

Agencia de  
Aprendizaje  
a lo largo  
de la vida

# Desarrollo Fullstack



# Les damos la bienvenida

Vamos a comenzar a grabar la clase



## Clase 12

### Javascript

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

## Clase 13

### Javascript

- Operadores Aritméticos
- Operadores Relacionales
- Operadores Lógicos
- Condicionales
- Bucles

## Clase 14

### Javascript

- Funciones
- Expresadas
- Declaradas
- Arrow
- Callbacks
- Scopes

# JS JAVASCRIPT

Lógica a su servicio

# Enlace de un archivo JS

# Asociando un archivo js al html

- Es una práctica común y recomendada colocar la etiqueta `<script>` que enlaza a archivos JavaScript al final del cuerpo (`</body>`) en lugar de en la cabecera (`<head>`) del documento HTML.
- Esto se debe a que el navegador carga y ejecuta los scripts de manera secuencial mientras analiza el documento HTML.
- Colocar los scripts al final del cuerpo asegura que el contenido principal de la página, como texto, imágenes y estilos CSS, se cargue primero antes de que se procesen los scripts.



# <script> al final del <body>

- **Mejora el rendimiento de la página:** El navegador puede cargar primero el contenido visible y hacer que la página sea perceptiblemente más rápida para el usuario. Esto es especialmente importante para páginas con mucho contenido o recursos pesados.
- **Evita bloqueos de renderizado:** Los scripts JavaScript pueden bloquear el proceso de renderizado de la página mientras se están descargando y ejecutando. Colocar los scripts al final del cuerpo permite que el navegador renderice el contenido principal de la página antes de ejecutar los scripts, lo que evita que los scripts bloqueen el renderizado.
- **Mejora la accesibilidad:** Al cargar el contenido principal primero, los usuarios pueden comenzar a interactuar con la página antes de que se carguen y ejecuten los scripts, lo que mejora la experiencia de usuario, especialmente en conexiones lentas o dispositivos con recursos limitados.

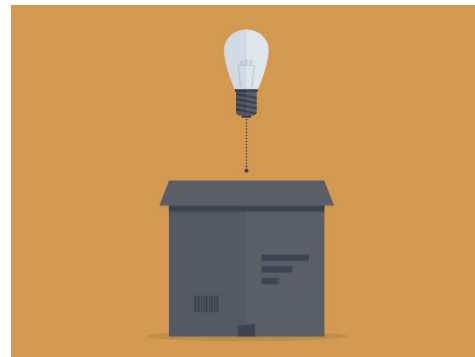


# Variable y declaración de variables



# Variables

- **Una variable** en programación es utilizada de forma habitual.
- Se trata de un elemento fundamental para **gestionar la información** que se decida incorporar en el código y básicamente **funciona para guardar datos**.
- **Una variable almacena un valor único.**



# Características de una variable en programación

1. Una variable **reserva espacio de memoria.**
2. Una variable debe ser **declarada para que exista.**
3. Una variable tiene un **alcance global** es decir puede ser utilizada en todo el proyecto o **local** (ámbito de bloque).



# Declaración de variables: **var**

1. **Ámbito de función:** Las variables declaradas con var tienen un ámbito de función, lo que significa que son accesibles en toda la función en la que se declaran, independientemente de bloques de código. Puede ser asimilada a una variable global. **Actualmente están en desuso.**
2. **Hoisting (elevantar):** Las declaraciones var son "elevadas" al principio de su contexto de ejecución. Esto significa que puedes usar la variable antes de su declaración sin generar un error. Sin embargo, el valor de la variable será undefined hasta que se le asigne un valor.



