

Esta clase va a ser

- grabada



COMISIÓN N°####

# Presentación del equipo

✓ Profesor/a responsable: Coordinador/a:

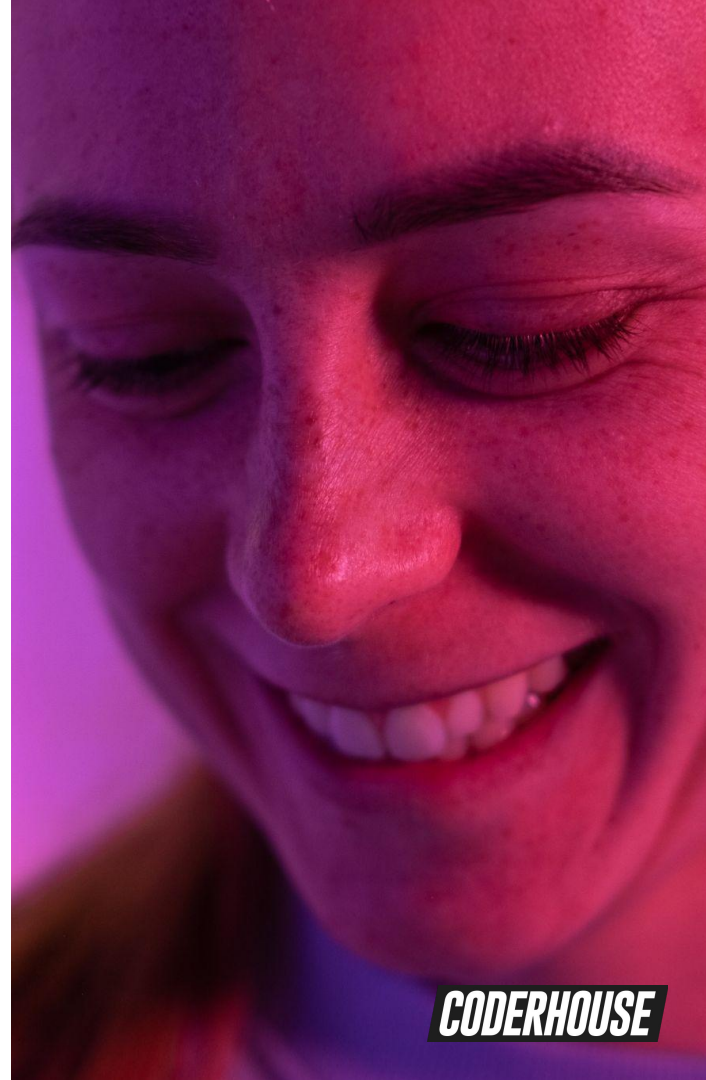
✓ Tutores y tutoras:

- ...
- ...
- ...
- ...
- ...
- ...
- ...

