



***¡LES DAMOS LA  
BIENVENIDA!***

¿Están listos?

***RECUERDA PONER A GRABAR LA  
CLASE***

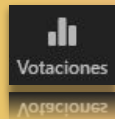


# ***PRESENTACIÓN DE ESTUDIANTES***



Por encuestas de Zoom:

1. País
2. Conocimientos previos en HTML, CSS y programación.
3. ¿Por qué elegiste el curso?





***¿DUDAS DEL ON-BOARDING?***

**MIRALO AQUI**

# ACUERDOS Y COMPROMISOS

## Convivencia:



- ✓ Chequea aquí nuestro [código de conducta](#) y ayúdanos a generar un ambiente de clases súper ameno.
- ✓ Al momento de interactuar (en las clases u otros espacios), ten en cuenta las [normas del buen hablante y del buen oyente](#), que nunca están de más.
- ✓ Durante las clases, emplea los medios de comunicación oficiales para canalizar tus dudas, consultas y/o comentarios: **chat Zoom público y privado y Slack.**
- ✓ Verifica el estado de la **cámara y/o el micrófono** (on/off) de manera que esto no afecte la dinámica de la clase.

# ACUERDOS Y COMPROMISOS

## Distractores:



- ✓ Encuentra tu espacio y crea el momento oportuno para **disfrutar de aprender**.
- ✓ Evita dispositivos y aplicaciones que puedan **robar tu atención**.
- ✓ Mantén la **mente abierta y flexible**; los prejuicios y paradigmas no están invitados.

# ACUERDOS Y COMPROMISOS

## Herramientas:



- ✓ Mantén a tu alcance **agua-mate-café**.
- ✓ Conéctate desde algún equipo (laptop, tablet) que te permita **realizar las actividades** sin complicaciones.
- ✓ Si lo necesitas ubica lápiz y papel para que no se escapen las ideas, sin embargo recuerda que en el **Drive** tienes **archivos** que te ayudarán a repasar, incluidas las **presentaciones**.
- ✓ Todas las clases quedarán grabadas y serán compartidas tanto en la **plataforma de Coderhouse** como por **Google Drive**.

# ACUERDOS Y COMPROMISOS

## Equipo:

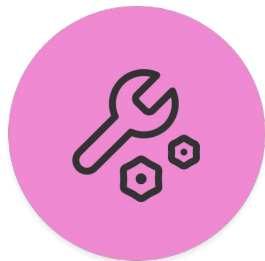


- ✓ **¡Participa de los Afters!** Son un gran espacio para atender dudas y mostrar avances.
- ✓ Intercambia ideas y consultas por el chat de Slack: **entre todos nos ayudamos a mejorar.**
- ✓ Siempre interactúa con otros/as **respetuosamente.**
- ✓ No te olvides de **valorar tu experiencia educativa** y de contarnos cómo te va.



# ***DESAFÍOS Y ENTREGABLES***

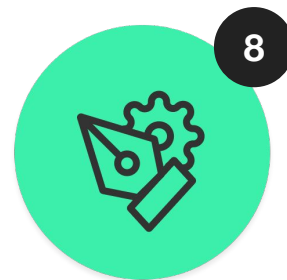
Son actividades o ejercicios que se realizan durante la cursada, para enfocarse en la práctica.



## **Actividad en clase**

### **¡Hands on!**

Ayudan a poner en práctica los conceptos y la teoría vista en clase. No deben ser subidos a la plataforma.



## **Desafíos entregables**

Relacionados completamente con el **Proyecto Final**. Deben ser subidos obligatoriamente a la plataforma hasta 7 días luego de la clase para que sean corregidos.

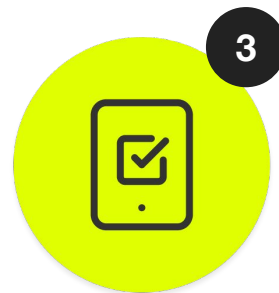
# DESAFÍOS Y ENTREGABLES

Son actividades o ejercicios que se realizan durante la cursada, para enfocarse en la práctica.



## Desafíos complementarios

Desafíos que complementan a los entregables. **Son optativos** y, de ser subidos a la plataforma a tiempo y aprobados, suman puntos para el top 10.