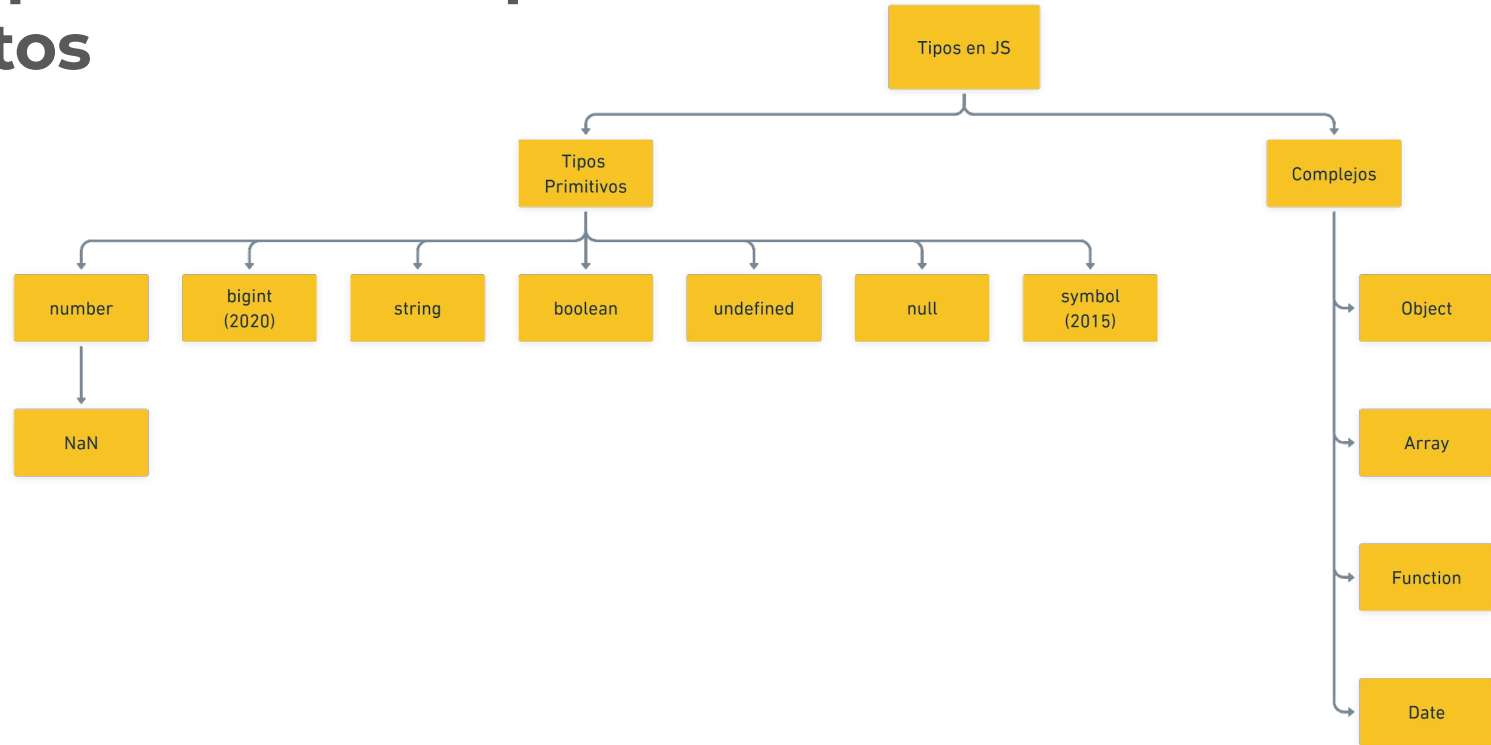
# Declaración de variables: **let**

1. **Ámbito de bloque:** Las variables declaradas con `let` tienen un ámbito de bloque, lo que significa que son válidas solo dentro del bloque en el que se declaran. De esta manera existe mayor control sobre ellas.
2. **No hoisting:** Las variables declaradas con `let` **no son "elevadas"** al principio de su contexto de ejecución. **Deben ser declaradas antes de ser utilizadas.** Lo cual otorga orden al código.