# Presentación de estudiantes

Por encuestas de Zoom

1. País
2. Conocimientos previos
3. ¿Por qué elegiste este curso?



**CODERHOUSE**

# ¿Dudas sobre el onboarding?

Mira los vídeos de  
Onboarding en la  
Plataforma



Lo que debes saber  
**antes de empezar**

# Acuerdos y compromisos

## ACUERDOS Y COMPROMISOS

# Convivencia

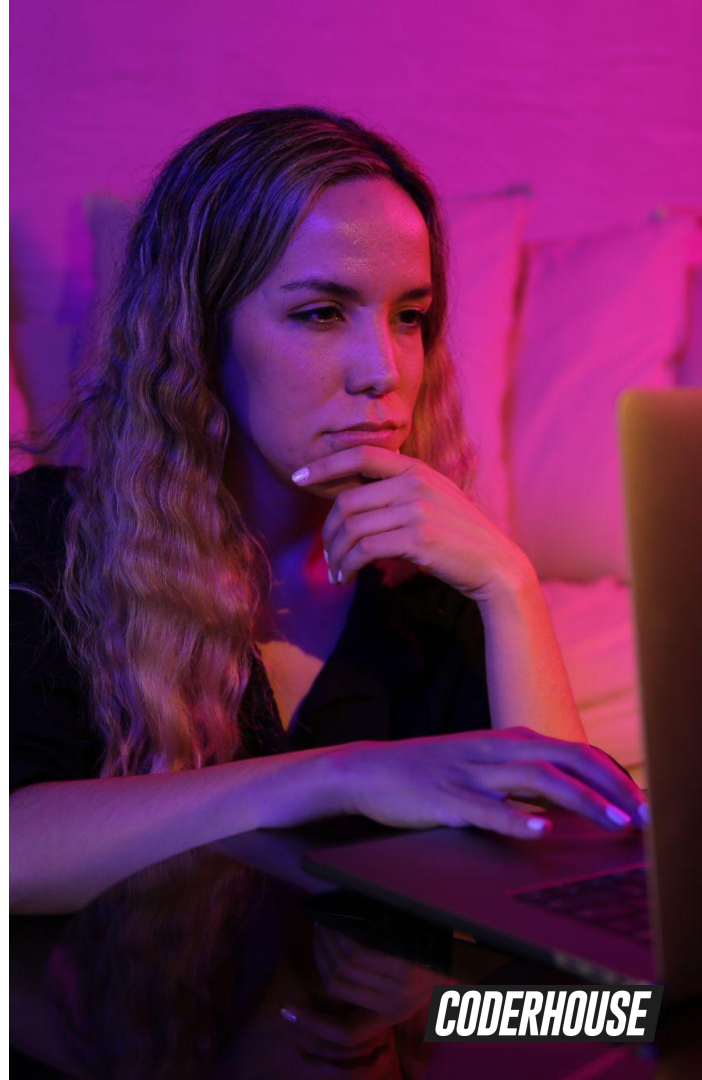
- ✓ Conoce aquí nuestro [código de conducta](#) y ayúdanos a generar un ambiente de clases súper ameno.
- ✓ Durante las clases, emplea los medios de comunicación oficiales para canalizar tus dudas, consultas y/o comentarios: **chat Zoom público y privado, y por el chat de la plataforma.**
- ✓ Ten en cuenta [las normas del buen hablante y del buen oyente](#), que nunca están de más.
- ✓ 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



**CODERHOUSE**



## ACUERDOS Y COMPROMISOS

# Herramientas

- ✓ Mantén a tu alcance **agua, mate o café**
- ✓ Si lo necesitas, ten a mano lápiz y papel para que no se escapen las ideas. Pero recuerda que **en Google Drive tienes archivos que te ayudarán a repasar, incluidas las presentaciones.**
- ✓ Conéctate desde algún equipo (laptop, tablet) que te permita **realizar las actividades** sin complicaciones.
- ✓ 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 After Class!** Son un gran espacio para atender dudas y mostrar avances.
- ✓ Intercambia ideas por **el chat de la plataforma.**
- ✓ Siempre **interactúa respetuosamente.**
- ✓ No te olvides de **valorar tu experiencia** educativa y de contarnos cómo te va.