## Entregas del Proyecto Final

Entregas con el estado de avance de tu **proyecto final** que deberás subir a la plataforma a lo largo del curso y hasta 7 días luego de la clase, para ser corregidas por tu docente o tutor/a.



# ***PROYECTO FINAL***

El Proyecto Final se construye a partir de los **desafíos** que se realizan clase a clase. Se va creando a medida que el estudiante sube los desafíos entregables a nuestra plataforma.

El objetivo es que cada estudiante pueda utilizar su Proyecto Final como parte de su portfolio personal.

El **proyecto final** se debe subir a la plataforma la ante-última o última clase del curso. *En caso de no hacerlo tendrás 20 días a partir de la finalización del curso para cargarlo en la plataforma. Pasados esos días el botón de entrega se inhabilitará.*

***¿CUÁL ES NUESTRO PROYECTO FINAL?***



# ***APLICACIÓN WEB INTERACTIVA***

Crearás una página web interactiva **de la temática que elijas** en JavaScript que permitirá simular distintos procesos. Un “simulador” es un programa que soluciona ciertas tareas y proporciona al usuario información de valor. Utilizarás AJAX y JSON para obtener datos y diversas herramientas de JS como librerías, promises, async y frameworks para controlar eventos en la interfaz y producir animaciones en respuesta.



# ***PROYECTOS DE NUESTROS ESTUDIANTES***

- Tienda de PC: <https://bcoalova.github.io/CODER-js-Tienda/>
- Tienda de Bebidas: <https://vanifederici.github.io/mono-galactico/>
- Conversor de divisas: <https://conversor-de-monedas.000webhostapp.com/index.html>
- Buscador de reservar: <https://antopr.github.io/Javascript-Coder/#>
- Simulador de préstamos: <https://mguidocaruso.github.io/AIPrestamo/>

## Grilla de entregas

A continuación verán la grilla de entregas. Cada módulo de trabajo tendrá un desafío entregable y/o pre-entrega.

[illegible]



<b><i>ENTREGA</i></b>	<b><i>REQUISITO</i></b>	<b><i>FECHA</i></b>
<b>1° entrega</b>	Estructura HTML y CSS del proyecto. Variables de JS necesarias. Objetos de JS.	<b>Clase N° 7</b>
<b>2° entrega</b>	Agregar uso de JSON y Storage. DOM y eventos del usuario.	<b>Clase N° 11</b>
<b>Proyecto Final</b>	Simulador final funcionando en un archivo HTML con sus archivos JS complementarios.	<b>Clase N° 16</b>



# ¡IMPORTANTE!

Los desafíos y entregas de proyecto se deben cargar hasta siete días después de finalizada la clase. Te sugerimos llevarlos al día.