# Tipos de datos

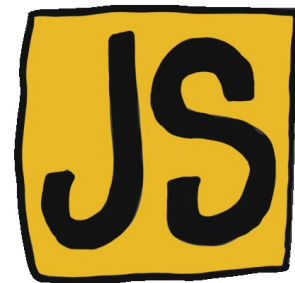
# Esquema de los tipos de datos



# Instrucciones de entrada - salida

# Instrucción de entrada

- Una **instrucción de entrada** nos permite interactuar con la aplicación.
- En JavaScript, las instrucciones de entrada típicamente se realizan para **obtener datos del usuario o del entorno** en el que se ejecuta el código.
- Estos datos **serán almacenados en una variable**.
- Posteriormente estos datos **serán procesados para producir un resultado**.



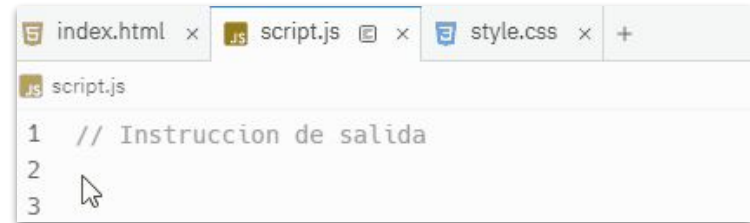


# Instrucción de salida alert( )

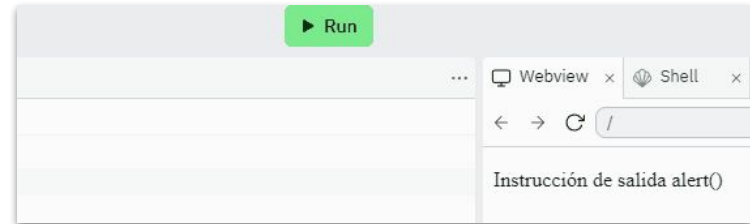
**alert( ):** despliega un mensaje en la ventana del navegador con el texto que reciba por parámetro.

La sintaxis **en JS** es la siguiente:

```
alert("Mensaje_de_alerta");
```



```
index.html x script.js x style.css x +
script.js
1 // Instruccion de salida
2
3
```

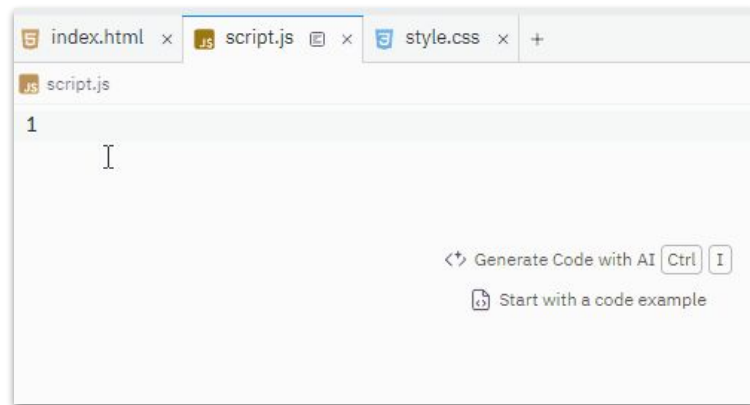


# Instrucción de salida console.log( )

**console.log( ):** imprime mensajes o lo que se le pase por parámetro, en la consola del navegador.

La sintaxis **en JS** es la siguiente:

```
console.log(<mensaje_expresion>);
```



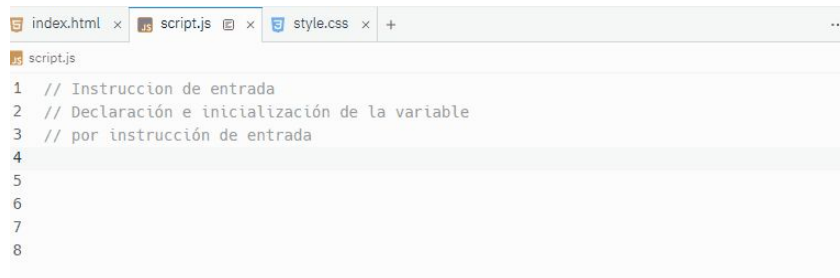
# Instrucción de entrada: prompt( )

**prompt( )**: función que muestra un cuadro de diálogo que solicita al usuario que ingrese información. Retorna el valor ingresado como una cadena (string).