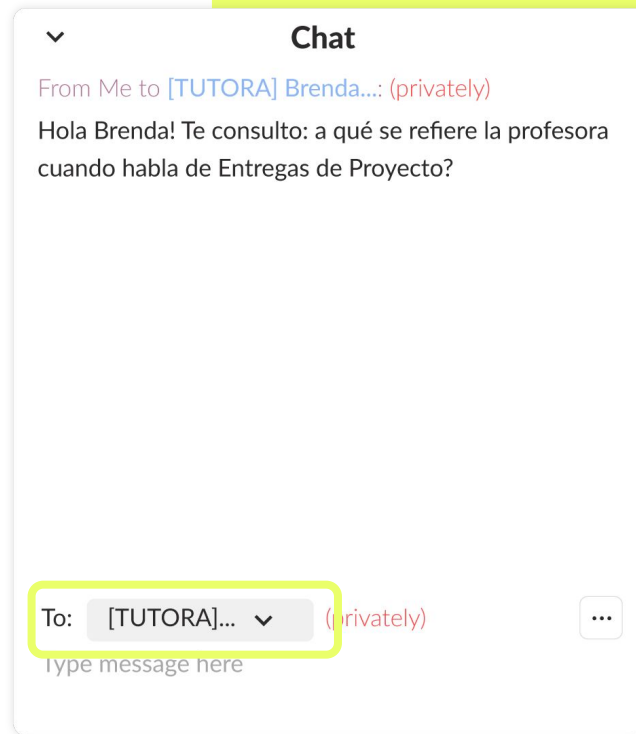
# Interacciones en clase

## INTERACCIONES EN CLASE

# Mientras el profesor explica

Para mantener una comunicación clara y fluida a lo largo de la clase, te proponemos mantener 2 reglas:

1. Si tienes dudas durante la explicación, debes consultarle directamente por privado a tu tutor por el chat de Zoom.

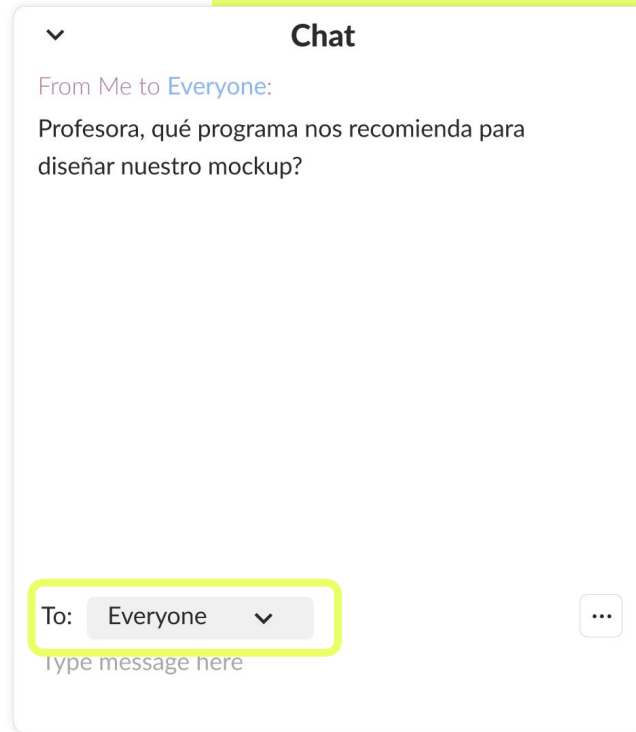


# Espacios para consultas

2. Entre contenido y contenido, se abrirán breves espacios de consulta. Allí puedes escribir en el chat tu pregunta.

**¡Tu duda puede ayudar a otras personas!**

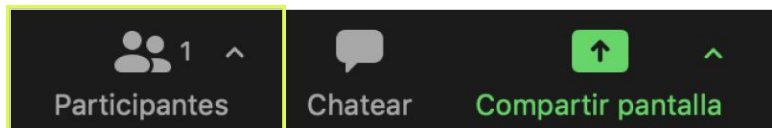
No olvides seleccionar “todos” para que todos puedan leerte (y no solo tu tutor).



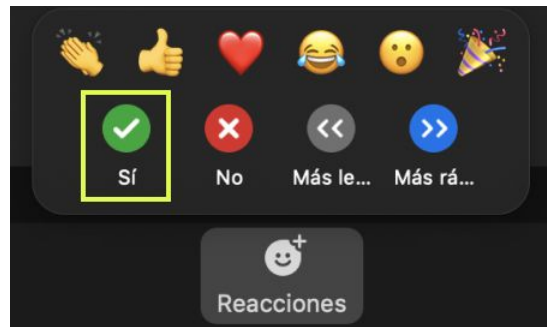
## INTERACCIONES EN CLASE

# Funcionalidades

Para **evitar saturar el chat de mensajes**, utiliza los signos que figuran en el apartado **Participantes**, dentro de Zoom.\*\*



**Por ejemplo:** si se pregunta si se escucha correctamente, debes seleccionar la opción "Sí" o "No".



\*\*Para quitar el signo, presiona el mismo botón nuevamente o la opción "clear all".

