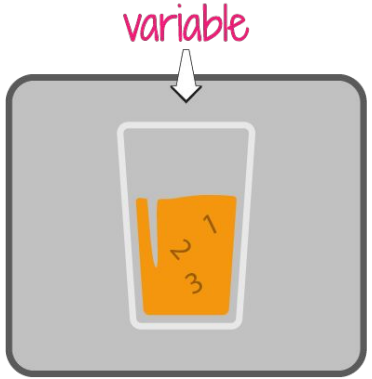
Diferencia entre Var y Let

VAR

Pertenecen **al ámbito o scope global** de nuestro documento, por lo que pueden ser **accedidas** y reasignadas desde cualquier lugar del mismo.

El uso de ésta, puede dar resultados inesperados, por eso, hay que tener cuidado de cómo se usa.

```
var nombreVariable;
var a;
var b;
```



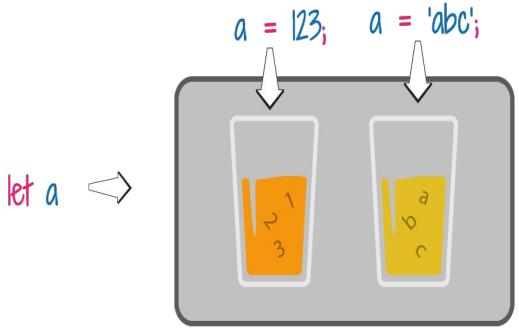
Diferencia entre Var y Let

LET

El **alcance** de estas variables es local, solo pueden ser **accedidas** dentro del bloque donde se definen.

También, permiten que su valor pueda ser reasignado.

```
let nombreVariable = 'texto';
let a = 'abc';
a = 123;
let b = 1;
b = 5;
```

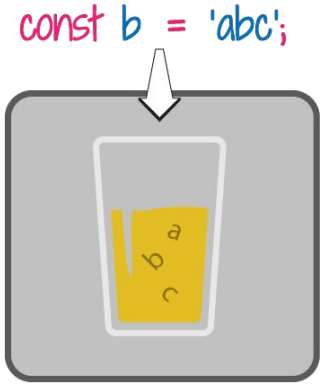


Las constantes

CONST

Solo pueden ser accedidas dentro del bloque donde están definidas, pero no permite que su **valor** sea reasignado, es decir, la **variable** se vuelve inmutable.

```
const nombreVariable = 'texto';
const a = 'Hola Mundo';
const b = 'abc';
const c = 123;
```



Hoisting

En JavaScript se puede hacer **referencia** a una variable que fue declarada más tarde. Este concepto se conoce como **hoisting**.

La declaración de las variables son elevadas a la parte superior del archivo, devolviendo el valor de ***"undefined"***.

Tipos de Datos

Javascript es un **lenguaje débilmente tipado** o de tipado "dinámico" donde una variable no estará atada a un **mismo tipo de dato** como sucede en otros lenguajes, si no que **podemos cambiarlo** durante la ejecución de nuestro programa.

En Javascript ***existen diferentes tipos de datos*** que podemos utilizar, vamos a conocerlos...

Tipos de datos

string: secuencia de caracteres que representan un valor. (**cadena de texto**)

number: valor **numérico**, entero o decimal.

boolean: valores **true** o **false**.

null: valor **nulo**.

undefined: valor **sin definir**.

symbol: tipo de dato cuyos casos son **únicos e inmutables**.

object: colección de datos en un conjunto de **propiedad/valor**.

```
let cadena = 'Hola Mundo';
let numero = 23;
let booleano = true;
let nulo = null;
let indefinido = undefined;
let symbol = Symbol()
let objeto = {
    propiedad: 'valor'
}
```

Métodos de String

Son funciones que nos ayudan a trabajar con nuestras **cadenas de texto**, *transformándolas*, *recortándolas* o simplemente para *saber su extensión*, entre otras cosas.

Funciones / Propiedad	Descripción
string.toUpperCase()	Retorna el mismo texto (string) con las letras en mayúsculas
string.toLowerCase()	Retorna el mismo texto (string) con las letras en minúsculas
string.length	Retorna la cantidad de letras del texto (string)
string.repeat(n)	Retorna un texto repetido n veces
string.replace(str1,str2)	Retorna un texto reemplazando el texto str1 con str2

Parsers

Son **funciones nativas** del lenguaje que nos permiten convertir nuestra información de un tipo de dato a otro tipo de dato distinto.

Por ejemplo, **recibimos** un valor en cadena de texto de “15” pero **necesitamos** usarlo en una función matemática, gracias a los parsers vamos a poder **transformarlo** a un tipo de dato numérico.

Parsers

parseInt() y parseFloat() son funciones creadas para analizar un string y devolver un número si es posible.

JavaScript **analiza la cadena** para extraer las cifras que encuentre al principio, **estas cifras** al principio del string son las que se transforman a tipo numérico.

Cuando se encuentra el primer carácter no numérico se ignora el resto de la cadena.

Si el primer carácter encontrado no es convertible a número, el resultado será NaN (Not a Number).

Number() ignora los espacios al principio y al final, pero, a diferencia de los métodos anteriores, **cuando un string contiene caracteres** no convertibles a números el resultado siempre es NaN, no trata de 'extraer' la parte numérica.

Con Number() **podemos convertir booleanos en números**. False siempre se convierte en 0 y true en 1.

```
let numero = '10a';

parseInt(numero); // 10
Number(numero); // NaN
```

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

Todo en el Aula Virtual.

Gracias