LUNES 16/03 20:30HS

10. Estrategia de contenido para Twitter y LinkedIn

DESAFÍO - EXPIRA EL 23/03/2020

Crear publicaciones para Twitter

👍

● HOY 20:30

📁

Tenes tiempo hasta el  
23/03/2020

↑ ENTREGAR



Clase 01. JAVASCRIPT

# ***CONCEPTOS GENERALES: SINTAXIS Y VARIABLES***



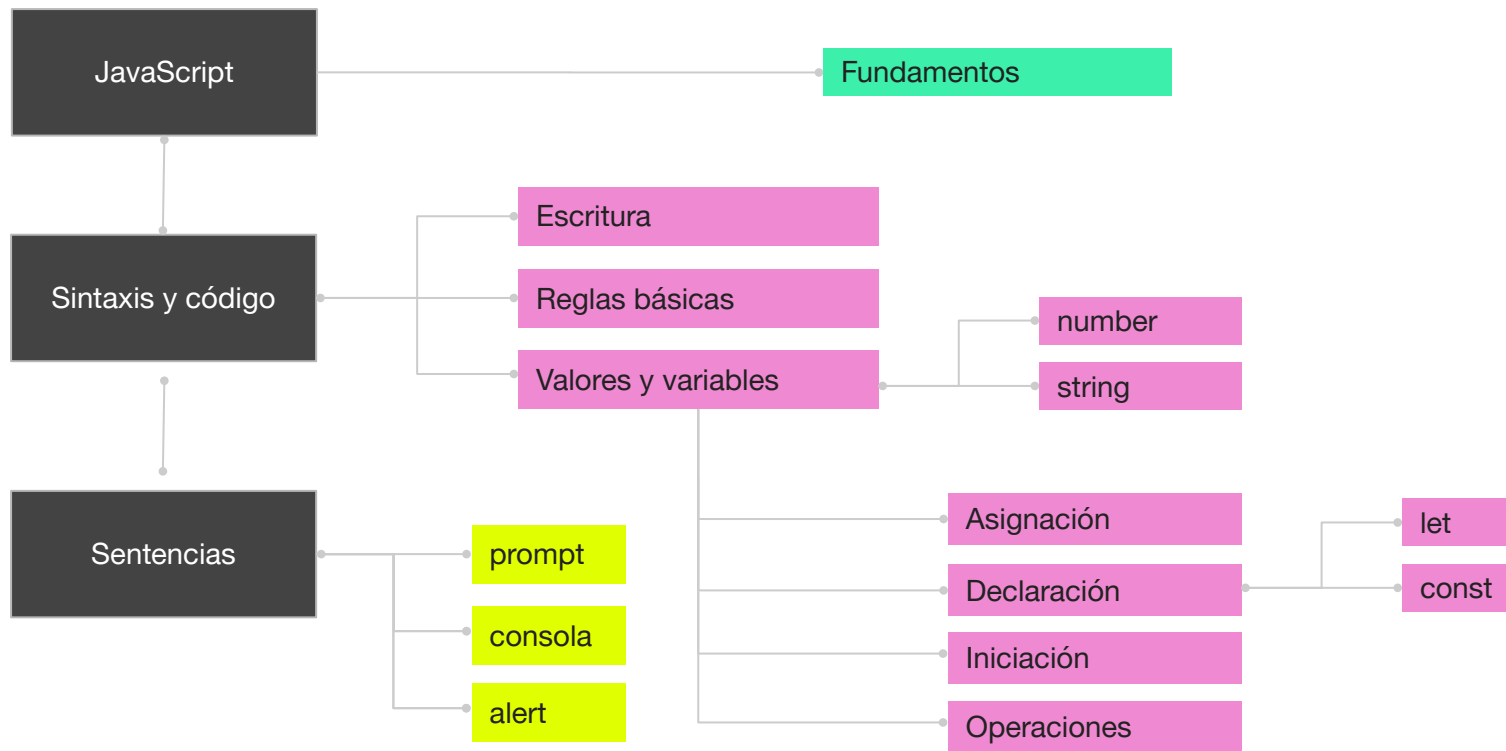
## ***OBJETIVOS DE LA CLASE***

- Conocer los fundamentos del lenguaje **JavaScript**.
- Identificar **código** JavaScript y sintaxis.
- Aprender qué es una **variable** y cómo declararla.
- Comprender cómo asignar y cambiar el **valor** de una variable.

# ***MAPA DE CONCEPTOS***

# MAPA DE CONCEPTOS CLASE 1

¡Para  
recordar!



# MÓDULOS DE TRABAJO

## MÓDULO 0 NIVELACIÓN

**CLASE 0 -**  
INTRODUCCIÓN A JAVASCRIPT

## MÓDULO 1 CONCEPTOS BÁSICOS

**CLASE 1 -**  
CONCEPTOS GENERALES:  
SINTAXIS Y VARIABLES

**CLASE 2 -**  
CONTROL DE FLUJOS

**CLASE 3 -**  
CICLOS E ITERACIONES

**CLASE 4 -**  
FUNCIONES

- **Desafío entregable**

## MÓDULO 2 OBJETOS & ARRAYS

**CLASE 5 -**  
OBJETOS

**CLASE 6 -**  
ARRAYS

**CLASE 7 -**  
FUNCIONES DE ORDEN SUPERIOR

- **1ra pre-entrega**



# ***HERRAMIENTAS DE LA CLASE***

*Les compartimos algunos recursos para acompañar la clase*

- Guión de clase N° 1 [aquí](#).
- Quizz de clase N° 1 [aquí](#)
- Booklet de Javascript [aquí](#)
- FAQs de Javascript [aquí](#)

# ***EN LA CLASE ANTERIOR...***

- Vimos que es Javascript.
- Conocimos su historia.
- Lo diferenciamos de Java.
- Entendimos la relación con HTML y CSS.
- Introducimos las herramientas de curso.
- Y vimos cuáles serán los contenidos de todo el curso.