**After Class**



AFTER CLASS

# ¿Qué son?

Te acompañamos para resolver tus consultas sobre el contenido en estos espacios.

Si hay **temas que no se entendieron o necesitan refuerzo** se trabajarán en una clase de 1 hs que opera como **espacio de consulta**.

No son obligatorias ni se toma asistencia, pero son el espacio uno a uno con tu profesor/a para responder dudas puntuales o reforzar conceptos.

Tu profesor/a está comprometido con tu educación, por lo tanto:

- ✓ Se responderán **dudas puntuales** que hayan quedado sobre los temas dados. ¡Vení preparado, queremos escucharte!
- ✓ Se verán temas de **conocimientos básicos** para la nivelación de saberes.



**¿Cuál es nuestro  
Proyecto final?**



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.



## PROYECTO FINAL

# Proyectos de estudiantes

Les compartimos los proyectos finales de estudiantes de este curso de comisiones anteriores.

- ✓ [Tienda de PC](#)
- ✓ [Tienda de Bebidas](#)
- ✓ [Conversor de divisas](#)
- ✓ [Buscador de reservas](#)
- ✓ [Simulador de préstamos](#)



**¡Esperamos que les resulten inspiradores!**



## PROYECTO FINAL

Entrega	Requisito	Fecha
1º entrega	Simulador interactivo.	Clase N° 4
2º entrega	Estructura, variables y objetos.	Clase N° 8
3º entrega	Optimización del proyecto.	Clase N° 12
Proyecto Final	Proyecto final.	Clase N° 16

# ¡Importante!

Las pre entregas del proyecto final se deben cargar hasta **siete días** después de finalizada la clase. Y contarás con **10 días** una vez finalizado el curso para entregar tu proyecto final.  
Te sugerimos llevarlos al día.



MARTES 25/01 19:00HS - VALORACIÓN REQUERIDA

## 2. Metodologías de diseño y UX research

DESAFÍO - EXPIRA EL MARTES 01/02/2022 23:59HS

Definiendo el problema, objetivo y solución

Se bloqueará en 7 días y 11:48hs luego no se podrá entregar.