La sintaxis **en JS** es la siguiente:

```
let <identificador> =  
    prompt("Mensaje");
```

- **let** declara una variable.
- **<identificador>** nombre de la variable.



```
index.html x script.js x style.css x +  
script.js  
1 // Instruccion de entrada  
2 // Declaración e inicialización de la variable  
3 // por instrucción de entrada  
4  
5  
6  
7  
8
```

# Operadores

# Operadores

- Los **operadores** nos ayudan a **modificar, reasignar y comprobar** el valor de las variables con una **sintaxis más sencilla** y acotada.



# Operadores de asignación

## Inicialización de variables

# Operador de asignación

- Cuando las variables **no están inicializadas, su valor es indefinido (*undefined*)**.
- Para iniciarlas en un valor, se utiliza el **operador de asignación** en **JS** es el **símbolo =**



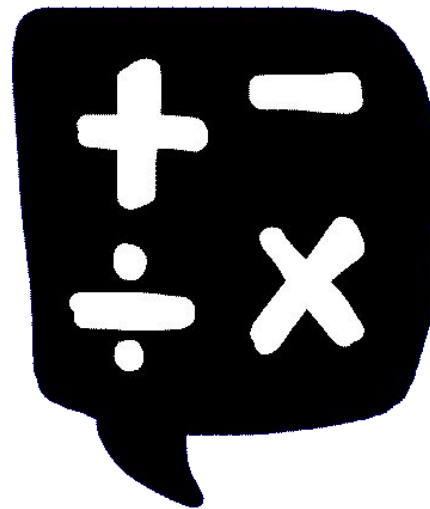
# Operadores Aritméticos

## Operaciones matemáticas



# Operadores aritméticos

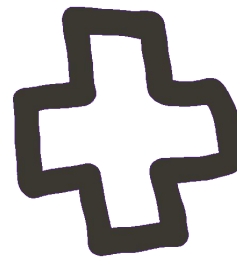
**Consta de los operadores fundamentales de la aritmética** con el agregado de la potencia y el módulo o residuo, que devuelve el resto entero que se produce al realizar un cociente entre dos números enteros.



Operador	Nombre	Ejemplo	Resultado	Descripción
+	Suma	$12 + 3$	15	Devuelve la suma de dos expresiones.
-	Resta	$12 - 3$	9	Devuelve la resta de dos expresiones.
*	Multiplicación	$12 * 3$	36	Devuelve la resta de dos expresiones.
/	División	$12 / 3$	4	Devuelve el cociente de dos expresiones.
**	Potenciación	$12 ** 3$	1728	Devuelve la potencia entre dos expresiones.
%	Módulo o Residuo	$12 \% 3$	0	Devuelve el resto del cociente entre dos enteros.

# Operador de concatenación

- Para hacer más atractivas las salidas, se utiliza el operador de concatenación, en Js existe dos:
  - **El signo mas “+” en general el mas utilizado.**
  - **La coma “,”.**
- Este permite unir cadenas de textos unas con otras o con contenidos de variables **para una mejor lectura en la salida de datos.**
- La **coma aporta un espacio** entre elementos concatenados, **el signo + no lo hace.**
- En ocasiones, por cuestiones de legibilidad del código, será más recomendable la utilización de uno sobre otro.



# Operador de concatenación

- colocar captura

# Operadores de Asignación (autoasignación)

# Operadores de Asignación - Continuación

El operador de asignación simple **es el signo igual (=)**, que asigna el valor de su operando derecho a su operando izquierdo.