***¿Empezamos?***



***CODER HOUSE***

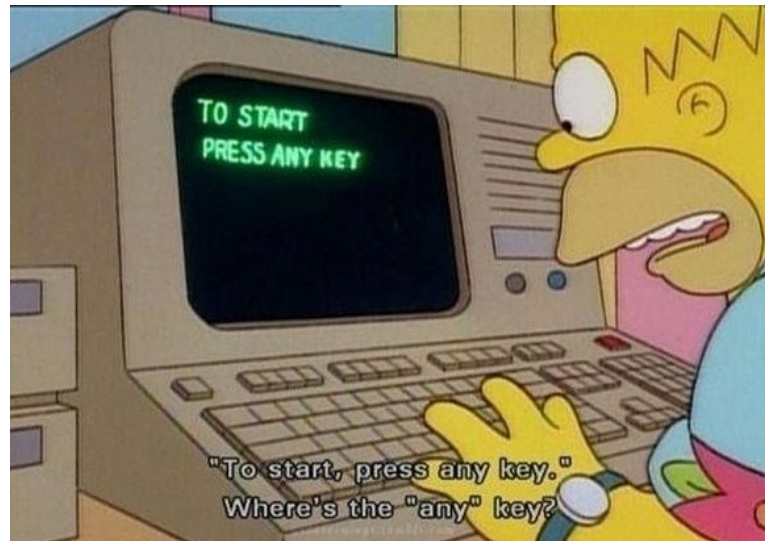
# ***SINTAXIS Y CÓDIGO***

# ***CÓDIGO JAVASCRIPT***



JavaScript tiene sus propias reglas para la sintaxis, aunque respeta los estándares de muchos lenguajes de programación lógicos.

Nuestro código JavaScript se asocia al archivo HTML que se carga en el navegador.  
**Tenemos dos maneras de escribir código JavaScript en nuestras aplicaciones web.**



1

# ¿CÓMO ESCRIBIR CÓDIGO JS?



- Dentro de un archivo html, entre medio de las etiquetas **<script>**

Ejemplo:

```
<script>  
    // Aquí se escribe el código JS  
</script>
```

# ¿CÓMO ESCRIBIR CÓDIGO JS?



- En un archivo individual con **extensión .js**

Ejemplo: mi-archivo.js

Se vincula con la etiqueta `<script>` y el atributo “**src**”

*Recuerda no utilizar espacios ni mayúsculas en los nombres de archivo.*

```
<script src="js/main.js"></script>
```

# ***SINTAXIS: REGLAS BÁSICAS***



- No se tienen en cuenta los espacios en blanco y las nuevas líneas (al igual que HTML).
- Case sensitive: se distingue mayúsculas de minúsculas.
- Se pueden incluir bloques de comentarios:

```
<script>  
    // Comentario simple: una línea  
    /* Comentario de más de una línea I  
       Comentario de más de una línea II */  
</script>
```



# ***SINTAXIS: PALABRAS RESERVADAS***

- Palabras reservadas: son aquellas que se utilizan para construir las sentencias de JavaScript y que por tanto no pueden ser utilizadas libremente.

Algunos ejemplos son:

`break, case, catch, continue, default, let  
delete, do, else, finally, for, function, if, in,  
instanceof, new, return, switch, this, throw, try,  
typeof, var, void, while, with, y varias más`

# ***VARIABLES Y VALORES***



# VARIABLES

Una **variable** es un espacio reservado en la memoria que, como su nombre indica, **puede cambiar de contenido a lo largo de la ejecución de un programa.** En las variables almacenamos diversos tipos de datos que utilizamos en la aplicación.

```
<script>
  //Declaración de variable ES5.
  var nombre = "John";

  //Declaración de variable ES6.
  let apellido = "Doe";
  const CURSO= "JavaScript";
</script>
```

# DECLARACIÓN

Declarar una variable significa **crearla**. Para esto usamos la palabra reservada `var`, `let` o `const`. Escribimos una de estas palabras claves seguido del nombre de nuestra variable. Para los nombres no debemos utilizar ni espacios ni caracteres especiales.

```
// correcto
let apellido
let telefono
const anioNacimiento

// incorrecto
var año
var teléfono móvil
```

# VALORES

Podemos asociar distintos valores a una variable en JavaScript. Para empezar trabajaremos con los tipos de datos más comunes, que son las cadenas de texto y los números:

- **Number:** un valor numérico puede ser entero o decimal.
- **String:** por otro lado, un *string* o cadena de texto, es un valor compuesto por uno o más caracteres, definido entre comillas simples o dobles.