↑ Entregar

Clase 01. JAVASCRIPT

# Conceptos generales: sintaxis y variables



# Temario

00

## Introducción a JavaScript

- ✓ Aplicaciones web
- ✓ JavaScript
- ✓ Herramientas del curso

01

## Sintaxis y variables

- ✓ Sintaxis y código
- ✓ Variables y Valores
- ✓ Prompt, consola y alert

02

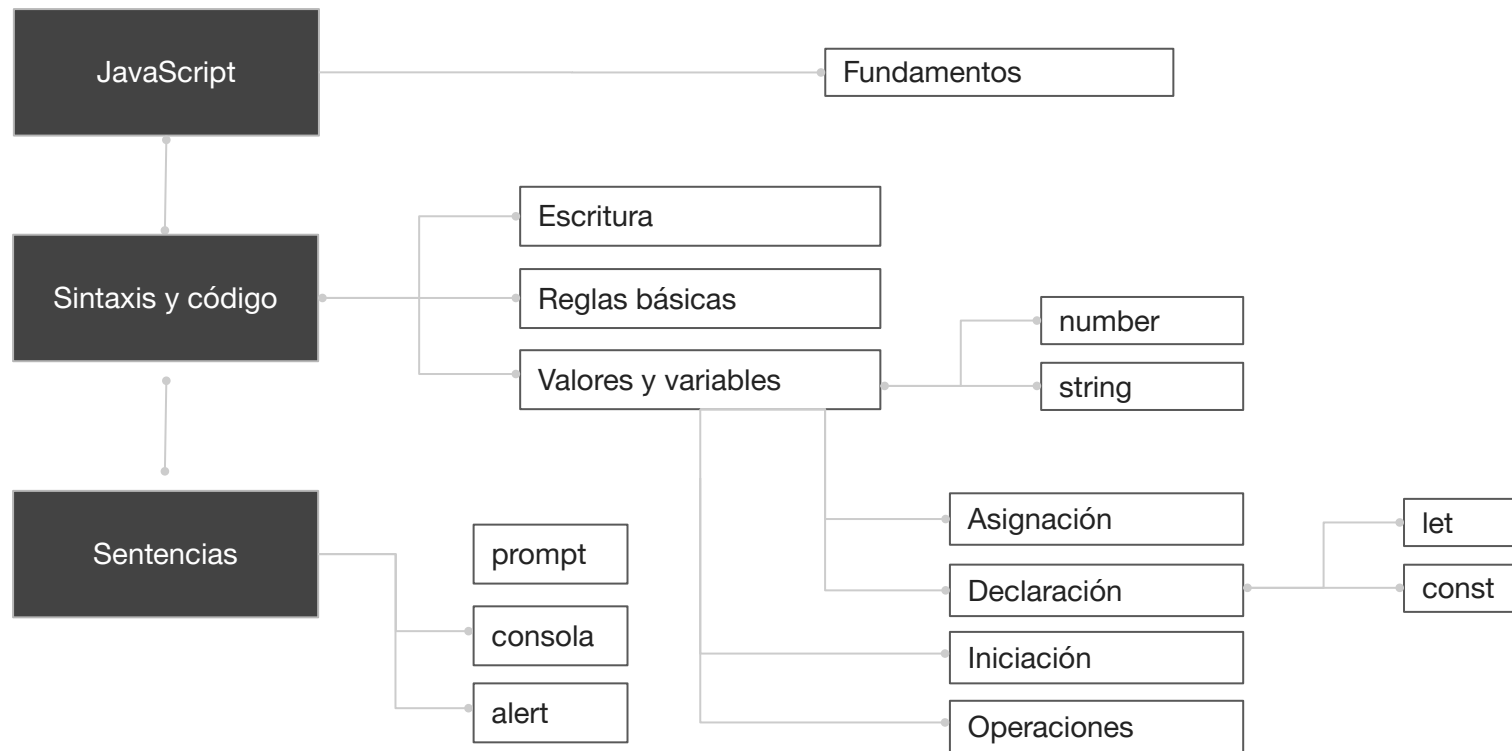
## Control de flujos

- ✓ Condicionales en JS
- ✓ Operadores lógicos

# Objetivos de la clase

- **Conocer** los fundamentos del lenguaje Javascript
- **Identificar** código Javascript y su sintaxis
- **Aprender** qué es una variable y cómo presentarla
- **Comprender** cómo asignar y cambiar el valor de una variable

## MAPA DE CONCEPTOS





# Repaso

En la clase anterior vimos:

- ✓ ¿Qué es JavaScript? Definición e historia.
- ✓ Diferencias entre JavaScript y Java.
- ✓ Relación con HTML y CSS.

# ¿Empezamos?



# 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.**





# ¿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

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

## 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

# Sintaxis

## Palabras Reservadas

Son aquellas que se utilizan para construir las sentencias de JavaScript y que por tanto no pueden ser utilizadas libremente.

```
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

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

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

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.

# 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, los cuales 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

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

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

# 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

```
let edad = 5
```

```
edad = 34
```

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

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

```
apellido = 'Trump'
```

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.

# Operaciones básicas

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

```
let numeroA = 1;
let numeroB = 2;
const NUMERO_C = 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 * NUMERO_C;
```

# 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;
```



## Ejemplo en vivo

¡Llevemos lo visto hasta el momento a la acción!

Vamos a concentrarnos en sus características y funcionamientos principales.



# Break

¡10 minutos y volvemos!