Combinados o compuestos, resumen una operación aritmética cuando involucra a la misma variable en la operación:

Anotación	Significado	Ejemplo
=	Asignación	a=5 equivale a decir que a toma el valor 5
+=	Suma a si mismo	a+=5 equivale a decir a = a+5
-=	Resta a si mismo	a-=5 equivale a decir a = a-5
/=	Divide a si mismo	a/=5 equivale a decir a = a/5
*=	Multiplica a si mismo	a*=5 equivale a decir a = a*5

# Operadores de Asignación - Continuación

falta captura

# Operadores Incrementales



# Operadores Incrementales

- Son utilizados especialmente y en particular cuando las variables son incrementadas o decrementadas en 1

**`++a`** que es lo mismo que **`a= a+1;`**

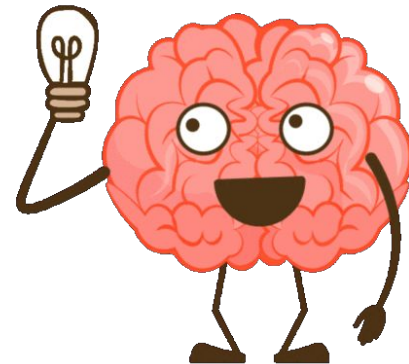
## Operadores incrementales

Anotación	Significado	Ejemplo
<b>++a</b>	Pre incrementa en uno	++a equivale a decir $a = a + 1$
<b>a++</b>	Post incrementa en uno	equivale a decir $a = a$ y luego será $a = a + 1$
<b>--a</b>	Pre decrementa en uno	--a equivale a decir $a = a - 1$
<b>a--</b>	Post decrementa en uno	a-- equivale a decir $a = a$ y luego será $a = a - 1$

# Operadores Relacionales

# Operadores relacionales

- Son aquellos que **permiten comparar expresiones**.
- Si la evaluación es **correcta**, la operación retorna un valor booleano **true** lo contrario, **retorna false**.



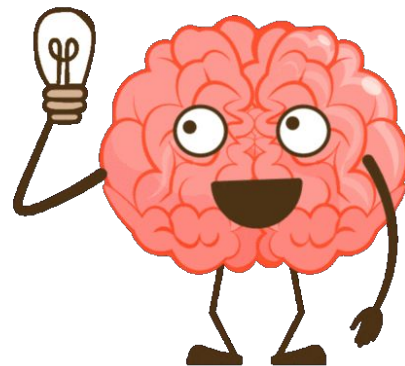
# Tabla de operadores relacionales

OPERADOR	NOMBRE	EJEMPLO	RESULTADO
<	Menor que	4<5	V
<=	Menor o igual que	5<=5	V
>	Mayor que	4>5	F
>=	Mayor o igual que	5>=5	V
==	Igual que	4==5	F
!=	Distinto que (para número)	4!=5	V
.equals( )	igual que (para cadenas)	var1.equals(var2)	

# Operadores Lógicos

# Operadores lógicos

- Con los operadores lógicos pueden realizarse condiciones más complejas que con los relacionales.
- Los operadores lógicos **tienen como “operandos” a los valores booleanos** (true, false).
- Los operadores lógicos **devuelven como resultado de la operación lógica un valor lógico** (true, false).
- Los “operandos” **trabajan de a pares**.



# Tabla de operadores lógicos

Operador	Nombre
	OR ú O lógico
&&	AND ó Y lógico
!	NOT ó NO lógico
??	Operador de Fusión de Nulos
??=	Operador de Asignación Lógica Nula

- Al igual que los operadores relacionales (< ; > ; == ; etc), **permiten operar con expresiones booleanas.**



# Herramientas que utilizamos en clases



VSCode+plugins

# No te olvides de dar el presente

## **Recordá:**

- **Revisar la Cartelera de Novedades.**
- **Hacer tus consultas en el Foro.**
- **Realizá los ejercicios obligatorios.**

**Todo en el Aula Virtual.**