# ASIGNACIÓN

En una variable podemos asignar distintos tipos de valores mediante el *operador de asignación*, que es el símbolo **=**

```
// declaración
let edad
let nombre
let apellido
// asignación
edad = 5
nombre = "John R."
apellido = 'Doe'
```

# ***INICIALIZAR VARIABLES***

Podemos declarar una variable y asignarle un valor inicial en el mismo proceso:

```
// variables inicializadas  
let edad = 5  
const nombre = "John R."  
const apellido = 'Doe'
```

# LET Y CONST

Las declaraciones con **let** y **const** tienen controles adicionales para las variables. Principalmente impiden que se puedan crear dos variables con el mismo nombre.

Una variable **let** puede recibir múltiples asignaciones en el transcurso de la aplicación, es decir que puede cambiar de valor varias veces. Una constante **const** recibe una única asignación al momento de su declaración, impidiendo que su valor se modifique luego.

```
let edad = 5
```

```
edad = 34
```

```
const apellido = 'Doe'
```

```
// no es posible cambiarlo
```

```
apellido = 'Trump'
```

---

# ***OPERACIONES BÁSICAS***

Con variables de valores numéricos puedes realizar operaciones matemáticas: sumas, restas, multiplicaciones, etc.

```
let  numeroA = 1;
let  numeroB = 2;
const NUMEROC = 3;
//Suma de dos números (1 + 2 = 3)
let resultadoSuma = numeroA + numeroB;
//Resta de dos números (2 - 1 = 1)
let resultadoResta = numeroB - numeroA;
//Producto de dos números (2 * 3 = 6)
let resultadoProducto = numeroB * NUMEROC;
```

# OPERACIONES BÁSICAS

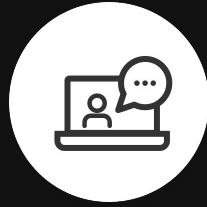
Con variables de tipo string (texto) se puede *concatenar* los valores. Es decir, combinarlas.

```
let textoA = "CODER";
let textoB = "HOUSE";
const BLANCO = " ";
//Concatenar textoA y textoB ("CODER" + "HOUSE" = "CODERHOUSE")
let resultadoA = textoA + textoB;
//Concatenar textoB y 1 ("HOUSE" + 1 = "HOUSE1")
let resultadoB = textoB + 1;
//Concatenar textoA, BLANCO y textoB ("CODER" + " " + "HOUSE" = "CODER HOUSE")
let resultadoC = textoA + BLANCO + textoB;
```





***¡HANDS ON!***



## ***EJEMPLO EN VIVO***

*¡Llevemos lo visto hasta el momento a la acción!  
Vamos a concentrarnos en sus características y funcionamientos  
principales.*



***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**

***PROMPT, CONSOLA Y ALERT***

# ***PROMPT***

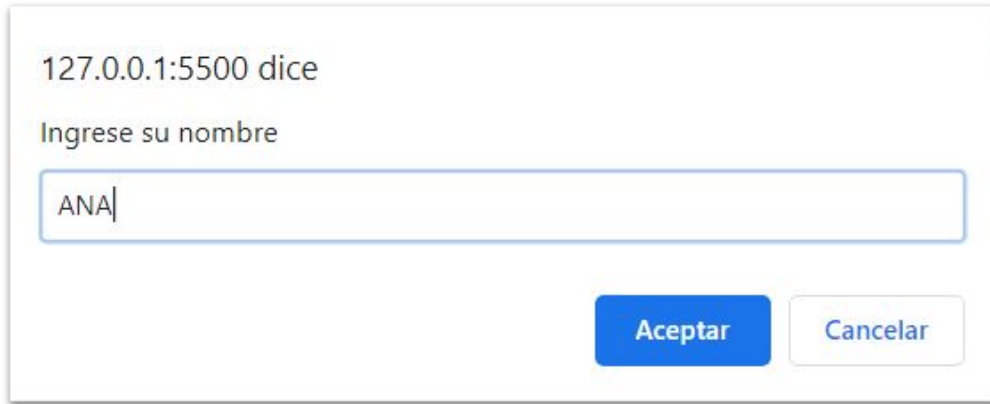
La sentencia **prompt()** mostrará un cuadro de diálogo para que el usuario ingrese un dato. Se puede proporcionar un mensaje que se colocará sobre el campo de texto. El valor que devuelve es una cadena que representa lo que el usuario ingresó.

```
let nombreIngresado = prompt("Ingrese su nombre");
```

## ***EJEMPLO DE PROMPT***

En la pantalla del navegador, el usuario verá una ventana sobre la web que le solicitará un dato.