Prompt, consola y alert

# Prompt

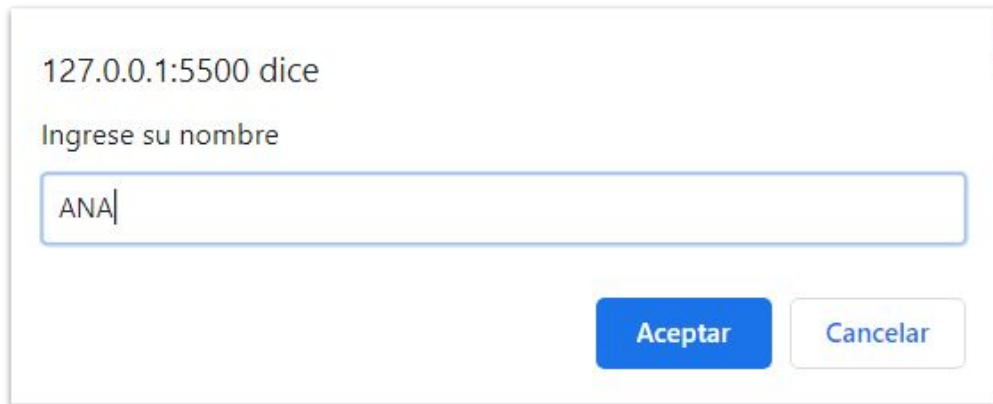
La sentencia **prompt()** mostrará un cuadro de diálogo para que el usuario ingrese un dato. Se puede proporcionar un mensaje que se colocará sobre el campo de texto. El valor que devuelve es una cadena que representa lo que el usuario ingresó.

```
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**.



127.0.0.1:5500 dice

Ingrese su nombre

Aceptar 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");
```

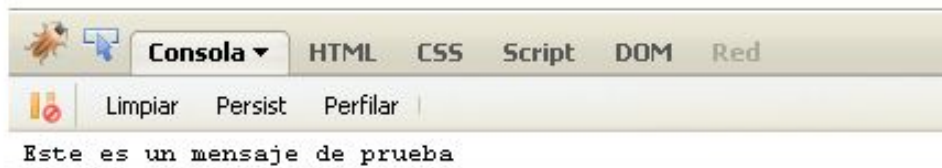
```
let nombre = "John Doe"
```

```
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

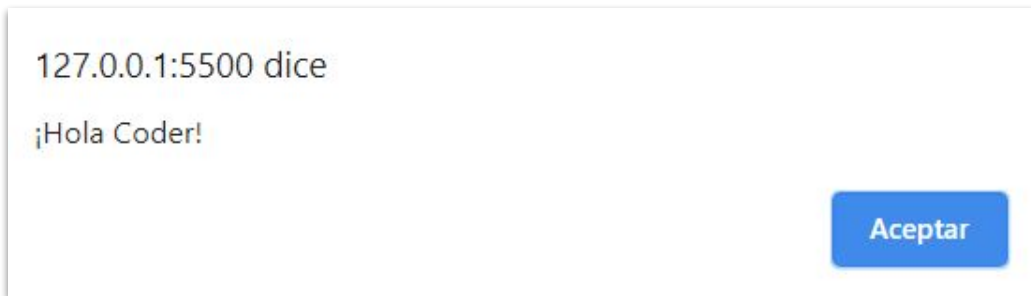
La sentencia **alert()** mostrará una ventana sobre la página web que estemos accediendo mostrando el mensaje que se pase como parámetro a la llamada.

```
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**.



# 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

A|

Aceptar

Cancelar

Obtengo...

127.0.0.1:5500 dice

A ingresada

Aceptar



# Crear un algoritmo JS Simple

Crea un script en JS que le solicite al usuario ingresar mínimo 1 dato y luego, mediante JavaScript, realiza operaciones sobre los mismos.

Duración: **25/30 minutos**



ACTIVIDAD EN CLASE

# Crear un algoritmo JS Simple

**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()`

**Formato:**

Código fuente en archivo JavaScript, formato `.js`, vinculado al archivo HTML correspondiente



## ACTIVIDAD EN CLASE

# Crear un algoritmo JS Simple

### 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()`

### Ejemplos:

- ✓ 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.

¿Preguntas?



**Completa esta 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
- ✓ [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





MATERIAL AMPLIADO

# Recursos multimedia

- ✓ [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.

Disponible en nuestro repositorio.

# Resumen de la clase hoy

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

**Opina y valora**  
**esta clase**

**#DemocratizandoLaEducación**

**Muchas gracias.**