Al valor que el usuario ingresa se lo conoce por el término de ***entrada***.



A screenshot of a web browser dialog box. The dialog box has a light gray border and a white background. At the top left, it says "127.0.0.1:5500 dice". Below that, it says "Ingrese su nombre". There is a text input field with a blue border containing the text "ANA". At the bottom right, there are two buttons: a blue button labeled "Aceptar" and a white button labeled "Cancelar".

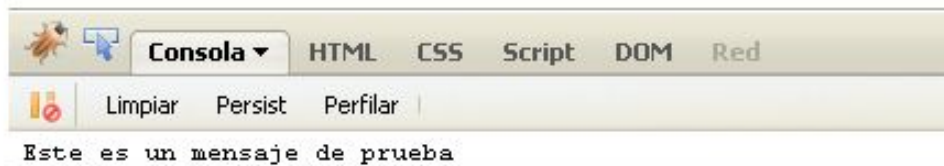
# ***CONSOLA***

La sentencia **console.log()** muestra el mensaje que pasemos como parámetro a la llamada en la consola JavaScript del Navegador web.

```
console.log("Mensaje de prueba");
```

```
let nombre = "John Doe"
```

```
console.log(nombre)
```



## ***EJEMPLO DE CONSOLE.LOG***

En Chrome, la consola del navegador está disponible accediendo mediante:

*Botón derecho sobre alguna  
parte de la web > Inspeccionar  
> Consola*



# ***ALERT***

La sentencia **alert()** mostrará una ventana sobre la página web que estemos accediendo mostrando el mensaje que se pase como parámetro a la llamada.

```
alert("¡Hola Coder!");
```

## ***EJEMPLO DE ALERT***

En la pantalla del navegador, el usuario verá una ventana sobre la web que muestra un mensaje.

Al valor que mostramos al usuario como un resultado se lo conoce por el término de ***salida***.

127.0.0.1:5500 dice

¡Hola Coder!

Aceptar

# ***EJEMPLO DE SCRIPT COMPLETO***

Este es un ejemplo de un Script JS corriendo en un archivo HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi primer App - CoderHouse</title>
    <script>
      let entrada = prompt("Ingresar una letra");
      let salida  = entrada + " " + "ingresada";
      alert(salida);
    </script>
  </head>
  <body>
    <h2>Esta página contiene una app</h2>
  </body>
</html>
```

***Si ingreso "A"...***



127.0.0.1:5500 dice

Ingresar una letra

**Aceptar** Cancelar

***Obtengo...***



127.0.0.1:5500 dice

A ingresada

**Aceptar**

***¡HANDS ON!***



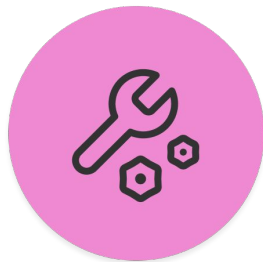
***CODER HOUSE***



## ***ACTIVIDAD EN CLASE***

*¡Llevemos lo visto hasta el momento a la acción!*  
*Les proponemos que en salas de zoom lideradas por su tutor/a*  
*puedan realizar la siguiente actividad.*

***Tiempo estimado 25/30 minutos***



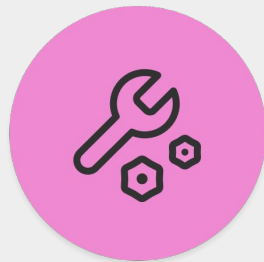
# ***CREAR UN ALGORITMO JS SIMPLE***

Crea un script en JS que le solicite al usuario ingresar datos y luego, mediante JavaScript, realiza operaciones sobre los mismos.

## CREAR UN ALGORITMO JS SIMPLE

**Formato:** Código fuente en archivo JavaScript, formato .js, vinculado al archivo HTML correspondiente.

**Sugerencia:** Usamos `prompt()` para solicitar datos al usuario y `console.log()` o `alert()` para mostrar el resultado de las operaciones realizadas con esos datos. Si vas a sumar una entrada, tené en cuenta que tenés que parsearla antes. Usando `parseInt()` o `parseFloat()`



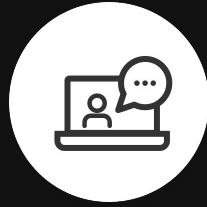
**>> Consigna:** Crea un script en JS que le solicite al usuario ingresar mínimo 1(un) dato. Luego, con JavaScript, realiza operaciones matemáticas o de concatenación sobre las entradas teniendo en cuenta el tipo de dato. Al finalizar mostrar el resultados con `alert()` o `console.log()`



## ***CREAR UN ALGORITMO JS SIMPLE***

### **>>Elegir un tipo de pedido. Por ejemplo:**

- Pedir nombre mediante prompt y mostrarlo en consola junto con algún texto de saludo.  
Ejemplo: ¡Hola, Juan!
- Pedir un número mediante prompt, parsearlo, sumarlo a otro que se encuentre almacenado en una variable y luego mostrar el resultado en consola.
- Pedir un texto mediante prompt, luego otro, concatenarlos y mostrarlo en un alerta.



# ***ACTIVIDAD EN CLASE***

*Puesta en común del ejercicio*

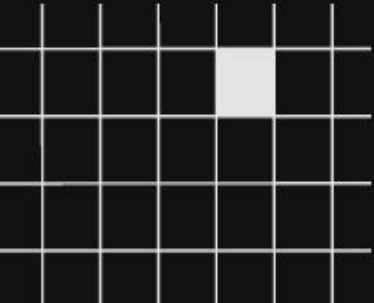
***¿PREGUNTAS?***





# ***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- Fundamentos de desarrollo con JS.
    - ¿Cómo escribir JavaScript?
    - Declaración de Variables.
  - Funciones de prompt, alert y console.
- 



***TE INVITAMOS A QUE COMPLEMENTES  
LA CLASE CON LOS SIGUIENTES  
CODERTIPS***



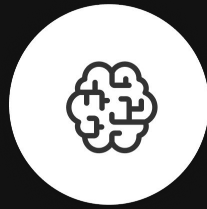
# ***VIDEOS Y PODCASTS***

- [Aprende Programación Web y construye el futuro de nuestra humanidad](#) | **Coderhouse**
- [Desarrollo freelance](#) | **Coderhouse**
- [Desarrollo profesional](#) | **Coderhouse**



# ***VIDEOS Y PODCASTS***

- [CoderNews](#) | **Coderhouse**
- [Serie de Branding](#) | **Coderhouse**
- [Serie para Emprendedores](#) | **Coderhouse**
- [Serie Aprende a Usar TikTok](#) | **Coderhouse**
- [Serie Finanzas Personales](#) | **Coderhouse**
- [CoderConf](#) | **Coderhouse**



## ***¡PARA PENSAR!***

*¿Te gustaría comprobar tus conocimientos de la clase?*

Te compartimos a través del chat de zoom  
el enlace a un breve quiz de tarea.

*Para el profesor:*

- *Acceder a la carpeta "Quizzes" de la camada*
  - *Ingresar al formulario de la clase*
  - *Pulsar el botón "Invitar"*
  - *Copiar el enlace*
- *Compartir el enlace a los alumnos a través del chat*





# ***RECURSOS:***

- Consola, variables y tipos de datos |  
***Los apuntes de Majo (Página 1 a 8).***
- Variables, valores y referencias |  
***Te lo explico con gatitos.***
- Práctica interactiva sobre Algoritmia |  
***La aventura del punto.***
- Herramienta recomendada |  
***Visual Studio Code.***



***¿YA CONOCES LOS BENEFICIOS QUE TIENES  
POR SER ESTUDIANTE DE CODERHOUSE?***

# ***BENEFICIOS*** 🧐

Haz clic [aquí](#) y conoce todos nuestros beneficios exclusivos para estudiantes de Coderhouse.



***OPINA Y VALORA ESTA CLASE***

***#DEMOCRATIZANDO LA EDUCACIÓN***

***CODER HOUSE***

***CODER HOUSE***

***¡GRACIAS POR ESTUDIAR CON  
NOSOTROS!***

---